

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		1

- 1** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`  
Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":
- A** `foo(2);foo(0);foo(5);foo(7);foo(8);`
  - B** `foo(1);foo(4);foo(6);foo(10);foo(8);`
  - C** `foo(5);foo(2);foo(5);foo(10);foo(7);`
  - D** `foo(2);foo(3);foo(10);foo(7);`
  - E** žádná z odpovědí není správná

- 2** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - C** vytvoří pole o celkovém počtu 64 prvků typu `int`
  - D** žádná z odpovědí není správná
  - E** způsobí zápis za konec přidělené paměti
  - F** vytvoří pole o celkovém počtu 49 prvků typu `int`

- 3** `#include <stdio.h>`  
`int main() {`  
    `float a = (int) 3.95;`  
    `int x = a;`  
    `printf("%f", x - a);`  
    `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** nelze přeložit
  - B** žádná z odpovědí není správná
  - C** 0.050000
  - D** 0.000000
  - E** 3.950000

- 4** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Vypíše řetězec "Hellorld"
  - B** Vypíše řetězec "Helrld"
  - C** Nelze přeložit
  - D** Vypíše řetězec "Hell"
  - E** Vypíše řetězec "Helord"
  - F** Vypíše řetězec "Hellrld"

- 5** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Vypíše hodnoty '1 1 6'
  - B** Vypíše hodnoty '3 3 6'
  - C** Nelze přeložit
  - D** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti `array`.
  - E** Žádná z odpovědí není správná
  - F** Vypíše hodnoty '3 3 7'

**6** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Hello Hello Hello Hello
- B** Hello World Hello World
- C** World Hello Hello Hello
- D** Nelze přeložit
- E** World Hello World Hello
- F** Žádná z odpovědí není správná

**7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- C** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na sémantické úrovni je typicky časově náročnější
- E** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Typová konverze na bitové úrovni je typicky časově náročnější

**8** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** pád programu
- B** žádná z odpovědí není správná
- C** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** memory leak o velikosti 60 bajtů
- E** memory leak o velikosti 120 bajtů
- F** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**10** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** vypíše World
- C** memory leak o velikosti 10 bajtů
- D** žádná z odpovědí není správná
- E** vypíše Hello

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		2

**1**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- B** způsobí zápis za konec přidělené paměti
- C** žádná z odpovědí není správná
- D** vytvoří pole o celkovém počtu 64 prvků typu int
- E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F** vytvoří pole o celkovém počtu 49 prvků typu int

**2**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše World
- B** žádná z odpovědí není správná
- C** memory leak o velikosti 10 bajtů
- D** nezpůsobí žádný memory leak
- E** vypíše Hello

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na bitové úrovni je typicky časově náročnější
- D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- E** Typová konverze na sémantické úrovni je typicky časově náročnější
- F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

**4**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 0.000000
- B** žádná z odpovědí není správná
- C** 3.950000
- D** nelze přeložit
- E** 0.050000

**5**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '3 3 7'
- B** Žádná z odpovědí není správná
- C** Nelze přeložit
- D** Vypíše hodnoty '1 1 6'
- E** Vypíše hodnoty '3 3 6'
- F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

**6**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Helrld"
- B** Vypíše řetězec "Hellorld"
- C** Nelze přeložit
- D** Vypíše řetězec "Hellrld"
- E** Vypíše řetězec "Helord"
- F** Vypíše řetězec "Hell"

**7**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Nelze přeložit
- B** Žádná z odpovědí není správná
- C** Hello World Hello World
- D** World Hello World Hello
- E** World Hello Hello Hello
- F** Hello Hello Hello Hello

**8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**9**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** memory leak o velikosti 120 bajtů
- C** memory leak o velikosti 60 bajtů
- D** pád programu
- E** žádná z odpovědí není správná
- F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

**10**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** foo(1);foo(4);foo(6);foo(10);foo(8);
- C** foo(5);foo(2);foo(5);foo(10);foo(7);
- D** foo(2);foo(3);foo(10);foo(7);
- E** foo(2);foo(0);foo(5);foo(7);foo(8);

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		3

- 1** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`  
Spuštění programu způsobí:
- A** memory leak o velikosti 60 bajtů
  - B** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)
  - C** pád programu
  - D** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)
  - E** žádná z odpovědí není správná
  - F** memory leak o velikosti 120 bajtů

- 2** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`  
Výše uvedený program vypíše:

- A** Hello Hello Hello Hello
- B** World Hello Hello Hello
- C** Hello World Hello World
- D** World Hello World Hello
- E** Nelze přeložit
- F** Žádná z odpovědí není správná

- 3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni je typicky časově náročnější
  - B** Typová konverze na sémantické úrovni je typicky časově náročnější
  - C** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - D** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

- 4** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** vytvoří pole o celkovém počtu 49 prvků typu `int`
  - B** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - C** žádná z odpovědí není správná
  - D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - E** vytvoří pole o celkovém počtu 64 prvků typu `int`
  - F** způsobí zápis za konec přidělené paměti

- 5** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Žádná z odpovědí není správná
  - B** Vypíše hodnoty '1 1 6'
  - C** Nelze přeložit
  - D** Vypíše hodnoty '3 3 7'
  - E** Vypíše hodnoty '3 3 6'
  - F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti `array`.

- 6** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Nelze přeložit
  - B** Vypíše řetězec "Helord"
  - C** Vypíše řetězec "Helrld"
  - D** Vypíše řetězec "Hllrld"
  - E** Vypíše řetězec "Hllorld"
  - F** Vypíše řetězec "Hell"

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

- 8** `#include <stdio.h>`  
`int main() {`  
    `float a = (int) 3.95;`  
    `int x = a;`  
    `printf("%f", x - a);`  
    `return 0;`  
`}`
- Výše uvedený program vypíše:
- A** žádná z odpovědí není správná
  - B** nelze přeložit
  - C** 0.000000
  - D** 0.050000
  - E** 3.950000

- 9** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
    `*array = malloc(10);`  
    `array = NULL;`  
`}`  
`int main() {`  
    `int* array = NULL;`  
    `foo(&array);`  
    `if (array != NULL) printf("Hello");`  
    `else printf("World");`  
    `free(array);`  
    `return 0;`  
`}`
- Výše uvedený program:
- A** memory leak o velikosti 10 bajtů
  - B** nezpůsobí žádný memory leak
  - C** žádná z odpovědí není správná
  - D** vypíše Hello
  - E** vypíše World

- 10** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`
- Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":
- A** `foo(1);foo(4);foo(6);foo(10);foo(8);`
  - B** `foo(5);foo(2);foo(5);foo(10);foo(7);`
  - C** `foo(2);foo(0);foo(5);foo(7);foo(8);`
  - D** `foo(2);foo(3);foo(10);foo(7);`
  - E** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		4

**1**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Nelze přeložit
- B World Hello World Hello
- C Žádná z odpovědí není správná
- D World Hello Hello Hello
- E Hello World Hello World
- F Hello Hello Hello Hello

**2**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A Vypíše řetězec "Hell"
- B Vypíše řetězec "Helrld"
- C Vypíše řetězec "Helord"
- D Vypíše řetězec "Hellrld"
- E Nelze přeložit
- F Vypíše řetězec "Helloworld"

**3**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A Vypíše hodnoty '3 3 6'
- B Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C Vypíše hodnoty '1 1 6'
- D Nelze přeložit
- E Žádná z odpovědí není správná
- F Vypíše hodnoty '3 3 7'

**4**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(1);foo(4);foo(6);foo(10);foo(8);
- B žádná z odpovědí není správná
- C foo(2);foo(3);foo(10);foo(7);
- D foo(5);foo(2);foo(5);foo(10);foo(7);
- E foo(2);foo(0);foo(5);foo(7);foo(8);

- 5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni je typicky časově náročnější
  - B** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - C** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - D** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Typová konverze na bitové úrovni je typicky časově náročnější
  - F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

- 6**
- ```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```
- Výše uvedený program:
- A** způsobí zápis za konec přidělené paměti
  - B** vytvoří pole o celkovém počtu 64 prvků typu int
  - C** žádná z odpovědí není správná
  - D** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - E** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - F** vytvoří pole o celkovém počtu 49 prvků typu int

- 7**
- ```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```
- Výše uvedený program vypíše:
- A** 0.050000
  - B** žádná z odpovědí není správná
  - C** 0.000000
  - D** nelze přeložit
  - E** 3.950000

- 8**
- ```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```
- Výše uvedený program:
- A** memory leak o velikosti 10 bajtů
  - B** nezpůsobí žádný memory leak
  - C** vypíše World
  - D** žádná z odpovědí není správná
  - E** vypíše Hello

- 9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

- 10**
- ```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```
- Spuštění programu způsobí:
- A** pád programu
  - B** memory leak o velikosti 120 bajtů
  - C** žádná z odpovědí není správná
  - D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - E** memory leak o velikosti 60 bajtů
  - F** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		5

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - C** Typová konverze na bitové úrovni je typicky časově náročnější
  - D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - E** Typová konverze na sémantické úrovni je typicky časově náročnější
  - F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

- 2**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:
- A** žádná z odpovědí není správná
  - B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - C** memory leak o velikosti 120 bajtů
  - D** pád programu
  - E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - F** memory leak o velikosti 60 bajtů

- 3**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(2);foo(3);foo(10);foo(7);
  - B** foo(1);foo(4);foo(6);foo(10);foo(8);
  - C** žádná z odpovědí není správná
  - D** foo(5);foo(2);foo(5);foo(10);foo(7);
  - E** foo(2);foo(0);foo(5);foo(7);foo(8);

- 4**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** vypíše World
- C** vypíše Hello
- D** nezpůsobí žádný memory leak
- E** žádná z odpovědí není správná

- 5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** způsobí zápis za konec přidělené paměti
  - B** vytvoří pole o celkovém počtu 49 prvků typu int
  - C** žádná z odpovědí není správná
  - D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - E** vytvoří pole o celkovém počtu 64 prvků typu int
  - F** vyplní všechny položky pole hodnotou z intervalu 1 do 7

- 6**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše řetězec "Hellrld"
  - B** Vypíše řetězec "Helord"
  - C** Vypíše řetězec "Hell"
  - D** Nelze přeložit
  - E** Vypíše řetězec "Hellrld"
  - F** Vypíše řetězec "Hellorld"

**7** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Nelze přeložit
- B** Žádná z odpovědí není správná
- C** Hello World Hello World
- D** World Hello Hello Hello
- E** Hello Hello Hello Hello
- F** World Hello World Hello

**8** `#include <stdio.h>`  
`int main() {`  
    `float a = (int) 3.95;`  
    `int x = a;`  
    `printf("%f", x - a);`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** 0.000000
- B** nelze přeložit
- C** 3.950000
- D** žádná z odpovědí není správná
- E** 0.050000

**9** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše hodnoty '3 3 6'
- B** Žádná z odpovědí není správná
- C** Nelze přeložit
- D** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- E** Vypíše hodnoty '1 1 6'
- F** Vypíše hodnoty '3 3 7'

**10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		6

**1**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A nelze přeložit
- B 0.050000
- C žádná z odpovědí není správná
- D 3.950000
- E 0.000000

**2**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(2);foo(3);foo(10);foo(7);
- B foo(5);foo(2);foo(5);foo(10);foo(7);
- C foo(1);foo(4);foo(6);foo(10);foo(8);
- D foo(2);foo(0);foo(5);foo(7);foo(8);
- E žádná z odpovědí není správná

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- C Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**4**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B Nelze přeložit
- C Žádná z odpovědí není správná
- D Vypíše hodnoty '3 3 6'
- E Vypíše hodnoty '1 1 6'
- F Vypíše hodnoty '3 3 7'

**5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A vytvoří pole o celkovém počtu 49 prvků typu int
- B způsobí zápis za konec přidělené paměti
- C vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D žádná z odpovědí není správná
- E vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F vytvoří pole o celkovém počtu 64 prvků typu int

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na bitové úrovni je typicky časově náročnější
- B Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- D Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F Typová konverze na sémantické úrovni je typicky časově náročnější

**7** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** pád programu
- C** memory leak o velikosti 60 bajtů
- D** memory leak o velikosti 120 bajtů
- E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** žádná z odpovědí není správná

**8** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** vypíše World
- C** nezpůsobí žádný memory leak
- D** vypíše Hello
- E** memory leak o velikosti 10 bajtů

**9** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Hellorld"
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Helrld"
- D** Vypíše řetězec "Hellrld"
- E** Vypíše řetězec "Hell"
- F** Nelze přeložit

**10** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** World Hello Hello Hello
- C** Hello World Hello World
- D** World Hello World Hello
- E** Nelze přeložit
- F** Hello Hello Hello Hello

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		7

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - B** Typová konverze na bitové úrovni je typicky časově náročnější
  - C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - D** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Typová konverze na sémantické úrovni je typicky časově náročnější

**2**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Nelze přeložit
- B** Hello World Hello World
- C** Hello Hello Hello Hello
- D** World Hello Hello Hello
- E** World Hello World Hello
- F** Žádná z odpovědí není správná

**3**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 3.950000
- B** 0.000000
- C** nelze přeložit
- D** 0.050000
- E** Žádná z odpovědí není správná

**4**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Žádná z odpovědí není správná
- B** Vypíše hodnoty '1 1 6'
- C** Nelze přeložit
- D** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- E** Vypíše hodnoty '3 3 6'
- F** Vypíše hodnoty '3 3 7'

**5**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** foo(1);foo(4);foo(6);foo(10);foo(8);
- C** foo(2);foo(0);foo(5);foo(7);foo(8);
- D** foo(2);foo(3);foo(10);foo(7);
- E** foo(5);foo(2);foo(5);foo(10);foo(7);

**6**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Helrld"
- D** Vypíše řetězec "Hellrld"
- E** Vypíše řetězec "Hell"
- F** Vypíše řetězec "Hellorld"

**7**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** vypíše Hello
- C** memory leak o velikosti 10 bajtů
- D** žádná z odpovědí není správná
- E** vypíše World

- 8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**9**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- B** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C** způsobí zápis za konec přidělené paměti
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** žádná z odpovědí není správná
- F** vytvoří pole o celkovém počtu 64 prvků typu int

**10**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 60 bajtů
- B** pád programu
- C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** žádná z odpovědí není správná
- F** memory leak o velikosti 120 bajtů

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		8

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - B** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

- 2**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:
- A** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - B** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - C** pád programu
  - D** memory leak o velikosti 120 bajtů
  - E** memory leak o velikosti 60 bajtů
  - F** žádná z odpovědí není správná

- 3**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše řetězec "Helord"
  - B** Vypíše řetězec "Hellrld"
  - C** Vypíše řetězec "Helrld"
  - D** Vypíše řetězec "Hellorld"
  - E** Nelze přeložit
  - F** Vypíše řetězec "Hell"

- 4**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** vytvoří pole o celkovém počtu 64 prvků typu int
  - B** žádná z odpovědí není správná
  - C** způsobí zápis za konec přidělené paměti
  - D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - F** vytvoří pole o celkovém počtu 49 prvků typu int

- 5**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:
- A** nezpůsobí žádný memory leak
  - B** žádná z odpovědí není správná
  - C** memory leak o velikosti 10 bajtů
  - D** vypíše Hello
  - E** vypíše World

- 6**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:
- A** Hello World Hello World
  - B** Nelze přeložit
  - C** Hello Hello Hello Hello
  - D** World Hello World Hello
  - E** Žádná z odpovědí není správná
  - F** World Hello Hello Hello

**7** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše hodnoty '1 1 6'
- B** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C** Žádná z odpovědí není správná
- D** Vypíše hodnoty '3 3 6'
- E** Nelze přeložit
- F** Vypíše hodnoty '3 3 7'

**8** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`  
 Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** foo(5);foo(2);foo(5);foo(10);foo(7);
- C** foo(2);foo(0);foo(5);foo(7);foo(8);
- D** foo(2);foo(3);foo(10);foo(7);
- E** foo(1);foo(4);foo(6);foo(10);foo(8);

**9** `#include <stdio.h>`  
`int main() {`  
 `float a = (int) 3.95;`  
 `int x = a;`  
 `printf("%f", x - a);`  
 `return 0;`  
`}`  
 Výše uvedený program vypíše:

- A** nelze přeložit
- B** 3.950000
- C** 0.050000
- D** žádná z odpovědí není správná
- E** 0.000000

**10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na bitové úrovni je typicky časově náročnější
- E** Typová konverze na sémantické úrovni je typicky časově náročnější
- F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		9

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

- 2**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:
- A** žádná z odpovědí není správná
  - B** 0.000000
  - C** 3.950000
  - D** nelze přeložit
  - E** 0.050000

- 3**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:
- A** memory leak o velikosti 10 bajtů
  - B** žádná z odpovědí není správná
  - C** vypíše Hello
  - D** vypíše World
  - E** nezpůsobí žádný memory leak

- 4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni je typicky časově náročnější
  - C** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - F** Typová konverze na bitové úrovni je typicky časově náročnější

- 5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - B** vytvoří pole o celkovém počtu 49 prvků typu int
  - C** způsobí zápis za konec přidělené paměti
  - D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - E** žádná z odpovědí není správná
  - F** vytvoří pole o celkovém počtu 64 prvků typu int

- 6**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(2);foo(0);foo(5);foo(7);foo(8);
  - B** foo(1);foo(4);foo(6);foo(10);foo(8);
  - C** foo(2);foo(3);foo(10);foo(7);
  - D** žádná z odpovědí není správná
  - E** foo(5);foo(2);foo(5);foo(10);foo(7);

- 7**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše řetězec "Hellrld"
  - B** Nelze přeložit
  - C** Vypíše řetězec "Hellorld"
  - D** Vypíše řetězec "Helord"
  - E** Vypíše řetězec "Helrld"
  - F** Vypíše řetězec "Hell"

**8** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** pád programu
- C** memory leak o velikosti 60 bajtů
- D** žádná z odpovědí není správná
- E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** memory leak o velikosti 120 bajtů

**10** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** World Hello World Hello
- B** Hello World Hello World
- C** Hello Hello Hello Hello
- D** World Hello Hello Hello
- E** Žádná z odpovědí není správná
- F** Nelze přeložit

**9** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše hodnoty '1 1 6'
- B** Vypíše hodnoty '3 3 6'
- C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D** Nelze přeložit
- E** Vypíše hodnoty '3 3 7'
- F** Žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		10

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - C** Typová konverze na bitové úrovni je typicky časově náročnější
  - D** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - F** Typová konverze na sémantické úrovni je typicky časově náročnější

**2**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hellorld"
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Hellrld"
- D** Nelze přeložit
- E** Vypíše řetězec "Hell"
- F** Vypíše řetězec "Helrld"

**3**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 3.950000
- B** žádná z odpovědí není správná
- C** 0.000000
- D** nelze přeložit
- E** 0.050000

- 4**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```
- Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(1);foo(4);foo(6);foo(10);foo(8);
  - B** foo(2);foo(3);foo(10);foo(7);
  - C** foo(2);foo(0);foo(5);foo(7);foo(8);
  - D** foo(5);foo(2);foo(5);foo(10);foo(7);
  - E** žádná z odpovědí není správná

- 5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**6**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** žádná z odpovědí není správná
- C** vypíše Hello
- D** nezpůsobí žádný memory leak
- E** vypíše World

- 7** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`  
Výše uvedený program:
- A** Žádná z odpovědí není správná
  - B** Vypíše hodnoty '3 3 7'
  - C** Nelze přeložit
  - D** Vypíše hodnoty '3 3 6'
  - E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
  - F** Vypíše hodnoty '1 1 6'

- 8** `int main() {`  
 `int array[7][7];`  
 `for (int i = 1; i <= 7; i++) {`  
 `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
 `}`  
 `return 0;`  
`}`  
Výše uvedený program:
- A** vytvoří pole o celkovém počtu 49 prvků typu int
  - B** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - C** vytvoří pole o celkovém počtu 64 prvků typu int
  - D** žádná z odpovědí není správná
  - E** způsobí zápis za konec přidělené paměti
  - F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

- 9** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** Hello World Hello World
  - B** Nelze přeložit
  - C** Hello Hello Hello Hello
  - D** World Hello World Hello
  - E** World Hello Hello Hello
  - F** Žádná z odpovědí není správná

- 10** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`  
Spuštění programu způsobí:
- A** memory leak o velikosti 60 bajtů
  - B** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - D** pád programu
  - E** memory leak o velikosti 120 bajtů
  - F** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		11

**1**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A vytvoří pole o celkovém počtu 64 prvků typu int
- B vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C vytvoří pole o celkovém počtu 49 prvků typu int
- D způsobí zápis za konec přidělené paměti
- E žádná z odpovědí není správná
- F vyplní všechny položky pole hodnotou z intervalu 1 do 7

**2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na bitové úrovni je typicky časově náročnější
- B Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- D Typová konverze na sémantické úrovni je typicky časově náročnější
- E Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

**3**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A Vypíše řetězec "Hell"
- B Vypíše řetězec "Hellorld"
- C Vypíše řetězec "Helrld"
- D Nelze přeložit
- E Vypíše řetězec "Helord"
- F Vypíše řetězec "Hellrld"

**4**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A 0.000000
- B 0.050000
- C žádná z odpovědí není správná
- D nelze přeložit
- E 3.950000

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**6**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C žádná z odpovědí není správná
- D memory leak o velikosti 60 bajtů
- E memory leak o velikosti 120 bajtů
- F pád programu

- 7** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`  
 Výše uvedený program:
- A** Vypíše hodnoty '3 3 7'  
**B** Vypíše hodnoty '3 3 6'  
**C** Nelze přeložit  
**D** Žádná z odpovědí není správná  
**E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**F** Vypíše hodnoty '1 1 6'

- 8** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`  
 Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(2);foo(0);foo(5);foo(7);foo(8);  
**B** foo(1);foo(4);foo(6);foo(10);foo(8);  
**C** foo(2);foo(3);foo(10);foo(7);  
**D** Žádná z odpovědí není správná  
**E** foo(5);foo(2);foo(5);foo(10);foo(7);

- 9** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`  
 Výše uvedený program vypíše:
- A** Hello World Hello World  
**B** Hello Hello Hello Hello  
**C** Žádná z odpovědí není správná  
**D** Nelze přeložit  
**E** World Hello World Hello  
**F** World Hello Hello Hello

- 10** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`  
 Výše uvedený program:
- A** vypíše World  
**B** memory leak o velikosti 10 bajtů  
**C** vypíše Hello  
**D** nezpůsobí žádný memory leak  
**E** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		12

**1**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** nelze přeložit
- B** žádná z odpovědí není správná
- C** 0.000000
- D** 3.950000
- E** 0.050000

**2**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(2);foo(3);foo(10);foo(7);
- B** foo(5);foo(2);foo(5);foo(10);foo(7);
- C** žádná z odpovědí není správná
- D** foo(2);foo(0);foo(5);foo(7);foo(8);
- E** foo(1);foo(4);foo(6);foo(10);foo(8);

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- B** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B** Typová konverze na bitové úrovni je typicky časově náročnější
- C** Typová konverze na sémantické úrovni je typicky časově náročnější
- D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

**5**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** pád programu
- B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** žádná z odpovědí není správná
- D** memory leak o velikosti 120 bajtů
- E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** memory leak o velikosti 60 bajtů

**6**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** Hello World Hello World
- C** Hello Hello Hello Hello
- D** World Hello World Hello
- E** World Hello Hello Hello
- F** Nelze přeložit

**7**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** způsobí zápis za konec přidělené paměti
- B** vytvoří pole o celkovém počtu 64 prvků typu int
- C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D** žádná z odpovědí není správná
- E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F** vytvoří pole o celkovém počtu 49 prvků typu int

**8**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Helord"
- B** Nelze přeložit
- C** Vypíše řetězec "Hell"
- D** Vypíše řetězec "Helrld"
- E** Vypíše řetězec "Hellorld"
- F** Vypíše řetězec "Hellrld"

**9**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '3 3 6'
- B** Nelze přeložit
- C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D** Vypíše hodnoty '3 3 7'
- E** Vypíše hodnoty '1 1 6'
- F** Žádná z odpovědí není správná

**10**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** vypíše World
- C** nezpůsobí žádný memory leak
- D** vypíše Hello
- E** žádná z odpovědí není správná



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		13

- 1**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

 Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(5);foo(2);foo(5);foo(10);foo(7);
  - B** žádná z odpovědí není správná
  - C** foo(1);foo(4);foo(6);foo(10);foo(8);
  - D** foo(2);foo(3);foo(10);foo(7);
  - E** foo(2);foo(0);foo(5);foo(7);foo(8);

- 2**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

 Výše uvedený program:
- A** Vypíše řetězec "Hellorld"
  - B** Vypíše řetězec "Hell"
  - C** Vypíše řetězec "Helord"
  - D** Vypíše řetězec "Helrld"
  - E** Vypíše řetězec "Hellrld"
  - F** Nelze přeložit

- 3**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

 Výše uvedený program:
- A** vypíše World
  - B** žádná z odpovědí není správná
  - C** vypíše Hello
  - D** nezpůsobí žádný memory leak
  - E** memory leak o velikosti 10 bajtů

- 4**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

 Výše uvedený program:
- A** Vypíše hodnoty '3 3 7'
  - B** Vypíše hodnoty '1 1 6'
  - C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
  - D** Vypíše hodnoty '3 3 6'
  - E** Žádná z odpovědí není správná
  - F** Nelze přeložit

**5**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** World Hello World Hello
- C** Hello Hello Hello Hello
- D** Hello World Hello World
- E** World Hello Hello Hello
- F** Nelze přeložit

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E** Typová konverze na sémantické úrovni je typicky časově náročnější
- F** Typová konverze na bitové úrovni je typicky časově náročnější

**7**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 60 bajtů
- B** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** memory leak o velikosti 120 bajtů
- D** pád programu
- E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** žádná z odpovědí není správná

**8**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 0.050000
- B** 3.950000
- C** žádná z odpovědí není správná
- D** nelze přeložit
- E** 0.000000

**9**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 49 prvků typu int
- B** žádná z odpovědí není správná
- C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D** vytvoří pole o celkovém počtu 64 prvků typu int
- E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F** způsobí zápis za konec přidělené paměti

**10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		14

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - C** Typová konverze na sémantické úrovni je typicky časově náročnější
  - D** Typová konverze na bitové úrovni je typicky časově náročnější
  - E** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

- 2**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:
- A** žádná z odpovědí není správná
  - B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - C** pád programu
  - D** memory leak o velikosti 120 bajtů
  - E** memory leak o velikosti 60 bajtů
  - F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

- 3**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:
- A** nezpůsobí žádný memory leak
  - B** memory leak o velikosti 10 bajtů
  - C** vypíše World
  - D** vypíše Hello
  - E** žádná z odpovědí není správná

- 4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - B** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

- 5**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(2);foo(0);foo(5);foo(7);foo(8);
  - B** foo(1);foo(4);foo(6);foo(10);foo(8);
  - C** žádná z odpovědí není správná
  - D** foo(5);foo(2);foo(5);foo(10);foo(7);
  - E** foo(2);foo(3);foo(10);foo(7);

- 6**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše řetězec "Hellrld"
  - B** Vypíše řetězec "Hell"
  - C** Vypíše řetězec "Helord"
  - D** Nelze přeložit
  - E** Vypíše řetězec "Hellorld"
  - F** Vypíše řetězec "Helrld"

```

7 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}

```

Výše uvedený program:

- A** Žádná z odpovědí není správná
- B** Vypíše hodnoty '3 3 6'
- C** Nelze přeložit
- D** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- E** Vypíše hodnoty '3 3 7'
- F** Vypíše hodnoty '1 1 6'

```

8 int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}

```

Výše uvedený program:

- A** způsobí zápis za konec přidělené paměti
- B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C** žádná z odpovědí není správná
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** vytvoří pole o celkovém počtu 64 prvků typu int
- F** vyplní všechny položky pole hodnotou z intervalu 1 do 7

```

9 #include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}

```

Výše uvedený program vypíše:

- A** 0.050000
- B** nelze přeložit
- C** žádná z odpovědí není správná
- D** 3.950000
- E** 0.000000

```

10 #include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}

```

Výše uvedený program vypíše:

- A** World Hello Hello Hello
- B** World Hello World Hello
- C** Hello World Hello World
- D** Nelze přeložit
- E** Hello Hello Hello Hello
- F** Žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		15

- 1** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`  
Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":
- A** žádná z odpovědí není správná
  - B** `foo(5);foo(2);foo(5);foo(10);foo(7);`
  - C** `foo(1);foo(4);foo(6);foo(10);foo(8);`
  - D** `foo(2);foo(0);foo(5);foo(7);foo(8);`
  - E** `foo(2);foo(3);foo(10);foo(7);`

- 2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni je typicky časově náročnější
  - B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - D** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Typová konverze na sémantické úrovni je typicky časově náročnější

- 3** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Vypíše řetězec "Hellorld"
  - B** Vypíše řetězec "Helrld"
  - C** Vypíše řetězec "Hellrld"
  - D** Vypíše řetězec "Helord"
  - E** Nelze přeložit
  - F** Vypíše řetězec "Hell"

- 4** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`  
Spuštění programu způsobí:
- A** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)
  - B** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)
  - C** žádná z odpovědí není správná
  - D** memory leak o velikosti 60 bajtů
  - E** memory leak o velikosti 120 bajtů
  - F** pád programu

- 5** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - B** žádná z odpovědí není správná
  - C** vytvoří pole o celkovém počtu 64 prvků typu `int`
  - D** vytvoří pole o celkovém počtu 49 prvků typu `int`
  - E** způsobí zápis za konec přidělené paměti
  - F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

- 6** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`  
Výše uvedený program:
- A** Žádná z odpovědí není správná  
**B** Vypíše hodnoty '3 3 6'  
**C** Nelze přeložit  
**D** Vypíše hodnoty '3 3 7'  
**E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**F** Vypíše hodnoty '1 1 6'

- 7** `#include <stdio.h>`  
`int main() {`  
 `float a = (int) 3.95;`  
 `int x = a;`  
 `printf("%f", x - a);`  
 `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** nelze přeložit  
**B** 0.000000  
**C** 3.950000  
**D** žádná z odpovědí není správná  
**E** 0.050000

- 8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel  
**B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel  
**C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind  
**D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

- 9** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`  
Výše uvedený program:
- A** vypíše Hello  
**B** nezpůsobí žádný memory leak  
**C** vypíše World  
**D** žádná z odpovědí není správná  
**E** memory leak o velikosti 10 bajtů

- 10** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** World Hello World Hello  
**B** Hello Hello Hello Hello  
**C** World Hello Hello Hello  
**D** Hello World Hello World  
**E** Nelze přeložit  
**F** Žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		16

**1**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A Vypíše hodnoty '3 3 6'
- B Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C Nelze přeložit
- D Vypíše hodnoty '3 3 7'
- E Vypíše hodnoty '1 1 6'
- F Žádná z odpovědí není správná

**2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- B Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**3**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A vypíše Hello
- B vypíše World
- C memory leak o velikosti 10 bajtů
- D nezpůsobí žádný memory leak
- E žádná z odpovědí není správná

**4**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B žádná z odpovědí není správná
- C způsobí zápis za konec přidělené paměti
- D vytvoří pole o celkovém počtu 64 prvků typu int
- E vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F vytvoří pole o celkovém počtu 49 prvků typu int

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na bitové úrovni je typicky časově náročnější
- B Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- C Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- D Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- E Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F Typová konverze na sémantické úrovni je typicky časově náročnější

**6**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A World Hello World Hello
- B Hello Hello Hello Hello
- C Nelze přeložit
- D World Hello Hello Hello
- E Hello World Hello World
- F Žádná z odpovědí není správná

```

7 #include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}

```

Výše uvedený program:

- A Vypíše řetězec "Helrld"
- B Vypíše řetězec "Hellorld"
- C Nelze přeložit
- D Vypíše řetězec "Hellrld"
- E Vypíše řetězec "Helord"
- F Vypíše řetězec "Hell"

```

8 #include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}

```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(1);foo(4);foo(6);foo(10);foo(8);
- B foo(5);foo(2);foo(5);foo(10);foo(7);
- C foo(2);foo(3);foo(10);foo(7);
- D žádná z odpovědí není správná
- E foo(2);foo(0);foo(5);foo(7);foo(8);

```

9 #include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}

```

Výše uvedený program vypíše:

- A žádná z odpovědí není správná
- B 3.950000
- C 0.000000
- D 0.050000
- E nelze přeložit

```

10 #include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}

```

Spuštění programu způsobí:

- A memory leak o velikosti 120 bajtů
- B žádná z odpovědí není správná
- C memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D memory leak o velikosti 60 bajtů
- E pád programu
- F memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		17

**1**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A 3.950000
- B 0.000000
- C nelze přeložit
- D žádná z odpovědí není správná
- E 0.050000

**2**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A Vypíše hodnoty '1 1 6'
- B Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C Nelze přeložit
- D Vypíše hodnoty '3 3 6'
- E Vypíše hodnoty '3 3 7'
- F Žádná z odpovědí není správná

**3**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A vypíše Hello
- B memory leak o velikosti 10 bajtů
- C nezpůsobí žádný memory leak
- D žádná z odpovědí není správná
- E vypíše World

**4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- C Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

**5**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A memory leak o velikosti 60 bajtů
- B memory leak o velikosti 120 bajtů
- C memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D žádná z odpovědí není správná
- E pád programu
- F memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na bitové úrovni je typicky časově náročnější
- B Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- C Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- E Typová konverze na sémantické úrovni je typicky časově náročnější
- F Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace

**7** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Helrld"
- B** Vypíše řetězec "Hellrld"
- C** Vypíše řetězec "Helord"
- D** Nelze přeložit
- E** Vypíše řetězec "Hellorld"
- F** Vypíše řetězec "Hell"

**8** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Hello World Hello World
- B** World Hello World Hello
- C** Žádná z odpovědí není správná
- D** Hello Hello Hello Hello
- E** Nelze přeložit
- F** World Hello Hello Hello

**9** `int main() {`  
 `int array[7][7];`  
 `for (int i = 1; i <= 7; i++) {`  
 `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
 `}`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Žádná z odpovědí není správná
- B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C** vytvoří pole o celkovém počtu 64 prvků typu int
- D** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E** způsobí zápis za konec přidělené paměti
- F** vytvoří pole o celkovém počtu 49 prvků typu int

**10** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** foo(2);foo(0);foo(5);foo(7);foo(8);
- C** foo(5);foo(2);foo(5);foo(10);foo(7);
- D** foo(1);foo(4);foo(6);foo(10);foo(8);
- E** foo(2);foo(3);foo(10);foo(7);

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		18

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**2**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B** Žádná z odpovědí není správná
- C** Nelze přeložit
- D** Vypíše hodnoty '1 1 6'
- E** Vypíše hodnoty '3 3 7'
- F** Vypíše hodnoty '3 3 6'

**3**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše řetězec "Hellrld"
- C** Vypíše řetězec "Hellorld"
- D** Vypíše řetězec "Hell"
- E** Vypíše řetězec "Helord"
- F** Vypíše řetězec "Helrld"

**4**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
Spuštění programu způsobí:
```

- A** memory leak o velikosti 60 bajtů
- B** pád programu
- C** memory leak o velikosti 120 bajtů
- D** Žádná z odpovědí není správná
- E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)

**5**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Hello Hello Hello Hello
- B** Nelze přeložit
- C** Hello World Hello World
- D** World Hello World Hello
- E** World Hello Hello Hello
- F** Žádná z odpovědí není správná

- 6**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

 Která sekvence volání funkce foo() vypíše řetězec "Freedom":
  - A** foo(2);foo(0);foo(5);foo(7);foo(8);
  - B** foo(5);foo(2);foo(5);foo(10);foo(7);
  - C** foo(1);foo(4);foo(6);foo(10);foo(8);
  - D** žádná z odpovědí není správná
  - E** foo(2);foo(3);foo(10);foo(7);

- 7**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

 Výše uvedený program:
  - A** žádná z odpovědí není správná
  - B** vytvoří pole o celkovém počtu 64 prvků typu int
  - C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - D** způsobí zápis za konec přidělené paměti
  - E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - F** vytvoří pole o celkovém počtu 49 prvků typu int

- 8**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

 Výše uvedený program vypíše:
  - A** 0.050000
  - B** 3.950000
  - C** nelze přeložit
  - D** 0.000000
  - E** žádná z odpovědí není správná

- 9**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

 Výše uvedený program:
  - A** memory leak o velikosti 10 bajtů
  - B** žádná z odpovědí není správná
  - C** nezpůsobí žádný memory leak
  - D** vypíše World
  - E** vypíše Hello

- 10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
  - A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - C** Typová konverze na sémantické úrovni je typicky časově náročnější
  - D** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - F** Typová konverze na bitové úrovni je typicky časově náročnější

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		19

**1** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Hello World Hello World
- B** World Hello World Hello
- C** Žádná z odpovědí není správná
- D** Hello Hello Hello Hello
- E** World Hello Hello Hello
- F** Nelze přeložit

**2** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** memory leak o velikosti 120 bajtů
- C** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** pád programu
- E** žádná z odpovědí není správná
- F** memory leak o velikosti 60 bajtů

**3** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Helrld"
- B** Vypíše řetězec "Hellrld"
- C** Nelze přeložit
- D** Vypíše řetězec "Helord"
- E** Vypíše řetězec "Hell"
- F** Vypíše řetězec "Hellorld"

**4** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(1);foo(4);foo(6);foo(10);foo(8);
- B** foo(2);foo(0);foo(5);foo(7);foo(8);
- C** žádná z odpovědí není správná
- D** foo(5);foo(2);foo(5);foo(10);foo(7);
- E** foo(2);foo(3);foo(10);foo(7);

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

- 6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - B** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - C** Typová konverze na bitové úrovni je typicky časově náročnější
  - D** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Typová konverze na sémantické úrovni je typicky časově náročnější
  - F** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace

**7**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B** způsobí zápis za konec přidělené paměti
- C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D** vytvoří pole o celkovém počtu 64 prvků typu int
- E** vytvoří pole o celkovém počtu 49 prvků typu int
- F** žádná z odpovědí není správná

**8**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '1 1 6'
- B** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C** Vypíše hodnoty '3 3 7'
- D** Žádná z odpovědí není správná
- E** Vypíše hodnoty '3 3 6'
- F** Nelze přeložit

**9**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše World
- B** nezpůsobí žádný memory leak
- C** vypíše Hello
- D** žádná z odpovědí není správná
- E** memory leak o velikosti 10 bajtů

**10**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** žádná z odpovědí není správná
- B** 0.000000
- C** nelze přeložit
- D** 0.050000
- E** 3.950000

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		20

**1**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A Nelze přeložit
- B Vypíše řetězec "Hell"
- C Vypíše řetězec "Helrld"
- D Vypíše řetězec "Hellorld"
- E Vypíše řetězec "Helord"
- F Vypíše řetězec "Hellrld"

**2**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A nezpůsobí žádný memory leak
- B vypíše Hello
- C vypíše World
- D memory leak o velikosti 10 bajtů
- E žádná z odpovědí není správná

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- B Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- D Typová konverze na sémantické úrovni je typicky časově náročnější
- E Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F Typová konverze na bitové úrovni je typicky časově náročnější

**4**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(2);foo(3);foo(10);foo(7);
- B foo(5);foo(2);foo(5);foo(10);foo(7);
- C žádná z odpovědí není správná
- D foo(2);foo(0);foo(5);foo(7);foo(8);
- E foo(1);foo(4);foo(6);foo(10);foo(8);

**5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A žádná z odpovědí není správná
- B vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C vytvoří pole o celkovém počtu 49 prvků typu int
- D vytvoří pole o celkovém počtu 64 prvků typu int
- E vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F způsobí zápis za konec přidělené paměti

**6**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A 0.050000
- B 3.950000
- C žádná z odpovědí není správná
- D 0.000000
- E nelze přeložit

**7** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** World Hello World Hello
- C** Nelze přeložit
- D** World Hello Hello Hello
- E** Hello World Hello World
- F** Hello Hello Hello Hello

**10** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 120 bajtů
- B** žádná z odpovědí není správná
- C** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** pád programu
- E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** memory leak o velikosti 60 bajtů

**8** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše hodnoty '3 3 7'
- C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D** Vypíše hodnoty '1 1 6'
- E** Žádná z odpovědí není správná
- F** Vypíše hodnoty '3 3 6'

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- B** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		21

**1**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A Vypíše řetězec "Hellrld"
- B Vypíše řetězec "Hell"
- C Vypíše řetězec "Hello world"
- D Vypíše řetězec "Helord"
- E Nelze přeložit
- F Vypíše řetězec "Helrld"

**2**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A vypíše World
- B nezpůsobí žádný memory leak
- C memory leak o velikosti 10 bajtů
- D vypíše Hello
- E žádná z odpovědí není správná

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- C Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- D Typová konverze na sémantické úrovni je typicky časově náročnější
- E Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F Typová konverze na bitové úrovni je typicky časově náročnější

**4**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B Žádná z odpovědí není správná
- C Vypíše hodnoty '3 3 6'
- D Vypíše hodnoty '1 1 6'
- E Vypíše hodnoty '3 3 7'
- F Nelze přeložit

**5**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Nelze přeložit
- B Žádná z odpovědí není správná
- C World Hello World Hello
- D World Hello Hello Hello
- E Hello Hello Hello Hello
- F Hello World Hello World

**6**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A 3.950000
- B nelze přeložit
- C 0.000000
- D 0.050000
- E žádná z odpovědí není správná

```

7 #include <stdlib.h>
  int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
  }

```

Spuštění programu způsobí:

- A pád programu
- B memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- D memory leak o velikosti 60 bajtů
- E žádná z odpovědí není správná
- F memory leak o velikosti 120 bajtů

8 Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D Memory leak lze detekovat s využitím nástrojů, např. Valgrind

```

9 #include <stdio.h>
  void foo(int a) {
    switch (a) {
      case 0: break;
      case 1: printf("Fr"); break;
      case 2: printf("F");
      case 3: printf("re"); break;
      case 4: printf("e");
      case 5:
      case 6: break;
      case 7:
      case 8: printf("m"); break;
      default: printf("edo");
    }
  }

```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(5);foo(2);foo(5);foo(10);foo(7);
- B foo(2);foo(0);foo(5);foo(7);foo(8);
- C žádná z odpovědí není správná
- D foo(1);foo(4);foo(6);foo(10);foo(8);
- E foo(2);foo(3);foo(10);foo(7);

```

10 int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}

```

Výše uvedený program:

- A žádná z odpovědí není správná
- B vytvoří pole o celkovém počtu 64 prvků typu int
- C vytvoří pole o celkovém počtu 49 prvků typu int
- D vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- E vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F způsobí zápis za konec přidělené paměti

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		22

**1**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

 Výše uvedený program vypíše:

- A nelze přeložit
- B 3.950000
- C 0.050000
- D žádná z odpovědí není správná
- E 0.000000

**2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na bitové úrovni je typicky časově náročnější
- B Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- C Typová konverze na sémantické úrovni je typicky časově náročnější
- D Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- E Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace

**3**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

 Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(2);foo(0);foo(5);foo(7);foo(8);
- B žádná z odpovědí není správná
- C foo(5);foo(2);foo(5);foo(10);foo(7);
- D foo(2);foo(3);foo(10);foo(7);
- E foo(1);foo(4);foo(6);foo(10);foo(8);

**4**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

 Výše uvedený program vypíše:

- A World Hello Hello Hello
- B Hello Hello Hello Hello
- C Hello World Hello World
- D Nelze přeložit
- E World Hello World Hello
- F Žádná z odpovědí není správná

**5**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

 Spuštění programu způsobí:

- A žádná z odpovědí není správná
- B memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- D memory leak o velikosti 120 bajtů
- E memory leak o velikosti 60 bajtů
- F pád programu

- 6** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`  
 Výše uvedený program:
- A** Vypíše hodnoty '3 3 6'
  - B** Vypíše hodnoty '1 1 6'
  - C** Vypíše hodnoty '3 3 7'
  - D** Žádná z odpovědí není správná
  - E** Nelze přeložit
  - F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

- 7** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`  
 Výše uvedený program:
- A** žádná z odpovědí není správná
  - B** memory leak o velikosti 10 bajtů
  - C** vypíše Hello
  - D** nezpůsobí žádný memory leak
  - E** vypíše World

- 8** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`  
 Výše uvedený program:
- A** Vypíše řetězec "Helrld"
  - B** Vypíše řetězec "Hell"
  - C** Vypíše řetězec "Helord"
  - D** Vypíše řetězec "Hellrld"
  - E** Nelze přeložit
  - F** Vypíše řetězec "Hellorld"

- 9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

- 10** `int main() {`  
 `int array[7][7];`  
 `for (int i = 1; i <= 7; i++) {`  
 `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
 `}`  
 `return 0;`  
`}`  
 Výše uvedený program:
- A** způsobí zápis za konec přidělené paměti
  - B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - C** vytvoří pole o celkovém počtu 49 prvků typu int
  - D** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - E** vytvoří pole o celkovém počtu 64 prvků typu int
  - F** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		23

**1** #include <stdio.h>

```
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A žádná z odpovědí není správná
- B nelze přeložit
- C 0.050000
- D 3.950000
- E 0.000000

**2** #include <stdio.h>

```
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(5);foo(2);foo(5);foo(10);foo(7);
- B žádná z odpovědí není správná
- C foo(1);foo(4);foo(6);foo(10);foo(8);
- D foo(2);foo(0);foo(5);foo(7);foo(8);
- E foo(2);foo(3);foo(10);foo(7);

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B Typová konverze na sémantické úrovni je typicky časově náročnější
- C Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E Typová konverze na bitové úrovni je typicky časově náročnější
- F Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace

**4** #include <stdlib.h>

```
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A memory leak o velikosti 60 bajtů
- B memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C pád programu
- D memory leak o velikosti 120 bajtů
- E žádná z odpovědí není správná
- F memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

**5** int main() {

```
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- B vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C vytvoří pole o celkovém počtu 64 prvků typu int
- D žádná z odpovědí není správná
- E vytvoří pole o celkovém počtu 49 prvků typu int
- F způsobí zápis za konec přidělené paměti

**6** #include <stdio.h>

```
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Nelze přeložit
- B World Hello World Hello
- C Hello World Hello World
- D Žádná z odpovědí není správná
- E World Hello Hello Hello
- F Hello Hello Hello Hello

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**8**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hellorld"
- B** Vypíše řetězec "Helrld"
- C** Vypíše řetězec "Helord"
- D** Vypíše řetězec "Hell"
- E** Vypíše řetězec "Hellrld"
- F** Nelze přeložit

**9**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše hodnoty '3 3 7'
- C** Vypíše hodnoty '3 3 6'
- D** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- E** Žádná z odpovědí není správná
- F** Vypíše hodnoty '1 1 6'

**10**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** žádná z odpovědí není správná
- C** vypíše Hello
- D** memory leak o velikosti 10 bajtů
- E** vypíše World

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		24

- 1**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** žádná z odpovědí není správná  
**B** foo(2);foo(3);foo(10);foo(7);  
**C** foo(2);foo(0);foo(5);foo(7);foo(8);  
**D** foo(1);foo(4);foo(6);foo(10);foo(8);  
**E** foo(5);foo(2);foo(5);foo(10);foo(7);

- 2**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** žádná z odpovědí není správná  
**B** vytvoří pole o celkovém počtu 64 prvků typu int  
**C** vyplní všechny položky pole hodnotou z intervalu 1 do 7  
**D** vytvoří pole o celkovém počtu 49 prvků typu int  
**E** způsobí zápis za konec přidělené paměti  
**F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

- 3**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše hodnoty '1 1 6'  
**B** Nelze přeložit  
**C** Žádná z odpovědí není správná  
**D** Vypíše hodnoty '3 3 7'  
**E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**F** Vypíše hodnoty '3 3 6'

- 4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind  
**B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel  
**C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel  
**D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

- 5**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:
- A** memory leak o velikosti 120 bajtů  
**B** pád programu  
**C** žádná z odpovědí není správná  
**D** memory leak o velikosti 60 bajtů  
**E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)  
**F** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)

**6** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Hellrld"
- D** Nelze přeložit
- E** Vypíše řetězec "Helloworld"
- F** Vypíše řetězec "Helrld"

**7** `#include <stdio.h>`  
`int main() {`  
 `float a = (int) 3.95;`  
 `int x = a;`  
 `printf("%f", x - a);`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** nelze přeložit
- B** žádná z odpovědí není správná
- C** 0.000000
- D** 0.050000
- E** 3.950000

**8** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Hello World Hello World
- B** Nelze přeložit
- C** World Hello Hello Hello
- D** Hello Hello Hello Hello
- E** Žádná z odpovědí není správná
- F** World Hello World Hello

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- B** Typová konverze na sémantické úrovni je typicky časově náročnější
- C** Typová konverze na bitové úrovni je typicky časově náročnější
- D** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

**10** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** vypíše Hello
- C** vypíše World
- D** memory leak o velikosti 10 bajtů
- E** nezpůsobí žádný memory leak



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		25

- 1** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**B** Vypíše hodnoty '1 1 6'  
**C** Vypíše hodnoty '3 3 7'  
**D** Žádná z odpovědí není správná  
**E** Vypíše hodnoty '3 3 6'  
**F** Nelze přeložit

- 2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**C** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ  
**D** Typová konverze na sémantické úrovni je typicky časově náročnější  
**E** Typová konverze na bitové úrovni je typicky časově náročnější  
**F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

- 3** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Vypíše řetězec "Hellrld"  
**B** Vypíše řetězec "Hell"  
**C** Nelze přeložit  
**D** Vypíše řetězec "Helord"  
**E** Vypíše řetězec "Hellorld"  
**F** Vypíše řetězec "HeLrld"

- 4** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** Žádná z odpovědí není správná  
**B** World Hello Hello Hello  
**C** Nelze přeložit  
**D** Hello Hello Hello Hello  
**E** World Hello World Hello  
**F** Hello World Hello World

- 5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel  
**B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel  
**C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind  
**D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

- 6** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`  
Spuštění programu způsobí:
- A** memory leak o velikosti 60 bajtů  
**B** žádná z odpovědí není správná  
**C** pád programu  
**D** memory leak o velikosti 120 bajtů  
**E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)  
**F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

```
7 #include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A vypíše Hello
- B memory leak o velikosti 10 bajtů
- C nezpůsobí žádný memory leak
- D vypíše World
- E žádná z odpovědí není správná

```
8 #include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(5);foo(2);foo(5);foo(10);foo(7);
- B foo(1);foo(4);foo(6);foo(10);foo(8);
- C žádná z odpovědí není správná
- D foo(2);foo(0);foo(5);foo(7);foo(8);
- E foo(2);foo(3);foo(10);foo(7);

```
9 #include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A nelze přeložit
- B žádná z odpovědí není správná
- C 0.000000
- D 3.950000
- E 0.050000

```
10 int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A vytvoří pole o celkovém počtu 49 prvků typu int
- B způsobí zápis za konec přidělené paměti
- C vytvoří pole o celkovém počtu 64 prvků typu int
- D vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		26

**1** #include <stdlib.h>

```
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- C pád programu
- D memory leak o velikosti 60 bajtů
- E žádná z odpovědí není správná
- F memory leak o velikosti 120 bajtů

**2** #include <stdio.h>

```
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Nelze přeložit
- B World Hello World Hello
- C Hello World Hello World
- D World Hello Hello Hello
- E Hello Hello Hello Hello
- F Žádná z odpovědí není správná

**3** int main() {

```
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A vytvoří pole o celkovém počtu 64 prvků typu int
- B vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C způsobí zápis za konec přidělené paměti
- D vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E žádná z odpovědí není správná
- F vytvoří pole o celkovém počtu 49 prvků typu int

**4** #include <stdlib.h>

```
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A vypíše World
- B žádná z odpovědí není správná
- C nezpůsobí žádný memory leak
- D vypíše Hello
- E memory leak o velikosti 10 bajtů

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- C Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D Typová konverze na bitové úrovni je typicky časově náročnější
- E Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- F Typová konverze na sémantické úrovni je typicky časově náročnější

**6** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Helord"
- B** Nelze přeložit
- C** Vypíše řetězec "Helrld"
- D** Vypíše řetězec "Hell"
- E** Vypíše řetězec "Hellorld"
- F** Vypíše řetězec "Hellrld"

**7** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Žádná z odpovědí není správná
- B** Vypíše hodnoty '1 1 6'
- C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D** Nelze přeložit
- E** Vypíše hodnoty '3 3 7'
- F** Vypíše hodnoty '3 3 6'

**8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

**9** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** Žádná z odpovědí není správná
- B** foo(2);foo(0);foo(5);foo(7);foo(8);
- C** foo(2);foo(3);foo(10);foo(7);
- D** foo(1);foo(4);foo(6);foo(10);foo(8);
- E** foo(5);foo(2);foo(5);foo(10);foo(7);

**10** `#include <stdio.h>`  
`int main() {`  
 `float a = (int) 3.95;`  
 `int x = a;`  
 `printf("%f", x - a);`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** nelze přeložit
- C** 3.950000
- D** 0.000000
- E** 0.050000

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		27

**1**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A vytvoří pole o celkovém počtu 49 prvků typu int
- B vytvoří pole o celkovém počtu 64 prvků typu int
- C způsobí zápis za konec přidělené paměti
- D žádná z odpovědí není správná
- E vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F vyplní všechny položky pole hodnotou z intervalu 1 do 7

**2**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A žádná z odpovědí není správná
- B 0.050000
- C 0.000000
- D 3.950000
- E nelze přeložit

**3**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(2);foo(3);foo(10);foo(7);
- B foo(1);foo(4);foo(6);foo(10);foo(8);
- C foo(5);foo(2);foo(5);foo(10);foo(7);
- D žádná z odpovědí není správná
- E foo(2);foo(0);foo(5);foo(7);foo(8);

**4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na sémantické úrovni je typicky časově náročnější
- B Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D Typová konverze na bitové úrovni je typicky časově náročnější
- E Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- F Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- B Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**6**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B Žádná z odpovědí není správná
- C Vypíše hodnoty '3 3 6'
- D Vypíše hodnoty '1 1 6'
- E Nelze přeložit
- F Vypíše hodnoty '3 3 7'

**7** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Nelze přeložit
- B** World Hello Hello Hello
- C** Hello Hello Hello Hello
- D** World Hello World Hello
- E** Žádná z odpovědí není správná
- F** Hello World Hello World

**8** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Hellrld"
- D** Nelze přeložit
- E** Vypíše řetězec "Hellorld"
- F** Vypíše řetězec "Helrld"

**9** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Žádná z odpovědí není správná
- B** memory leak o velikosti 10 bajtů
- C** vypíše World
- D** vypíše Hello
- E** nezpůsobí žádný memory leak

**10** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 120 bajtů
- B** memory leak o velikosti 60 bajtů
- C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** pád programu
- E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		28

**1** `#include <stdio.h>`  
`int main() {`  
    `float a = (int) 3.95;`  
    `int x = a;`  
    `printf("%f", x - a);`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** 3.950000
- B** žádná z odpovědí není správná
- C** 0.000000
- D** nelze přeložit
- E** 0.050000

**2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

**3** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`

Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** `foo(2);foo(0);foo(5);foo(7);foo(8);`
- C** `foo(5);foo(2);foo(5);foo(10);foo(7);`
- D** `foo(2);foo(3);foo(10);foo(7);`
- E** `foo(1);foo(4);foo(6);foo(10);foo(8);`

**4** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
    `*array = malloc(10);`  
    `array = NULL;`  
`}`  
`int main() {`  
    `int* array = NULL;`  
    `foo(&array);`  
    `if (array != NULL) printf("Hello");`  
    `else printf("World");`  
    `free(array);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** vypíše Hello
- C** nezpůsobí žádný memory leak
- D** vypíše World
- E** žádná z odpovědí není správná

**5** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`

Výše uvedený program:

- A** způsobí zápis za konec přidělené paměti
- B** žádná z odpovědí není správná
- C** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- D** vytvoří pole o celkovém počtu 64 prvků typu `int`
- E** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F** vytvoří pole o celkovém počtu 49 prvků typu `int`

**6** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Žádná z odpovědí není správná
- B** Vypíše hodnoty '3 3 7'
- C** Vypíše hodnoty '1 1 6'
- D** Vypíše hodnoty '3 3 6'
- E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- F** Nelze přeložit

**7** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Hellrld"
- B** Vypíše řetězec "Hell"
- C** Vypíše řetězec "Helrld"
- D** Vypíše řetězec "Helord"
- E** Nelze přeložit
- F** Vypíše řetězec "Helloworld"

**8** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 120 bajtů
- B** memory leak o velikosti 60 bajtů
- C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** pád programu
- F** žádná z odpovědí není správná

**9** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Nelze přeložit
- B** Žádná z odpovědí není správná
- C** Hello World Hello World
- D** World Hello Hello Hello
- E** World Hello World Hello
- F** Hello Hello Hello Hello

**10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni je typicky časově náročnější
- B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- F** Typová konverze na bitové úrovni je typicky časově náročnější



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		29

- 1**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše hodnoty '3 3 6'  
**B** Nelze přeložit  
**C** Vypíše hodnoty '3 3 7'  
**D** Žádná z odpovědí není správná  
**E** Vypíše hodnoty '1 1 6'  
**F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

- 2**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:
- A** World Hello World Hello  
**B** Hello World Hello World  
**C** Žádná z odpovědí není správná  
**D** Nelze přeložit  
**E** Hello Hello Hello Hello  
**F** World Hello Hello Hello

- 3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni je typicky časově náročnější  
**B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ  
**C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**D** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**E** Typová konverze na sémantické úrovni je typicky časově náročnější  
**F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

- 4**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:
- A** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)  
**B** pád programu  
**C** memory leak o velikosti 60 bajtů  
**D** memory leak o velikosti 120 bajtů  
**E** žádná z odpovědí není správná  
**F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

- 5**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:
- A** Žádná z odpovědí není správná  
**B** 0.000000  
**C** 3.950000  
**D** 0.050000  
**E** nelze přeložit

- 6**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:
- A** memory leak o velikosti 10 bajtů  
**B** vypíše Hello  
**C** vypíše World  
**D** žádná z odpovědí není správná  
**E** nezpůsobí žádný memory leak

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

**8**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 64 prvků typu int
- B** vytvoří pole o celkovém počtu 49 prvků typu int
- C** způsobí zápis za konec přidělené paměti
- D** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F** žádná z odpovědí není správná

**9**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(1);foo(4);foo(6);foo(10);foo(8);
- B** žádná z odpovědí není správná
- C** foo(2);foo(3);foo(10);foo(7);
- D** foo(2);foo(0);foo(5);foo(7);foo(8);
- E** foo(5);foo(2);foo(5);foo(10);foo(7);

**10**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Hellorld"
- C** Nelze přeložit
- D** Vypíše řetězec "Helrld"
- E** Vypíše řetězec "Hellrld"
- F** Vypíše řetězec "Helord"

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		30

- 1**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

 Spuštění programu způsobí:
- A** žádná z odpovědí není správná
  - B** memory leak o velikosti 120 bajtů
  - C** pád programu
  - D** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - F** memory leak o velikosti 60 bajtů

- 2**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

 Výše uvedený program:
- A** způsobí zápis za konec přidělené paměti
  - B** vytvoří pole o celkovém počtu 49 prvků typu int
  - C** vytvoří pole o celkovém počtu 64 prvků typu int
  - D** žádná z odpovědí není správná
  - E** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - F** vyplní všechny položky pole hodnotou z intervalu 1 do 7

- 3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni je typicky časově náročnější
  - C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - D** Typová konverze na bitové úrovni je typicky časově náročnější
  - E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

- 4**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

 Výše uvedený program vypíše:
- A** 0.050000
  - B** žádná z odpovědí není správná
  - C** nelze přeložit
  - D** 3.950000
  - E** 0.000000

- 5**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

 Výše uvedený program vypíše:
- A** World Hello Hello Hello
  - B** Hello Hello Hello Hello
  - C** Žádná z odpovědí není správná
  - D** Hello World Hello World
  - E** Nelze přeložit
  - F** World Hello World Hello

- 6**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

 Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(5);foo(2);foo(5);foo(10);foo(7);
  - B** foo(2);foo(0);foo(5);foo(7);foo(8);
  - C** foo(2);foo(3);foo(10);foo(7);
  - D** foo(1);foo(4);foo(6);foo(10);foo(8);
  - E** žádná z odpovědí není správná

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

**8**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** nezpůsobí žádný memory leak
- C** vypíše Hello
- D** žádná z odpovědí není správná
- E** vypíše World

**9**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hellrld"
- B** Vypíše řetězec "Hell"
- C** Vypíše řetězec "Hellorld"
- D** Nelze přeložit
- E** Vypíše řetězec "Helrld"
- F** Vypíše řetězec "Helord"

**10**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '3 3 6'
- B** Nelze přeložit
- C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D** Vypíše hodnoty '3 3 7'
- E** Žádná z odpovědí není správná
- F** Vypíše hodnoty '1 1 6'

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		31

**1**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hellorld"
- B** Vypíše řetězec "Helrld"
- C** Vypíše řetězec "Hell"
- D** Vypíše řetězec "Hellrld"
- E** Vypíše řetězec "Helord"
- F** Nelze přeložit

**2**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** způsobí zápis za konec přidělené paměti
- B** vytvoří pole o celkovém počtu 64 prvků typu int
- C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** žádná z odpovědí není správná
- F** vyplní všechny položky pole hodnotou z intervalu 1 do 7

**3**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 0.000000
- B** žádná z odpovědí není správná
- C** 3.950000
- D** nelze přeložit
- E** 0.050000

**4**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Žádná z odpovědí není správná
- B** Vypíše hodnoty '1 1 6'
- C** Vypíše hodnoty '3 3 6'
- D** Vypíše hodnoty '3 3 7'
- E** Nelze přeložit
- F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

**5**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** World Hello World Hello
- C** Hello World Hello World
- D** Hello Hello Hello Hello
- E** World Hello Hello Hello
- F** Nelze přeložit

- 6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - E** Typová konverze na sémantické úrovni je typicky časově náročnější
  - F** Typová konverze na bitové úrovni je typicky časově náročnější

- 7**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

 Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(1);foo(4);foo(6);foo(10);foo(8);
  - B** foo(2);foo(3);foo(10);foo(7);
  - C** žádná z odpovědí není správná
  - D** foo(5);foo(2);foo(5);foo(10);foo(7);
  - E** foo(2);foo(0);foo(5);foo(7);foo(8);

- 8**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

 Spuštění programu způsobí:
- A** pád programu
  - B** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - C** memory leak o velikosti 60 bajtů
  - D** žádná z odpovědí není správná
  - E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - F** memory leak o velikosti 120 bajtů

- 9**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

 Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** vypíše World
- C** žádná z odpovědí není správná
- D** vypíše Hello
- E** nezpůsobí žádný memory leak

- 10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		32

**1**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše Hello
- B** vypíše World
- C** žádná z odpovědí není správná
- D** nezpůsobí žádný memory leak
- E** memory leak o velikosti 10 bajtů

**2**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(5);foo(2);foo(5);foo(10);foo(7);
- B** foo(1);foo(4);foo(6);foo(10);foo(8);
- C** foo(2);foo(0);foo(5);foo(7);foo(8);
- D** žádná z odpovědí není správná
- E** foo(2);foo(3);foo(10);foo(7);

**3**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 120 bajtů
- B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** pád programu
- D** žádná z odpovědí není správná
- E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** memory leak o velikosti 60 bajtů

**4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**5**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 3.950000
- B** žádná z odpovědí není správná
- C** 0.050000
- D** nelze přeložit
- E** 0.000000

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na sémantické úrovni je typicky časově náročnější
- D** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- E** Typová konverze na bitové úrovni je typicky časově náročnější
- F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

```

7 #include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}

```

Výše uvedený program vypíše:

- A Hello World Hello World
- B World Hello World Hello
- C Žádná z odpovědí není správná
- D Nelze přeložit
- E Hello Hello Hello Hello
- F World Hello Hello Hello

```

8 #include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}

```

Výše uvedený program:

- A Nelze přeložit
- B Vypíše řetězec "Hellrld"
- C Vypíše řetězec "Hell"
- D Vypíše řetězec "Hellorld"
- E Vypíše řetězec "Helrld"
- F Vypíše řetězec "Helord"

```

9 int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}

```

Výše uvedený program:

- A vytvoří pole o celkovém počtu 49 prvků typu int
- B vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C způsobí zápis za konec přidělené paměti
- D vytvoří pole o celkovém počtu 64 prvků typu int
- E vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F Žádná z odpovědí není správná

```

10 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}

```

Výše uvedený program:

- A Žádná z odpovědí není správná
- B Vypíše hodnoty '3 3 7'
- C Nelze přeložit
- D Vypíše hodnoty '3 3 6'
- E Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- F Vypíše hodnoty '1 1 6'



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		33

- 1** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`  
Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":
- A** `foo(5);foo(2);foo(5);foo(10);foo(7);`  
**B** `foo(2);foo(3);foo(10);foo(7);`  
**C** `foo(1);foo(4);foo(6);foo(10);foo(8);`  
**D** žádná z odpovědí není správná  
**E** `foo(2);foo(0);foo(5);foo(7);foo(8);`

- 2** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d",array[0],array[6],value);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Nelze přeložit  
**B** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**C** Vypíše hodnoty '3 3 7'  
**D** Vypíše hodnoty '3 3 6'  
**E** Vypíše hodnoty '1 1 6'  
**F** Žádná z odpovědí není správná

- 3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind  
**B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel  
**C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel  
**D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

- 4** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7  
**B** vytvoří pole o celkovém počtu 64 prvků typu int  
**C** vytvoří pole o celkovém počtu 49 prvků typu int  
**D** vyplní všechny položky pole hodnotou z intervalu 1 do 7  
**E** žádná z odpovědí není správná  
**F** způsobí zápis za konec přidělené paměti

- 5** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** World Hello World Hello  
**B** Nelze přeložit  
**C** Hello World Hello World  
**D** World Hello Hello Hello  
**E** Hello Hello Hello Hello  
**F** Žádná z odpovědí není správná

- 6** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Nelze přeložit  
**B** Vypíše řetězec "Hellrld"  
**C** Vypíše řetězec "Helord"  
**D** Vypíše řetězec "Helrld"  
**E** Vypíše řetězec "Hellorld"  
**F** Vypíše řetězec "Hell"

**7**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** žádná z odpovědí není správná
- B** 3.950000
- C** 0.050000
- D** 0.000000
- E** nelze přeložit

**8**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 120 bajtů
- B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** pád programu
- D** žádná z odpovědí není správná
- E** memory leak o velikosti 60 bajtů
- F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B** Typová konverze na sémantické úrovni je typicky časově náročnější
- C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na bitové úrovni je typicky časově náročnější
- E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

**10**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** memory leak o velikosti 10 bajtů
- C** vypíše Hello
- D** vypíše World
- E** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		34

**1**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

 Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** vypíše World
- C** vypíše Hello
- D** nezpůsobí žádný memory leak
- E** žádná z odpovědí není správná

**2**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

 Výše uvedený program vypíše:

- A** Hello World Hello World
- B** Nelze přeložit
- C** Hello Hello Hello Hello
- D** Žádná z odpovědí není správná
- E** World Hello Hello Hello
- F** World Hello World Hello

**3**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

 Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Helrld"
- D** Vypíše řetězec "Hellrld"
- E** Nelze přeložit
- F** Vypíše řetězec "Helloworld"

**4**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

 Výše uvedený program:

- A** způsobí zápis za konec přidělené paměti
- B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C** žádná z odpovědí není správná
- D** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E** vytvoří pole o celkovém počtu 49 prvků typu int
- F** vytvoří pole o celkovém počtu 64 prvků typu int

**5**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

 Výše uvedený program vypíše:

- A** 0.000000
- B** 3.950000
- C** 0.050000
- D** nelze přeložit
- E** žádná z odpovědí není správná

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**7** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** memory leak o velikosti 60 bajtů
- D** memory leak o velikosti 120 bajtů
- E** pád programu
- F** žádná z odpovědí není správná

**8** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše hodnoty '3 3 7'
- B** Vypíše hodnoty '1 1 6'
- C** Vypíše hodnoty '3 3 6'
- D** Žádná z odpovědí není správná
- E** Nelze přeložit
- F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na bitové úrovni je typicky časově náročnější
- B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- C** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E** Typová konverze na sémantické úrovni je typicky časově náročnější
- F** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace

**10** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(2);foo(3);foo(10);foo(7);
- B** foo(5);foo(2);foo(5);foo(10);foo(7);
- C** foo(1);foo(4);foo(6);foo(10);foo(8);
- D** foo(2);foo(0);foo(5);foo(7);foo(8);
- E** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		35

- 1**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Nelze přeložit  
**B** Vypíše hodnoty '1 1 6'  
**C** Vypíše hodnoty '3 3 6'  
**D** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**E** Vypíše hodnoty '3 3 7'  
**F** Žádná z odpovědí není správná

- 2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel  
**B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel  
**C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel  
**D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

- 3**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** způsobí zápis za konec přidělené paměti  
**B** žádná z odpovědí není správná  
**C** vyplní všechny položky pole hodnotou z intervalu 1 do 7  
**D** vytvoří pole o celkovém počtu 49 prvků typu int  
**E** vytvoří pole o celkovém počtu 64 prvků typu int  
**F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

- 4**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:
- A** vypíše Hello  
**B** memory leak o velikosti 10 bajtů  
**C** vypíše World  
**D** nezpůsobí žádný memory leak  
**E** žádná z odpovědí není správná

- 5**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(2);foo(3);foo(10);foo(7);  
**B** foo(2);foo(0);foo(5);foo(7);foo(8);  
**C** foo(1);foo(4);foo(6);foo(10);foo(8);  
**D** foo(5);foo(2);foo(5);foo(10);foo(7);  
**E** žádná z odpovědí není správná

```

6 #include <stdio.h>
  int main() {
      unsigned char value1 = 0x55;
      unsigned char value2 = 0xAA;
      if (value1 & value2) printf("Hello ");
      else printf("World ");
      if (value1 && value2) printf("Hello ");
      else printf("World ");
      if (value1 | value2) printf("Hello ");
      else printf("World ");
      if (value1 || value2) printf("Hello ");
      else printf("World ");
      return 0;
  }

```

Výše uvedený program vypíše:

- A Hello World Hello World
- B World Hello World Hello
- C Nelze přeložit
- D World Hello Hello Hello
- E Hello Hello Hello Hello
- F Žádná z odpovědí není správná

```

7 #include <stdio.h>
  int main() {
      float a = (int) 3.95;
      int x = a;
      printf("%f", x - a);
      return 0;
  }

```

Výše uvedený program vypíše:

- A 0.050000
- B 3.950000
- C 0.000000
- D žádná z odpovědí není správná
- E nelze přeložit

```

8 #include <stdlib.h>
  int main() {
      int* pArray1 = NULL;
      int* pArray2 = NULL;
      int* pArray3 = NULL;
      pArray1 = malloc(60);
      pArray2 = pArray1;
      pArray1 += 8;
      pArray3 = malloc(60);
      free(pArray2);
      pArray2 = pArray3;
      free(pArray2);
      return 0;
  }

```

Spuštění programu způsobí:

- A memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B pád programu
- C memory leak o velikosti 60 bajtů
- D memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E memory leak o velikosti 120 bajtů
- F žádná z odpovědí není správná

```

9 #include <stdio.h>
  #include <string.h>
  int main() {
      char str1[20];
      strcpy(str1, "Hello world");
      str1[4] = 0;
      printf("%s", str1);
      char* str2 = str1 + 8;
      printf("%s", str2);
      return 0;
  }

```

Výše uvedený program:

- A Vypíše řetězec "Helrld"
- B Vypíše řetězec "Helord"
- C Nelze přeložit
- D Vypíše řetězec "Hellrld"
- E Vypíše řetězec "Hellorld"
- F Vypíše řetězec "Hell"

10 Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- D Typová konverze na bitové úrovni je typicky časově náročnější
- E Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F Typová konverze na sémantické úrovni je typicky časově náročnější

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		36

**1** #include <stdio.h>

```
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A 0.000000
- B 3.950000
- C 0.050000
- D žádná z odpovědí není správná
- E nelze přeložit

**2** #include <stdio.h>

```
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(2);foo(0);foo(5);foo(7);foo(8);
- B foo(1);foo(4);foo(6);foo(10);foo(8);
- C žádná z odpovědí není správná
- D foo(2);foo(3);foo(10);foo(7);
- E foo(5);foo(2);foo(5);foo(10);foo(7);

**3** int main() {  
 int array[7][7];  
 for (int i = 1; i <= 7; i++) {  
 for (int j = 1; j <= 7; j++) array[i][j] = i;  
 }  
 return 0;  
}

Výše uvedený program:

- A způsobí zápis za konec přidělené paměti
- B vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C vytvoří pole o celkovém počtu 49 prvků typu int
- D vytvoří pole o celkovém počtu 64 prvků typu int
- E žádná z odpovědí není správná
- F vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

**4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- B Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- D Typová konverze na sémantické úrovni je typicky časově náročnější
- E Typová konverze na bitové úrovni je typicky časově náročnější
- F Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace

**5** #include <stdio.h>

```
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Žádná z odpovědí není správná
- B Nelze přeložit
- C Hello Hello Hello Hello
- D World Hello World Hello
- E World Hello Hello Hello
- F Hello World Hello World

**6** #include <stdlib.h>

```
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A vypíše Hello
- B memory leak o velikosti 10 bajtů
- C vypíše World
- D nezpůsobí žádný memory leak
- E žádná z odpovědí není správná

```

7 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}

```

Výše uvedený program:

- A Vypíše hodnoty '3 3 6'
- B Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C Vypíše hodnoty '3 3 7'
- D Nelze přeložit
- E Žádná z odpovědí není správná
- F Vypíše hodnoty '1 1 6'

```

8 #include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}

```

Výše uvedený program:

- A Vypíše řetězec "Hell"
- B Nelze přeložit
- C Vypíše řetězec "Hellrld"
- D Vypíše řetězec "Hellorld"
- E Vypíše řetězec "Helrld"
- F Vypíše řetězec "Helord"

```

9 #include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}

```

Spuštění programu způsobí:

- A memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B memory leak o velikosti 60 bajtů
- C pád programu
- D memory leak o velikosti 120 bajtů
- E žádná z odpovědí není správná
- F memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

10 Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		37

**1**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Hello Hello Hello Hello
- B Nelze přeložit
- C Hello World Hello World
- D World Hello World Hello
- E Žádná z odpovědí není správná
- F World Hello Hello Hello

**2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B Typová konverze na bitové úrovni je typicky časově náročnější
- C Typová konverze na sémantické úrovni je typicky časově náročnější
- D Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- E Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

**3**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A memory leak o velikosti 120 bajtů
- B memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- C memory leak o velikosti 60 bajtů
- D žádná z odpovědí není správná
- E pád programu
- F memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)

**4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- C Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**5**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A žádná z odpovědí není správná
- B memory leak o velikosti 10 bajtů
- C vypíše World
- D vypíše Hello
- E nezpůsobí žádný memory leak

**6**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A Žádná z odpovědí není správná
- B Vypíše hodnoty '3 3 6'
- C Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D Nelze přeložit
- E Vypíše hodnoty '3 3 7'
- F Vypíše hodnoty '1 1 6'

**7** `#include <stdio.h>`  
`int main() {`  
    `float a = (int) 3.95;`  
    `int x = a;`  
    `printf("%f", x - a);`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** 0.050000
- B** žádná z odpovědí není správná
- C** nelze přeložit
- D** 0.000000
- E** 3.950000

**8** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`

Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":

- A** `foo(2);foo(0);foo(5);foo(7);foo(8);`
- B** žádná z odpovědí není správná
- C** `foo(5);foo(2);foo(5);foo(10);foo(7);`
- D** `foo(1);foo(4);foo(6);foo(10);foo(8);`
- E** `foo(2);foo(3);foo(10);foo(7);`

**9** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 64 prvků typu `int`
- B** vytvoří pole o celkovém počtu 49 prvků typu `int`
- C** způsobí zápis za konec přidělené paměti
- D** žádná z odpovědí není správná
- E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

**10** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše řetězec "Helrld"
- C** Vypíše řetězec "Hellrld"
- D** Vypíše řetězec "Hellorld"
- E** Vypíše řetězec "Hell"
- F** Vypíše řetězec "Helord"

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		38

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - D** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Typová konverze na sémantické úrovni je typicky časově náročnější
  - F** Typová konverze na bitové úrovni je typicky časově náročnější

- 2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

- 3**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:
- A** žádná z odpovědí není správná
  - B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - C** pád programu
  - D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - E** memory leak o velikosti 60 bajtů
  - F** memory leak o velikosti 120 bajtů

- 4**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** žádná z odpovědí není správná
  - B** foo(1);foo(4);foo(6);foo(10);foo(8);
  - C** foo(2);foo(3);foo(10);foo(7);
  - D** foo(5);foo(2);foo(5);foo(10);foo(7);
  - E** foo(2);foo(0);foo(5);foo(7);foo(8);

- 5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** vytvoří pole o celkovém počtu 49 prvků typu int
  - B** způsobí zápis za konec přidělené paměti
  - C** vytvoří pole o celkovém počtu 64 prvků typu int
  - D** žádná z odpovědí není správná
  - E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

- 6**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:
- A** memory leak o velikosti 10 bajtů
  - B** nezpůsobí žádný memory leak
  - C** vypíše Hello
  - D** vypíše World
  - E** žádná z odpovědí není správná

- 7** `#include <stdio.h>`  
`int main() {`  
 `float a = (int) 3.95;`  
 `int x = a;`  
 `printf("%f", x - a);`  
 `return 0;`  
`}`  
 Výše uvedený program vypíše:
- A** 0.050000
  - B** 0.000000
  - C** nelze přeložit
  - D** žádná z odpovědí není správná
  - E** 3.950000

- 8** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`  
 Výše uvedený program:
- A** Vypíše řetězec "Helrld"
  - B** Vypíše řetězec "Helord"
  - C** Vypíše řetězec "Hellorld"
  - D** Vypíše řetězec "Hellrld"
  - E** Nelze přeložit
  - F** Vypíše řetězec "Hell"

- 9** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`  
 Výše uvedený program vypíše:
- A** Nelze přeložit
  - B** World Hello World Hello
  - C** Hello World Hello World
  - D** World Hello Hello Hello
  - E** Žádná z odpovědí není správná
  - F** Hello Hello Hello Hello

- 10** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`  
 Výše uvedený program:
- A** Nelze přeložit
  - B** Žádná z odpovědí není správná
  - C** Vypíše hodnoty '3 3 7'
  - D** Vypíše hodnoty '3 3 6'
  - E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
  - F** Vypíše hodnoty '1 1 6'

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		39

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

**2**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** World Hello Hello Hello
- C** Nelze přeložit
- D** Hello Hello Hello Hello
- E** Hello World Hello World
- F** World Hello World Hello

**3**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B** Nelze přeložit
- C** Vypíše hodnoty '3 3 7'
- D** Vypíše hodnoty '3 3 6'
- E** Žádná z odpovědí není správná
- F** Vypíše hodnoty '1 1 6'

**4**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** nelze přeložit
- B** 3.950000
- C** 0.050000
- D** žádná z odpovědí není správná
- E** 0.000000

**5**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(5);foo(2);foo(5);foo(10);foo(7);
- B** foo(2);foo(0);foo(5);foo(7);foo(8);
- C** foo(2);foo(3);foo(10);foo(7);
- D** žádná z odpovědí není správná
- E** foo(1);foo(4);foo(6);foo(10);foo(8);

**6**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Helrld"
- B** Vypíše řetězec "Hellorld"
- C** Vypíše řetězec "Helord"
- D** Vypíše řetězec "Hell"
- E** Vypíše řetězec "Hellrld"
- F** Nelze přeložit

**7**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** způsobí zápis za konec přidělené paměti
- B** žádná z odpovědí není správná
- C** vytvoří pole o celkovém počtu 49 prvků typu int
- D** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E** vytvoří pole o celkovém počtu 64 prvků typu int
- F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

**8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B** Typová konverze na sémantické úrovni je typicky časově náročnější
- C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na bitové úrovni je typicky časově náročnější
- E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

**9**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** žádná z odpovědí není správná
- B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** pád programu
- D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** memory leak o velikosti 120 bajtů
- F** memory leak o velikosti 60 bajtů

**10**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše Hello
- B** vypíše World
- C** nezpůsobí žádný memory leak
- D** memory leak o velikosti 10 bajtů
- E** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		40

**1**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše Hello
- B** memory leak o velikosti 10 bajtů
- C** žádná z odpovědí není správná
- D** vypíše World
- E** nezpůsobí žádný memory leak

**2**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Hellorld"
- C** Vypíše řetězec "Helrld"
- D** Vypíše řetězec "Helord"
- E** Nelze přeložit
- F** Vypíše řetězec "Hellrld"

**3**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** World Hello World Hello
- C** Nelze přeložit
- D** World Hello Hello Hello
- E** Hello Hello Hello Hello
- F** Hello World Hello World

**4**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(1);foo(4);foo(6);foo(10);foo(8);
- B** foo(5);foo(2);foo(5);foo(10);foo(7);
- C** foo(2);foo(3);foo(10);foo(7);
- D** žádná z odpovědí není správná
- E** foo(2);foo(0);foo(5);foo(7);foo(8);

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni je typicky časově náročnější
- B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na bitové úrovni je typicky časově náročnější
- D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- F** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace

```

6 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}

```

Výše uvedený program:

- A Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B Vypíše hodnoty '3 3 6'
- C Vypíše hodnoty '3 3 7'
- D Žádná z odpovědí není správná
- E Vypíše hodnoty '1 1 6'
- F Nelze přeložit

7 Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

```

8 int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}

```

Výše uvedený program:

- A vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- B žádná z odpovědí není správná
- C způsobí zápis za konec přidělené paměti
- D vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E vytvoří pole o celkovém počtu 64 prvků typu int
- F vytvoří pole o celkovém počtu 49 prvků typu int

```

9 #include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}

```

Spuštění programu způsobí:

- A memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B žádná z odpovědí není správná
- C pád programu
- D memory leak o velikosti 60 bajtů
- E memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- F memory leak o velikosti 120 bajtů

```

10 #include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}

```

Výše uvedený program vypíše:

- A 3.950000
- B nelze přeložit
- C 0.000000
- D žádná z odpovědí není správná
- E 0.050000



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		41

- 1**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Žádná z odpovědí není správná  
**B** Nelze přeložit  
**C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**D** Vypíše hodnoty '3 3 7'  
**E** Vypíše hodnoty '3 3 6'  
**F** Vypíše hodnoty '1 1 6'

- 2**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:
- A** Žádná z odpovědí není správná  
**B** World Hello Hello Hello  
**C** Hello Hello Hello Hello  
**D** Nelze přeložit  
**E** Hello World Hello World  
**F** World Hello World Hello

- 3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel  
**B** Memory leak lze detekovat s využitím nástrojů, např. Valgrind  
**C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel  
**D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

- 4**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:
- A** nezpůsobí žádný memory leak  
**B** memory leak o velikosti 10 bajtů  
**C** vypíše Hello  
**D** žádná z odpovědí není správná  
**E** vypíše World

- 5**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše řetězec "Hellrld"  
**B** Vypíše řetězec "Hell"  
**C** Vypíše řetězec "Helrld"  
**D** Vypíše řetězec "Helord"  
**E** Nelze přeložit  
**F** Vypíše řetězec "Helloworld"

- 6**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** způsobí zápis za konec přidělené paměti  
**B** vytvoří pole o celkovém počtu 49 prvků typu int  
**C** vytvoří pole o celkovém počtu 64 prvků typu int  
**D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7  
**E** vyplní všechny položky pole hodnotou z intervalu 1 do 7  
**F** žádná z odpovědí není správná

```
7 #include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A žádná z odpovědí není správná
- B memory leak o velikosti 120 bajtů
- C memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- D pád programu
- E memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F memory leak o velikosti 60 bajtů

```
8 #include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A žádná z odpovědí není správná
- B foo(1);foo(4);foo(6);foo(10);foo(8);
- C foo(2);foo(3);foo(10);foo(7);
- D foo(2);foo(0);foo(5);foo(7);foo(8);
- E foo(5);foo(2);foo(5);foo(10);foo(7);

```
9 #include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A 0.000000
- B 0.050000
- C 3.950000
- D nelze přeložit
- E žádná z odpovědí není správná

10 Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B Typová konverze na sémantické úrovni je typicky časově náročnější
- C Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- D Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- E Typová konverze na bitové úrovni je typicky časově náročnější
- F Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		42

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni je typicky časově náročnější
  - C** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - D** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - F** Typová konverze na bitové úrovni je typicky časově náročnější

- 2**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:
- A** 3.950000
  - B** 0.050000
  - C** žádná z odpovědí není správná
  - D** 0.000000
  - E** nelze přeložit

- 3**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše hodnoty '3 3 7'
  - B** Nelze přeložit
  - C** Vypíše hodnoty '3 3 6'
  - D** Vypíše hodnoty '1 1 6'
  - E** Žádná z odpovědí není správná
  - F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

- 4**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:
- A** nezpůsobí žádný memory leak
  - B** vypíše World
  - C** žádná z odpovědí není správná
  - D** memory leak o velikosti 10 bajtů
  - E** vypíše Hello

- 5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** vytvoří pole o celkovém počtu 64 prvků typu int
  - B** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - C** vytvoří pole o celkovém počtu 49 prvků typu int
  - D** způsobí zápis za konec přidělené paměti
  - E** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - F** žádná z odpovědí není správná

- 6**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:
- A** memory leak o velikosti 60 bajtů
  - B** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - C** memory leak o velikosti 120 bajtů
  - D** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - E** žádná z odpovědí není správná
  - F** pád programu

- 7** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`  
 Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":
- A** Žádná z odpovědí není správná
  - B** `foo(5);foo(2);foo(5);foo(10);foo(7);`
  - C** `foo(2);foo(3);foo(10);foo(7);`
  - D** `foo(1);foo(4);foo(6);foo(10);foo(8);`
  - E** `foo(2);foo(0);foo(5);foo(7);foo(8);`

- 8** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`  
 Výše uvedený program vypíše:
- A** World Hello Hello Hello
  - B** Hello Hello Hello Hello
  - C** Hello World Hello World
  - D** Žádná z odpovědí není správná
  - E** Nelze přeložit
  - F** World Hello World Hello

- 9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

- 10** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`  
 Výše uvedený program:
- A** Vypíše řetězec "Helrld"
  - B** Vypíše řetězec "Hellorld"
  - C** Vypíše řetězec "Hellrld"
  - D** Nelze přeložit
  - E** Vypíše řetězec "Helord"
  - F** Vypíše řetězec "Hell"

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		43

**1**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A vypíše Hello
- B vypíše World
- C žádná z odpovědí není správná
- D memory leak o velikosti 10 bajtů
- E nezpůsobí žádný memory leak

**2**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A Vypíše hodnoty '3 3 7'
- B Nelze přeložit
- C Vypíše hodnoty '1 1 6'
- D Žádná z odpovědí není správná
- E Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- F Vypíše hodnoty '3 3 6'

**3**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Hello World Hello World
- B World Hello World Hello
- C Hello Hello Hello Hello
- D Žádná z odpovědí není správná
- E Nelze přeložit
- F World Hello Hello Hello

- 4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - B Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - D Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A vytvoří pole o celkovém počtu 64 prvků typu int
- B vytvoří pole o celkovém počtu 49 prvků typu int
- C vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D způsobí zápis za konec přidělené paměti
- E vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F žádná z odpovědí není správná

- 6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni je typicky časově náročnější
  - B** Typová konverze na sémantické úrovni je typicky časově náročnější
  - C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

- 7** `#include <stdlib.h>`  
`int main() {`  
`int* pArray1 = NULL;`  
`int* pArray2 = NULL;`  
`int* pArray3 = NULL;`  
`pArray1 = malloc(60);`  
`pArray2 = pArray1;`  
`pArray1 += 8;`  
`pArray3 = malloc(60);`  
`free(pArray2);`  
`pArray2 = pArray3;`  
`free(pArray2);`  
`return 0;`  
`}`
- Spuštění programu způsobí:
- A** žádná z odpovědí není správná
  - B** pád programu
  - C** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)
  - D** memory leak o velikosti 120 bajtů
  - E** memory leak o velikosti 60 bajtů
  - F** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)

- 8** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
`char str1[20];`  
`strcpy(str1, "Hello world");`  
`str1[4] = 0;`  
`printf("%s", str1);`  
`char* str2 = str1 + 8;`  
`printf("%s", str2);`  
`return 0;`  
`}`
- Výše uvedený program:
- A** Vypíše řetězec "Helord"
  - B** Vypíše řetězec "Helrld"
  - C** Vypíše řetězec "Hellrld"
  - D** Vypíše řetězec "Hellorld"
  - E** Nelze přeložit
  - F** Vypíše řetězec "Hell"

- 9** `#include <stdio.h>`  
`int main() {`  
`float a = (int) 3.95;`  
`int x = a;`  
`printf("%f", x - a);`  
`return 0;`  
`}`
- Výše uvedený program vypíše:
- A** 3.950000
  - B** 0.000000
  - C** žádná z odpovědí není správná
  - D** nelze přeložit
  - E** 0.050000

- 10** `#include <stdio.h>`  
`void foo(int a) {`  
`switch (a) {`  
`case 0: break;`  
`case 1: printf("Fr"); break;`  
`case 2: printf("F");`  
`case 3: printf("re"); break;`  
`case 4: printf("e");`  
`case 5:`  
`case 6: break;`  
`case 7:`  
`case 8: printf("m"); break;`  
`default: printf("edo");`  
`}`  
`}`
- Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":
- A** `foo(1);foo(4);foo(6);foo(10);foo(8);`
  - B** žádná z odpovědí není správná
  - C** `foo(2);foo(0);foo(5);foo(7);foo(8);`
  - D** `foo(5);foo(2);foo(5);foo(10);foo(7);`
  - E** `foo(2);foo(3);foo(10);foo(7);`

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		44

**1**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 64 prvků typu int
- B** žádná z odpovědí není správná
- C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F** způsobí zápis za konec přidělené paměti

**2**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(2);foo(0);foo(5);foo(7);foo(8);
- B** foo(5);foo(2);foo(5);foo(10);foo(7);
- C** foo(1);foo(4);foo(6);foo(10);foo(8);
- D** foo(2);foo(3);foo(10);foo(7);
- E** žádná z odpovědí není správná

**3**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Hello Hello Hello Hello
- B** Nelze přeložit
- C** Žádná z odpovědí není správná
- D** Hello World Hello World
- E** World Hello World Hello
- F** World Hello Hello Hello

**4**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** pád programu
- C** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** memory leak o velikosti 120 bajtů
- E** memory leak o velikosti 60 bajtů
- F** žádná z odpovědí není správná

**5**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 0.050000
- B** nelze přeložit
- C** žádná z odpovědí není správná
- D** 0.000000
- E** 3.950000

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

```

7 #include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}

```

Výše uvedený program:

- A nezpůsobí žádný memory leak
- B vypíše Hello
- C memory leak o velikosti 10 bajtů
- D žádná z odpovědí není správná
- E vypíše World

8 Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na sémantické úrovni je typicky časově náročnější
- B Typová konverze na bitové úrovni je typicky časově náročnější
- C Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- E Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

```

9 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}

```

Výše uvedený program:

- A Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B Vypíše hodnoty '3 3 6'
- C Vypíše hodnoty '1 1 6'
- D Vypíše hodnoty '3 3 7'
- E Žádná z odpovědí není správná
- F Nelze přeložit

```

10 #include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}

```

Výše uvedený program:

- A Vypíše řetězec "Helrld"
- B Vypíše řetězec "Hellorld"
- C Vypíše řetězec "Helord"
- D Vypíše řetězec "Hell"
- E Vypíše řetězec "Hellrld"
- F Nelze přeložit



# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		45

- 1**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Žádná z odpovědí není správná  
**B** Vypíše hodnoty '3 3 6'  
**C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**D** Nelze přeložit  
**E** Vypíše hodnoty '3 3 7'  
**F** Vypíše hodnoty '1 1 6'

- 2**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše řetězec "Hellorld"  
**B** Vypíše řetězec "Helrld"  
**C** Vypíše řetězec "Hellrld"  
**D** Vypíše řetězec "Helord"  
**E** Vypíše řetězec "Hell"  
**F** Nelze přeložit

- 3**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7  
**B** žádná z odpovědí není správná  
**C** vytvoří pole o celkovém počtu 49 prvků typu int  
**D** způsobí zápis za konec přidělené paměti  
**E** vytvoří pole o celkovém počtu 64 prvků typu int  
**F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

- 4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ  
**B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem  
**D** Typová konverze na bitové úrovni je typicky časově náročnější  
**E** Typová konverze na sémantické úrovni je typicky časově náročnější  
**F** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace

- 5**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:
- A** pád programu  
**B** žádná z odpovědí není správná  
**C** memory leak o velikosti 60 bajtů  
**D** memory leak o velikosti 120 bajtů  
**E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)  
**F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

- 6**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** žádná z odpovědí není správná  
**B** foo(2);foo(0);foo(5);foo(7);foo(8);  
**C** foo(2);foo(3);foo(10);foo(7);  
**D** foo(5);foo(2);foo(5);foo(10);foo(7);  
**E** foo(1);foo(4);foo(6);foo(10);foo(8);

```
7 #include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Nelze přeložit
- B Hello World Hello World
- C Hello Hello Hello Hello
- D Žádná z odpovědí není správná
- E World Hello Hello Hello
- F World Hello World Hello

```
8 #include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A memory leak o velikosti 10 bajtů
- B vypíše World
- C nezpůsobí žádný memory leak
- D vypíše Hello
- E žádná z odpovědí není správná

9 Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- D Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

```
10 #include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A žádná z odpovědí není správná
- B 0.000000
- C 0.050000
- D 3.950000
- E nelze přeložit

# PB071: Prubezna\_20111024

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		46

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - D** Typová konverze na sémantické úrovni je typicky časově náročnější
  - E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Typová konverze na bitové úrovni je typicky časově náročnější

- 2**
- ```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```
- Výše uvedený program:
- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
  - B** Vypíše hodnoty '1 1 6'
  - C** Vypíše hodnoty '3 3 7'
  - D** Vypíše hodnoty '3 3 6'
  - E** Žádná z odpovědí není správná
  - F** Nelze přeložit

- 3**
- ```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```
- Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(1);foo(4);foo(6);foo(10);foo(8);
  - B** Žádná z odpovědí není správná
  - C** foo(5);foo(2);foo(5);foo(10);foo(7);
  - D** foo(2);foo(0);foo(5);foo(7);foo(8);
  - E** foo(2);foo(3);foo(10);foo(7);

- 4**
- ```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```
- Výše uvedený program:
- A** Vypíše řetězec "Hellrld"
  - B** Vypíše řetězec "Helord"
  - C** Vypíše řetězec "Hellerld"
  - D** Vypíše řetězec "Hell"
  - E** Nelze přeložit
  - F** Vypíše řetězec "Helrld"

**5**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** memory leak o velikosti 60 bajtů
- C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** memory leak o velikosti 120 bajtů
- E** pád programu
- F** žádná z odpovědí není správná

**6**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** World Hello World Hello
- B** Žádná z odpovědí není správná
- C** Hello Hello Hello Hello
- D** Hello World Hello World
- E** Nelze přeložit
- F** World Hello Hello Hello

**7**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 0.000000
- B** 3.950000
- C** Žádná z odpovědí není správná
- D** 0.050000
- E** nelze přeložit

**8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**9**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše Hello
- B** memory leak o velikosti 10 bajtů
- C** vypíše World
- D** nezpůsobí žádný memory leak
- E** žádná z odpovědí není správná

**10**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B** vytvoří pole o celkovém počtu 49 prvků typu int
- C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D** žádná z odpovědí není správná
- E** vytvoří pole o celkovém počtu 64 prvků typu int
- F** způsobí zápis za konec přidělené paměti

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 47           |

**1**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** žádná z odpovědí není správná
- C** memory leak o velikosti 10 bajtů
- D** vypíše Hello
- E** vypíše World

**2**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 60 bajtů
- B** memory leak o velikosti 120 bajtů
- C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** pád programu
- E** žádná z odpovědí není správná
- F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

**3**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(2);foo(3);foo(10);foo(7);
- B** foo(1);foo(4);foo(6);foo(10);foo(8);
- C** foo(5);foo(2);foo(5);foo(10);foo(7);
- D** foo(2);foo(0);foo(5);foo(7);foo(8);
- E** žádná z odpovědí není správná

**4**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 49 prvků typu int
- B** žádná z odpovědí není správná
- C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D** způsobí zápis za konec přidělené paměti
- E** vytvoří pole o celkovém počtu 64 prvků typu int
- F** vyplní všechny položky pole hodnotou z intervalu 1 do 7

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na bitové úrovni je typicky časově náročnější
- B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Typová konverze na sémantické úrovni je typicky časově náročnější

**6** `#include <stdio.h>`  
`int main() {`  
 `float a = (int) 3.95;`  
 `int x = a;`  
 `printf("%f", x - a);`  
 `return 0;`  
`}`  
 Výše uvedený program vypíše:

- A** 0.000000
- B** 0.050000
- C** 3.950000
- D** nelze přeložit
- E** žádná z odpovědí není správná

**7** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`  
 Výše uvedený program vypíše:

- A** World Hello Hello Hello
- B** Hello World Hello World
- C** Žádná z odpovědí není správná
- D** Hello Hello Hello Hello
- E** World Hello World Hello
- F** Nelze přeložit

**8** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`  
 Výše uvedený program:

- A** Vypíše hodnoty '3 3 7'
- B** Žádná z odpovědí není správná
- C** Nelze přeložit
- D** Vypíše hodnoty '1 1 6'
- E** Vypíše hodnoty '3 3 6'
- F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

**10** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`  
 Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Hellorld"
- D** Vypíše řetězec "Hell"
- E** Vypíše řetězec "Helrld"
- F** Vypíše řetězec "Hellrld"

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 48           |

**1**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hellrld"
- B** Vypíše řetězec "Hell"
- C** Vypíše řetězec "Hellorld"
- D** Vypíše řetězec "Helord"
- E** Nelze přeložit
- F** Vypíše řetězec "Helrld"

**2**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- B** vytvoří pole o celkovém počtu 64 prvků typu int
- C** vytvoří pole o celkovém počtu 49 prvků typu int
- D** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E** způsobí zápis za konec přidělené paměti
- F** žádná z odpovědí není správná

**3**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Žádná z odpovědí není správná
- B** Vypíše hodnoty '3 3 7'
- C** Nelze přeložit
- D** Vypíše hodnoty '3 3 6'
- E** Vypíše hodnoty '1 1 6'
- F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

**4**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(2);foo(0);foo(5);foo(7);foo(8);
- B** žádná z odpovědí není správná
- C** foo(2);foo(3);foo(10);foo(7);
- D** foo(1);foo(4);foo(6);foo(10);foo(8);
- E** foo(5);foo(2);foo(5);foo(10);foo(7);

**5**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše World
- B** nezpůsobí žádný memory leak
- C** memory leak o velikosti 10 bajtů
- D** žádná z odpovědí není správná
- E** vypíše Hello

**6** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Nelze přeložit
- B** Hello Hello Hello Hello
- C** Hello World Hello World
- D** World Hello World Hello
- E** World Hello Hello Hello
- F** Žádná z odpovědí není správná

**7** `#include <stdio.h>`  
`int main() {`  
    `float a = (int) 3.95;`  
    `int x = a;`  
    `printf("%f", x - a);`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** 0.000000
- B** 3.950000
- C** žádná z odpovědí není správná
- D** 0.050000
- E** nelze přeložit

**8** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`

Spuštění programu způsobí:

- A** žádná z odpovědí není správná
- B** memory leak o velikosti 120 bajtů
- C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** memory leak o velikosti 60 bajtů
- E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** pád programu

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na bitové úrovni je typicky časově náročnější
- D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- E** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Typová konverze na sémantické úrovni je typicky časově náročnější

**10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 49           |

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na bitové úrovni je typicky časově náročnější
  - C** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Typová konverze na sémantické úrovni je typicky časově náročnější

**2**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** World Hello Hello Hello
- B** Nelze přeložit
- C** World Hello World Hello
- D** Žádná z odpovědí není správná
- E** Hello Hello Hello Hello
- F** Hello World Hello World

**3**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** memory leak o velikosti 120 bajtů
- C** pád programu
- D** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** žádná z odpovědí není správná
- F** memory leak o velikosti 60 bajtů

**4**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** žádná z odpovědí není správná
- B** 0.050000
- C** nelze přeložit
- D** 0.000000
- E** 3.950000

**5**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '1 1 6'
- B** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C** Vypíše hodnoty '3 3 7'
- D** Žádná z odpovědí není správná
- E** Nelze přeložit
- F** Vypíše hodnoty '3 3 6'

- 6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

**7**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** foo(5);foo(2);foo(5);foo(10);foo(7);
- C** foo(1);foo(4);foo(6);foo(10);foo(8);
- D** foo(2);foo(3);foo(10);foo(7);
- E** foo(2);foo(0);foo(5);foo(7);foo(8);

**8**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše World
- B** nezpůsobí žádný memory leak
- C** žádná z odpovědí není správná
- D** memory leak o velikosti 10 bajtů
- E** vypíše Hello

**9**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Hellorld"
- C** Vypíše řetězec "Hellrld"
- D** Vypíše řetězec "Helrld"
- E** Vypíše řetězec "Helord"
- F** Nelze přeložit

**10**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** způsobí zápis za konec přidělené paměti
- C** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F** vytvoří pole o celkovém počtu 64 prvků typu int

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 50           |

**1** #include <stdio.h>  
int main() {  
 unsigned char value1 = 0x55;  
 unsigned char value2 = 0xAA;  
 if (value1 & value2) printf("Hello ");  
 else printf("World ");  
 if (value1 && value2) printf("Hello ");  
 else printf("World ");  
 if (value1 | value2) printf("Hello ");  
 else printf("World ");  
 if (value1 || value2) printf("Hello ");  
 else printf("World ");  
 return 0;  
}

Výše uvedený program vypíše:

- A Hello Hello Hello Hello
- B Žádná z odpovědí není správná
- C Hello World Hello World
- D Nelze přeložit
- E World Hello World Hello
- F World Hello Hello Hello

**2** #include <stdio.h>  
int main() {  
 float a = (int) 3.95;  
 int x = a;  
 printf("%f", x - a);  
 return 0;  
}

Výše uvedený program vypíše:

- A nelze přeložit
- B 0.000000
- C Žádná z odpovědí není správná
- D 3.950000
- E 0.050000

**3** #include <stdio.h>  
void foo(int a) {  
 switch (a) {  
 case 0: break;  
 case 1: printf("Fr"); break;  
 case 2: printf("F");  
 case 3: printf("re"); break;  
 case 4: printf("e");  
 case 5:  
 case 6: break;  
 case 7:  
 case 8: printf("m"); break;  
 default: printf("edo");  
 }  
}

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(2);foo(0);foo(5);foo(7);foo(8);
- B foo(5);foo(2);foo(5);foo(10);foo(7);
- C foo(1);foo(4);foo(6);foo(10);foo(8);
- D foo(2);foo(3);foo(10);foo(7);
- E Žádná z odpovědí není správná

**4** #include <stdio.h>  
#include <string.h>  
int main() {  
 char str1[20];  
 strcpy(str1, "Hello world");  
 str1[4] = 0;  
 printf("%s", str1);  
 char\* str2 = str1 + 8;  
 printf("%s", str2);  
 return 0;  
}

Výše uvedený program:

- A Vypíše řetězec "Hell"
- B Vypíše řetězec "Hellrld"
- C Vypíše řetězec "Hellorld"
- D Vypíše řetězec "Helord"
- E Vypíše řetězec "Helrld"
- F Nelze přeložit

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B Typová konverze na sémantické úrovni je typicky časově náročnější
- C Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E Typová konverze na bitové úrovni je typicky časově náročnější
- F Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

**6** #include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
void foo(unsigned char\* pArray, int value) {  
 pArray[0] += 2;  
 pArray[value] += 2;  
 value++;  
 pArray[value] += 2;  
}  
int main(void) {  
 unsigned char array[10];  
 int value = 6;  
 memset(array, 1, sizeof(array));  
 foo(array, value);  
 printf("%d %d %d", array[0], array[6], value);  
 return 0;  
}

Výše uvedený program:

- A Nelze přeložit
- B Žádná z odpovědí není správná
- C Vypíše hodnoty '3 3 6'
- D Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- E Vypíše hodnoty '3 3 7'
- F Vypíše hodnoty '1 1 6'

**7**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- B** vytvoří pole o celkovém počtu 49 prvků typu int
- C** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- D** způsobí zápis za konec přidělené paměti
- E** vytvoří pole o celkovém počtu 64 prvků typu int
- F** žádná z odpovědí není správná

**10**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** nezpůsobí žádný memory leak
- C** vypíše Hello
- D** memory leak o velikosti 10 bajtů
- E** vypíše World

**8**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** žádná z odpovědí není správná
- B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** memory leak o velikosti 60 bajtů
- E** pád programu
- F** memory leak o velikosti 120 bajtů

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 51           |

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni je typicky časově náročnější
  - C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - D** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - F** Typová konverze na bitové úrovni je typicky časově náročnější

**2**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** způsobí zápis za konec přidělené paměti
- B** vytvoří pole o celkovém počtu 64 prvků typu int
- C** vytvoří pole o celkovém počtu 49 prvků typu int
- D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F** žádná z odpovědí není správná

**3**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** nelze přeložit
- B** 3.950000
- C** 0.000000
- D** 0.050000
- E** žádná z odpovědí není správná

- 4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**5**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** vypíše World
- C** vypíše Hello
- D** nezpůsobí žádný memory leak
- E** memory leak o velikosti 10 bajtů

**6**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '3 3 6'
- B** Žádná z odpovědí není správná
- C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D** Nelze přeložit
- E** Vypíše hodnoty '3 3 7'
- F** Vypíše hodnoty '1 1 6'

```

7 #include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}

```

Výše uvedený program:

- A Vypíše řetězec "Helord"
- B Vypíše řetězec "Helrld"
- C Nelze přeložit
- D Vypíše řetězec "Helloworld"
- E Vypíše řetězec "Hellrld"
- F Vypíše řetězec "Hell"

```

8 #include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}

```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A žádná z odpovědí není správná
- B foo(2);foo(3);foo(10);foo(7);
- C foo(1);foo(4);foo(6);foo(10);foo(8);
- D foo(5);foo(2);foo(5);foo(10);foo(7);
- E foo(2);foo(0);foo(5);foo(7);foo(8);

```

9 #include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}

```

Výše uvedený program vypíše:

- A Žádná z odpovědí není správná
- B Hello Hello Hello Hello
- C World Hello World Hello
- D Hello World Hello World
- E Nelze přeložit
- F World Hello Hello Hello

```

10 #include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}

```

Spuštění programu způsobí:

- A žádná z odpovědí není správná
- B pád programu
- C memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- D memory leak o velikosti 120 bajtů
- E memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F memory leak o velikosti 60 bajtů

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 52           |

- 1** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`  
Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":
- A** žádná z odpovědí není správná  
**B** `foo(1);foo(4);foo(6);foo(10);foo(8);`  
**C** `foo(2);foo(0);foo(5);foo(7);foo(8);`  
**D** `foo(2);foo(3);foo(10);foo(7);`  
**E** `foo(5);foo(2);foo(5);foo(10);foo(7);`

- 2** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** žádná z odpovědí není správná  
**B** způsobí zápis za konec přidělené paměti  
**C** vytvoří pole o celkovém počtu 49 prvků typu `int`  
**D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7  
**E** vyplní všechny položky pole hodnotou z intervalu 1 do 7  
**F** vytvoří pole o celkovém počtu 64 prvků typu `int`

- 3** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`  
Spuštění programu způsobí:
- A** memory leak o velikosti 120 bajtů  
**B** pád programu  
**C** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)  
**D** žádná z odpovědí není správná  
**E** memory leak o velikosti 60 bajtů  
**F** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)

- 4** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** Hello Hello Hello Hello  
**B** World Hello Hello Hello  
**C** Nelze přeložit  
**D** Hello World Hello World  
**E** World Hello World Hello  
**F** Žádná z odpovědí není správná

- 5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel  
**B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel  
**C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel  
**D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**6**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše řetězec "Helrld"
- C** Vypíše řetězec "Hellorld"
- D** Vypíše řetězec "Helord"
- E** Vypíše řetězec "Hell"
- F** Vypíše řetězec "Hellrld"

**7**

```
#include <stdio.h>
int main(){
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 3.950000
- B** 0.000000
- C** nelze přeložit
- D** 0.050000
- E** žádná z odpovědí není správná

**8**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** vypíše World
- C** nezpůsobí žádný memory leak
- D** žádná z odpovědí není správná
- E** vypíše Hello

**9**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Žádná z odpovědí není správná
- B** Vypíše hodnoty '3 3 6'
- C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D** Vypíše hodnoty '1 1 6'
- E** Nelze přeložit
- F** Vypíše hodnoty '3 3 7'

**10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni je typicky časově náročnější
- B** Typová konverze na bitové úrovni je typicky časově náročnější
- C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 53           |

**1**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše World
- B** memory leak o velikosti 10 bajtů
- C** vypíše Hello
- D** nezpůsobí žádný memory leak
- E** žádná z odpovědí není správná

**2**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** foo(2);foo(0);foo(5);foo(7);foo(8);
- C** foo(2);foo(3);foo(10);foo(7);
- D** foo(1);foo(4);foo(6);foo(10);foo(8);
- E** foo(5);foo(2);foo(5);foo(10);foo(7);

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na sémantické úrovni je typicky časově náročnější
- D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- F** Typová konverze na bitové úrovni je typicky časově náročnější

**4**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** World Hello World Hello
- B** Hello Hello Hello Hello
- C** Nelze přeložit
- D** Žádná z odpovědí není správná
- E** Hello World Hello World
- F** World Hello Hello Hello

**5**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** žádná z odpovědí není správná
- B** 0.050000
- C** nelze přeložit
- D** 3.950000
- E** 0.000000

**6**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Hellrld"
- D** Vypíše řetězec "Helrld"
- E** Vypíše řetězec "Helloworld"
- F** Nelze přeložit

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**8**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše hodnoty '3 3 7'
- C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D** Vypíše hodnoty '3 3 6'
- E** Vypíše hodnoty '1 1 6'
- F** Žádná z odpovědí není správná

**9**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** pád programu
- B** žádná z odpovědí není správná
- C** memory leak o velikosti 60 bajtů
- D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** memory leak o velikosti 120 bajtů
- F** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)

**10**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B** vytvoří pole o celkovém počtu 64 prvků typu int
- C** žádná z odpovědí není správná
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** způsobí zápis za konec přidělené paměti
- F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 54           |

- 1** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`  
Spuštění programu způsobí:
- A** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)  
**B** memory leak o velikosti 60 bajtů  
**C** žádná z odpovědí není správná  
**D** pád programu  
**E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)  
**F** memory leak o velikosti 120 bajtů

- 2** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**B** Vypíše hodnoty '3 3 6'  
**C** Žádná z odpovědí není správná  
**D** Vypíše hodnoty '3 3 7'  
**E** Vypíše hodnoty '1 1 6'  
**F** Nelze přeložit

- 3** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`  
Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(2);foo(0);foo(5);foo(7);foo(8);  
**B** foo(2);foo(3);foo(10);foo(7);  
**C** foo(5);foo(2);foo(5);foo(10);foo(7);  
**D** žádná z odpovědí není správná  
**E** foo(1);foo(4);foo(6);foo(10);foo(8);

- 4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**C** Typová konverze na bitové úrovni je typicky časově náročnější  
**D** Typová konverze na sémantické úrovni je typicky časově náročnější  
**E** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem  
**F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

- 5** `#include <stdio.h>`  
`int main() {`  
    `float a = (int) 3.95;`  
    `int x = a;`  
    `printf("%f", x - a);`  
    `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** žádná z odpovědí není správná  
**B** 0.000000  
**C** 0.050000  
**D** 3.950000  
**E** nelze přeložit

**6** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Hellrld"
- B** Vypíše řetězec "Hell"
- C** Vypíše řetězec "Helord"
- D** Vypíše řetězec "Hellorld"
- E** Nelze přeložit
- F** Vypíše řetězec "Helrld"

**7** `int main() {`  
 `int array[7][7];`  
 `for (int i = 1; i <= 7; i++) {`  
 `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
 `}`  
 `return 0;`  
`}`

Výše uvedený program:

- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B** vytvoří pole o celkovém počtu 64 prvků typu int
- C** žádná z odpovědí není správná
- D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- E** vytvoří pole o celkovém počtu 49 prvků typu int
- F** způsobí zápis za konec přidělené paměti

**8** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** World Hello Hello Hello
- B** Hello Hello Hello Hello
- C** Hello World Hello World
- D** Žádná z odpovědí není správná
- E** World Hello World Hello
- F** Nelze přeložit

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**10** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** vypíše World
- B** vypíše Hello
- C** žádná z odpovědí není správná
- D** memory leak o velikosti 10 bajtů
- E** nezpůsobí žádný memory leak

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 55           |

**1** `#include <stdio.h>`  
`int main() {`  
    `float a = (int) 3.95;`  
    `int x = a;`  
    `printf("%f", x - a);`  
    `return 0;`  
`}`  
Výše uvedený program vypíše:

- A** 0.000000
- B** 3.950000
- C** žádná z odpovědí není správná
- D** 0.050000
- E** nelze přeložit

**2** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`  
Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":

- A** `foo(5);foo(2);foo(5);foo(10);foo(7);`
- B** `foo(2);foo(0);foo(5);foo(7);foo(8);`
- C** `foo(1);foo(4);foo(6);foo(10);foo(8);`
- D** `foo(2);foo(3);foo(10);foo(7);`
- E** žádná z odpovědí není správná

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**4** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`  
Spuštění programu způsobí:

- A** žádná z odpovědí není správná
- B** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- C** pád programu
- D** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- E** memory leak o velikosti 120 bajtů
- F** memory leak o velikosti 60 bajtů

**5** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
    `*array = malloc(10);`  
    `array = NULL;`  
`}`  
`int main() {`  
    `int* array = NULL;`  
    `foo(&array);`  
    `if (array != NULL) printf("Hello");`  
    `else printf("World");`  
    `free(array);`  
    `return 0;`  
`}`  
Výše uvedený program:

- A** vypíše Hello
- B** žádná z odpovědí není správná
- C** nezpůsobí žádný memory leak
- D** vypíše World
- E** memory leak o velikosti 10 bajtů

**6**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hellorld"
- B** Vypíše řetězec "Helrld"
- C** Vypíše řetězec "Hell"
- D** Vypíše řetězec "Hellrld"
- E** Nelze přeložit
- F** Vypíše řetězec "Helord"

**7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na bitové úrovni je typicky časově náročnější
- E** Typová konverze na sémantické úrovni je typicky časově náročnější
- F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

**8**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** World Hello World Hello
- B** Hello Hello Hello Hello
- C** World Hello Hello Hello
- D** Hello World Hello World
- E** Žádná z odpovědí není správná
- F** Nelze přeložit

**9**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- B** způsobí zápis za konec přidělené paměti
- C** vytvoří pole o celkovém počtu 64 prvků typu int
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F** žádná z odpovědí není správná

**10**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B** Nelze přeložit
- C** Vypíše hodnoty '3 3 7'
- D** Žádná z odpovědí není správná
- E** Vypíše hodnoty '3 3 6'
- F** Vypíše hodnoty '1 1 6'

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 56           |

**1**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B** vytvoří pole o celkovém počtu 64 prvků typu int
- C** způsobí zápis za konec přidělené paměti
- D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- E** vytvoří pole o celkovém počtu 49 prvků typu int
- F** žádná z odpovědí není správná

**2**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '3 3 7'
- B** Vypíše hodnoty '3 3 6'
- C** Vypíše hodnoty '1 1 6'
- D** Nelze přeložit
- E** Žádná z odpovědí není správná
- F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

**3**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Helord"
- B** Vypíše řetězec "Hell"
- C** Vypíše řetězec "Hellrld"
- D** Vypíše řetězec "Helrld"
- E** Nelze přeložit
- F** Vypíše řetězec "Helloworld"

**4**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše World
- B** vypíše Hello
- C** nezpůsobí žádný memory leak
- D** memory leak o velikosti 10 bajtů
- E** žádná z odpovědí není správná

**5**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Hello Hello Hello Hello
- B** World Hello Hello Hello
- C** World Hello World Hello
- D** Hello World Hello World
- E** Žádná z odpovědí není správná
- F** Nelze přeložit

**6**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** nelze přeložit
- B** 3.950000
- C** 0.050000
- D** žádná z odpovědí není správná
- E** 0.000000

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni je typicky časově náročnější
  - C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - E** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Typová konverze na bitové úrovni je typicky časově náročnější

- 8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

- 9** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`  
Spuštění programu způsobí:
- A** pád programu
  - B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - C** memory leak o velikosti 120 bajtů
  - D** memory leak o velikosti 60 bajtů
  - E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - F** žádná z odpovědí není správná

- 10** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`  
Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":
- A** `foo(1);foo(4);foo(6);foo(10);foo(8);`
  - B** žádná z odpovědí není správná
  - C** `foo(5);foo(2);foo(5);foo(10);foo(7);`
  - D** `foo(2);foo(3);foo(10);foo(7);`
  - E** `foo(2);foo(0);foo(5);foo(7);foo(8);`



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 57           |

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**2**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Hellrld"
- C** Vypíše řetězec "Helrld"
- D** Vypíše řetězec "Helord"
- E** Vypíše řetězec "Helloworld"
- F** Nelze přeložit

**3**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 64 prvků typu int
- B** způsobí zápis za konec přidělené paměti
- C** žádná z odpovědí není správná
- D** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E** vytvoří pole o celkovém počtu 49 prvků typu int
- F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

**4**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše hodnoty '3 3 7'
- C** Žádná z odpovědí není správná
- D** Vypíše hodnoty '1 1 6'
- E** Vypíše hodnoty '3 3 6'
- F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

- 5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - C** Typová konverze na bitové úrovni je typicky časově náročnější
  - D** Typová konverze na sémantické úrovni je typicky časově náročnější
  - E** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - F** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace

**6**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** vypíše World
- B** memory leak o velikosti 10 bajtů
- C** nezpůsobí žádný memory leak
- D** žádná z odpovědí není správná
- E** vypíše Hello

**7** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** Hello World Hello World
- C** World Hello World Hello
- D** World Hello Hello Hello
- E** Hello Hello Hello Hello
- F** Nelze přeložit

**8** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`

Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":

- A** `foo(5);foo(2);foo(5);foo(10);foo(7);`
- B** Žádná z odpovědí není správná
- C** `foo(1);foo(4);foo(6);foo(10);foo(8);`
- D** `foo(2);foo(3);foo(10);foo(7);`
- E** `foo(2);foo(0);foo(5);foo(7);foo(8);`

**9** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- B** memory leak o velikosti 120 bajtů
- C** Žádná z odpovědí není správná
- D** memory leak o velikosti 60 bajtů
- E** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- F** pád programu

**10** `#include <stdio.h>`  
`int main() {`  
 `float a = (int) 3.95;`  
 `int x = a;`  
 `printf("%f", x - a);`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** 0.000000
- B** Žádná z odpovědí není správná
- C** nelze přeložit
- D** 3.950000
- E** 0.050000

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 58           |

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - C** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - D** Typová konverze na sémantické úrovni je typicky časově náročnější
  - E** Typová konverze na bitové úrovni je typicky časově náročnější
  - F** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace

- 2**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:
- A** 3.950000
  - B** nelze přeložit
  - C** 0.000000
  - D** 0.050000
  - E** žádná z odpovědí není správná

- 3**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Žádná z odpovědí není správná
  - B** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
  - C** Vypíše hodnoty '3 3 7'
  - D** Nelze přeložit
  - E** Vypíše hodnoty '3 3 6'
  - F** Vypíše hodnoty '1 1 6'

- 4**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(1);foo(4);foo(6);foo(10);foo(8);
  - B** foo(2);foo(0);foo(5);foo(7);foo(8);
  - C** foo(5);foo(2);foo(5);foo(10);foo(7);
  - D** foo(2);foo(3);foo(10);foo(7);
  - E** žádná z odpovědí není správná

- 5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

- 6**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:
- A** Hello World Hello World
  - B** World Hello World Hello
  - C** Hello Hello Hello Hello
  - D** Nelze přeložit
  - E** Žádná z odpovědí není správná
  - F** World Hello Hello Hello

```

7 #include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}

```

Výše uvedený program:

- A Vypíše řetězec "Hell"
- B Vypíše řetězec "Hellrld"
- C Vypíše řetězec "Helord"
- D Vypíše řetězec "Hellorld"
- E Vypíše řetězec "Helrld"
- F Nelze přeložit

```

8 int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}

```

Výše uvedený program:

- A vytvoří pole o celkovém počtu 64 prvků typu int
- B způsobí zápis za konec přidělené paměti
- C vyplní všechny položky pole hodnotou z intervalu 1 do 7
- D vytvoří pole o celkovém počtu 49 prvků typu int
- E žádná z odpovědí není správná
- F vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

```

9 #include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}

```

Výše uvedený program:

- A žádná z odpovědí není správná
- B vypíše Hello
- C memory leak o velikosti 10 bajtů
- D nezpůsobí žádný memory leak
- E vypíše World

```

10 #include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}

```

Spuštění programu způsobí:

- A pád programu
- B memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- C žádná z odpovědí není správná
- D memory leak o velikosti 120 bajtů
- E memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F memory leak o velikosti 60 bajtů

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 59           |

**1**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A World Hello Hello Hello
- B Žádná z odpovědí není správná
- C Hello Hello Hello Hello
- D World Hello World Hello
- E Nelze přeložit
- F Hello World Hello World

**2**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A Vypíše řetězec "Hell"
- B Vypíše řetězec "Hellorld"
- C Nelze přeložit
- D Vypíše řetězec "Hellrld"
- E Vypíše řetězec "Helrld"
- F Vypíše řetězec "Helord"

**3**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A nelze přeložit
- B 3.950000
- C 0.050000
- D 0.000000
- E Žádná z odpovědí není správná

**4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na sémantické úrovni je typicky časově náročnější
- B Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- D Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F Typová konverze na bitové úrovni je typicky časově náročnější

**5**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(2);foo(3);foo(10);foo(7);
- B foo(2);foo(0);foo(5);foo(7);foo(8);
- C foo(5);foo(2);foo(5);foo(10);foo(7);
- D foo(1);foo(4);foo(6);foo(10);foo(8);
- E Žádná z odpovědí není správná

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**7** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`  
 Výše uvedený program:

- A** vypíše World
- B** vypíše Hello
- C** nezpůsobí žádný memory leak
- D** žádná z odpovědí není správná
- E** memory leak o velikosti 10 bajtů

**8** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`  
 Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše hodnoty '3 3 6'
- C** Vypíše hodnoty '3 3 7'
- D** Žádná z odpovědí není správná
- E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- F** Vypíše hodnoty '1 1 6'

**9** `int main() {`  
 `int array[7][7];`  
 `for (int i = 1; i <= 7; i++) {`  
 `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
 `}`  
 `return 0;`  
`}`  
 Výše uvedený program:

- A** žádná z odpovědí není správná
- B** vytvoří pole o celkovém počtu 64 prvků typu int
- C** způsobí zápis za konec přidělené paměti
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

**10** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`  
 Spuštění programu způsobí:

- A** memory leak o velikosti 60 bajtů
- B** pád programu
- C** žádná z odpovědí není správná
- D** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** memory leak o velikosti 120 bajtů

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 60           |

**1** `#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
 pArray[0] += 2;
 pArray[value] += 2;
 value++;
 pArray[value] += 2;
}
int main(void) {
 unsigned char array[10];
 int value = 6;
 memset(array, 1, sizeof(array));
 foo(array, value);
 printf("%d %d %d", array[0], array[6], value);
 return 0;
}`

Výše uvedený program:

- A** Vypíše hodnoty '3 3 6'
- B** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C** Žádná z odpovědí není správná
- D** Vypíše hodnoty '1 1 6'
- E** Nelze přeložit
- F** Vypíše hodnoty '3 3 7'

**2** `#include <stdlib.h>
int main() {
 int* pArray1 = NULL;
 int* pArray2 = NULL;
 int* pArray3 = NULL;
 pArray1 = malloc(60);
 pArray2 = pArray1;
 pArray1 += 8;
 pArray3 = malloc(60);
 free(pArray2);
 pArray2 = pArray3;
 free(pArray2);
 return 0;
}`

Spuštění programu způsobí:

- A** žádná z odpovědí není správná
- B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** pád programu
- D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** memory leak o velikosti 60 bajtů
- F** memory leak o velikosti 120 bajtů

**3** `int main() {
 int array[7][7];
 for (int i = 1; i <= 7; i++) {
 for (int j = 1; j <= 7; j++) array[i][j] = i;
 }
 return 0;
}`

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 64 prvků typu int
- B** vytvoří pole o celkovém počtu 49 prvků typu int
- C** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- D** způsobí zápis za konec přidělené paměti
- E** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F** žádná z odpovědí není správná

**4** `#include <stdio.h>
int main() {
 unsigned char value1 = 0x55;
 unsigned char value2 = 0xAA;
 if (value1 & value2) printf("Hello ");
 else printf("World ");
 if (value1 && value2) printf("Hello ");
 else printf("World ");
 if (value1 | value2) printf("Hello ");
 else printf("World ");
 if (value1 || value2) printf("Hello ");
 else printf("World ");
 return 0;
}`

Výše uvedený program vypíše:

- A** Nelze přeložit
- B** World Hello World Hello
- C** Hello Hello Hello Hello
- D** World Hello Hello Hello
- E** Hello World Hello World
- F** Žádná z odpovědí není správná

**5** `#include <stdio.h>
int main() {
 float a = (int) 3.95;
 int x = a;
 printf("%f", x - a);
 return 0;
}`

Výše uvedený program vypíše:

- A** 3.950000
- B** nelze přeložit
- C** žádná z odpovědí není správná
- D** 0.050000
- E** 0.000000

```

6 #include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}

```

Výše uvedený program:

- A memory leak o velikosti 10 bajtů
- B vypíše Hello
- C nezpůsobí žádný memory leak
- D vypíše World
- E žádná z odpovědí není správná

7 Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- D Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E Typová konverze na sémantické úrovni je typicky časově náročnější
- F Typová konverze na bitové úrovni je typicky časově náročnější

```

8 #include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}

```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(2);foo(0);foo(5);foo(7);foo(8);
- B foo(5);foo(2);foo(5);foo(10);foo(7);
- C žádná z odpovědí není správná
- D foo(2);foo(3);foo(10);foo(7);
- E foo(1);foo(4);foo(6);foo(10);foo(8);

```

9 #include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}

```

Výše uvedený program:

- A Vypíše řetězec "Hellrld"
- B Vypíše řetězec "Helloworld"
- C Vypíše řetězec "Hell"
- D Vypíše řetězec "Helord"
- E Nelze přeložit
- F Vypíše řetězec "Helrld"

10 Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- B Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 61           |

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - B** Typová konverze na sémantické úrovni je typicky časově náročnější
  - C** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - E** Typová konverze na bitové úrovni je typicky časově náročnější
  - F** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace

- 2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**3**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** World Hello World Hello
- B** World Hello Hello Hello
- C** Nelze přeložit
- D** Žádná z odpovědí není správná
- E** Hello World Hello World
- F** Hello Hello Hello Hello

**4**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** nelze přeložit
- B** 0.000000
- C** 0.050000
- D** 3.950000
- E** Žádná z odpovědí není správná

**5**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** vypíše Hello
- C** žádná z odpovědí není správná
- D** nezpůsobí žádný memory leak
- E** vypíše World

**6**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C** vytvoří pole o celkovém počtu 64 prvků typu int
- D** způsobí zápis za konec přidělené paměti
- E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F** vytvoří pole o celkovém počtu 49 prvků typu int

**7**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** pád programu
- B** memory leak o velikosti 120 bajtů
- C** memory leak o velikosti 60 bajtů
- D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** žádná z odpovědí není správná

**8** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B** Nelze přeložit
- C** Vypíše hodnoty '3 3 7'
- D** Žádná z odpovědí není správná
- E** Vypíše hodnoty '1 1 6'
- F** Vypíše hodnoty '3 3 6'

**9** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Hellrld"
- B** Nelze přeložit
- C** Vypíše řetězec "Hell"
- D** Vypíše řetězec "Hellorld"
- E** Vypíše řetězec "Helord"
- F** Vypíše řetězec "Helrld"

**10** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`  
Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** `foo(2);foo(0);foo(5);foo(7);foo(8);`
- B** `foo(5);foo(2);foo(5);foo(10);foo(7);`
- C** `foo(1);foo(4);foo(6);foo(10);foo(8);`
- D** `foo(2);foo(3);foo(10);foo(7);`
- E** Žádná z odpovědí není správná

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 62           |

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - D** Typová konverze na sémantické úrovni je typicky časově náročnější
  - E** Typová konverze na bitové úrovni je typicky časově náročnější
  - F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

- 2**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:
- A** 0.050000
  - B** nelze přeložit
  - C** 0.000000
  - D** žádná z odpovědí není správná
  - E** 3.950000

- 3**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše řetězec "Hellorld"
  - B** Nelze přeložit
  - C** Vypíše řetězec "Helrld"
  - D** Vypíše řetězec "Hellrld"
  - E** Vypíše řetězec "Hell"
  - F** Vypíše řetězec "Helord"

- 4**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** žádná z odpovědí není správná
  - B** foo(2);foo(0);foo(5);foo(7);foo(8);
  - C** foo(1);foo(4);foo(6);foo(10);foo(8);
  - D** foo(2);foo(3);foo(10);foo(7);
  - E** foo(5);foo(2);foo(5);foo(10);foo(7);

- 5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - B** žádná z odpovědí není správná
  - C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - D** způsobí zápis za konec přidělené paměti
  - E** vytvoří pole o celkovém počtu 64 prvků typu int
  - F** vytvoří pole o celkovém počtu 49 prvků typu int

- 6**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:
- A** žádná z odpovědí není správná
  - B** nezpůsobí žádný memory leak
  - C** memory leak o velikosti 10 bajtů
  - D** vypíše World
  - E** vypíše Hello

**7**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Hello Hello Hello Hello
- B** World Hello World Hello
- C** World Hello Hello Hello
- D** Nelze přeložit
- E** Žádná z odpovědí není správná
- F** Hello World Hello World

**10**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** žádná z odpovědí není správná
- C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** memory leak o velikosti 60 bajtů
- E** memory leak o velikosti 120 bajtů
- F** pád programu

**8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

**9**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '3 3 7'
- B** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C** Nelze přeložit
- D** Žádná z odpovědí není správná
- E** Vypíše hodnoty '1 1 6'
- F** Vypíše hodnoty '3 3 6'

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 63           |

- 1**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

  
Výše uvedený program:
- A** žádná z odpovědí není správná
  - B** způsobí zápis za konec přidělené paměti
  - C** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - E** vytvoří pole o celkovém počtu 64 prvků typu int
  - F** vytvoří pole o celkovém počtu 49 prvků typu int

- 2**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

  
Výše uvedený program:
- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
  - B** Vypíše hodnoty '3 3 7'
  - C** Vypíše hodnoty '3 3 6'
  - D** Žádná z odpovědí není správná
  - E** Vypíše hodnoty '1 1 6'
  - F** Nelze přeložit

- 3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni je typicky časově náročnější
  - C** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - D** Typová konverze na bitové úrovni je typicky časově náročnější
  - E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

- 4**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

  
Výše uvedený program vypíše:
- A** 0.000000
  - B** nelze přeložit
  - C** 0.050000
  - D** žádná z odpovědí není správná
  - E** 3.950000

- 5**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

  
Výše uvedený program:
- A** Vypíše řetězec "Hellrld"
  - B** Vypíše řetězec "Helord"
  - C** Vypíše řetězec "Hellorld"
  - D** Nelze přeložit
  - E** Vypíše řetězec "Helrld"
  - F** Vypíše řetězec "Hell"

- 6**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

  
Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** žádná z odpovědí není správná
  - B** foo(2);foo(3);foo(10);foo(7);
  - C** foo(2);foo(0);foo(5);foo(7);foo(8);
  - D** foo(5);foo(2);foo(5);foo(10);foo(7);
  - E** foo(1);foo(4);foo(6);foo(10);foo(8);

**7** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** World Hello Hello Hello
- B** World Hello World Hello
- C** Hello World Hello World
- D** Nelze přeložit
- E** Žádná z odpovědí není správná
- F** Hello Hello Hello Hello

**8** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** memory leak o velikosti 10 bajtů
- C** vypíše Hello
- D** nezpůsobí žádný memory leak
- E** vypíše World

- 9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**10** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** memory leak o velikosti 120 bajtů
- C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** pád programu
- E** žádná z odpovědí není správná
- F** memory leak o velikosti 60 bajtů

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 64           |

**1**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A World Hello World Hello
- B World Hello Hello Hello
- C Nelze přeložit
- D Hello World Hello World
- E Žádná z odpovědí není správná
- F Hello Hello Hello Hello

**2**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A způsobí zápis za konec přidělené paměti
- B vytvoří pole o celkovém počtu 64 prvků typu int
- C vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D vytvoří pole o celkovém počtu 49 prvků typu int
- E Žádná z odpovědí není správná
- F vyplní všechny položky pole hodnotou z intervalu 1 do 7

**3**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A vypíše World
- B vypíše Hello
- C Žádná z odpovědí není správná
- D nezpůsobí žádný memory leak
- E memory leak o velikosti 10 bajtů

**4**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A 0.050000
- B Žádná z odpovědí není správná
- C 3.950000
- D nelze přeložit
- E 0.000000

**5**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A Žádná z odpovědí není správná
- B foo(1);foo(4);foo(6);foo(10);foo(8);
- C foo(2);foo(3);foo(10);foo(7);
- D foo(5);foo(2);foo(5);foo(10);foo(7);
- E foo(2);foo(0);foo(5);foo(7);foo(8);

**6**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A Vypíše hodnoty '1 1 6'
- B Vypíše hodnoty '3 3 6'
- C Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D Nelze přeložit
- E Vypíše hodnoty '3 3 7'
- F Žádná z odpovědí není správná

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - D** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Typová konverze na sémantické úrovni je typicky časově náročnější
  - F** Typová konverze na bitové úrovni je typicky časově náročnější

- 8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**9** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Hell"
- D** Vypíše řetězec "Hellorld"
- E** Vypíše řetězec "Helrld"
- F** Vypíše řetězec "Hellrld"

**10** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`

Spuštění programu způsobí:

- A** pád programu
- B** memory leak o velikosti 120 bajtů
- C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** memory leak o velikosti 60 bajtů
- E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** žádná z odpovědí není správná



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 65           |

**1**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A 0.050000
- B žádná z odpovědí není správná
- C 0.000000
- D 3.950000
- E nelze přeložit

**2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- C Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D Typová konverze na sémantické úrovni je typicky časově náročnější
- E Typová konverze na bitové úrovni je typicky časově náročnější
- F Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

**3**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A Vypíše řetězec "Helrld"
- B Vypíše řetězec "Hell"
- C Nelze přeložit
- D Vypíše řetězec "Hellorld"
- E Vypíše řetězec "Helrld"
- F Vypíše řetězec "Helord"

**4**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A vypíše World
- B nezpůsobí žádný memory leak
- C vypíše Hello
- D žádná z odpovědí není správná
- E memory leak o velikosti 10 bajtů

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**6**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Nelze přeložit
- B Žádná z odpovědí není správná
- C Hello Hello Hello Hello
- D Hello World Hello World
- E World Hello World Hello
- F World Hello Hello Hello

**7** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`

Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":

- A** `foo(5);foo(2);foo(5);foo(10);foo(7);`
- B** `foo(2);foo(0);foo(5);foo(7);foo(8);`
- C** `foo(1);foo(4);foo(6);foo(10);foo(8);`
- D** `foo(2);foo(3);foo(10);foo(7);`
- E** žádná z odpovědí není správná

**8** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- B** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- C** memory leak o velikosti 60 bajtů
- D** memory leak o velikosti 120 bajtů
- E** žádná z odpovědí není správná
- F** pád programu

**9** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d",array[0],array[6],value);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše hodnoty '1 1 6'
- C** Vypíše hodnoty '3 3 7'
- D** Vypíše hodnoty '3 3 6'
- E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- F** Žádná z odpovědí není správná

**10** `int main() {`  
 `int array[7][7];`  
 `for (int i = 1; i <= 7; i++) {`  
 `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
 `}`  
 `return 0;`  
`}`

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 49 prvků typu `int`
- B** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C** způsobí zápis za konec přidělené paměti
- D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- E** žádná z odpovědí není správná
- F** vytvoří pole o celkovém počtu 64 prvků typu `int`

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 66           |

- 1**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Žádná z odpovědí není správná  
**B** Vypíše hodnoty '3 3 6'  
**C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**D** Vypíše hodnoty '1 1 6'  
**E** Nelze přeložit  
**F** Vypíše hodnoty '3 3 7'

- 2**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:
- A** 3.950000  
**B** žádná z odpovědí není správná  
**C** 0.000000  
**D** nelze přeložit  
**E** 0.050000

- 3**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** žádná z odpovědí není správná  
**B** foo(5);foo(2);foo(5);foo(10);foo(7);  
**C** foo(2);foo(0);foo(5);foo(7);foo(8);  
**D** foo(2);foo(3);foo(10);foo(7);  
**E** foo(1);foo(4);foo(6);foo(10);foo(8);

- 4**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:
- A** Hello Hello Hello Hello  
**B** Žádná z odpovědí není správná  
**C** Hello World Hello World  
**D** Nelze přeložit  
**E** World Hello Hello Hello  
**F** World Hello World Hello

- 5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**C** Typová konverze na bitové úrovni je typicky časově náročnější  
**D** Typová konverze na sémantické úrovni je typicky časově náročnější  
**E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ  
**F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

- 6**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše řetězec "Helrld"  
**B** Nelze přeložit  
**C** Vypíše řetězec "Hellorld"  
**D** Vypíše řetězec "Helord"  
**E** Vypíše řetězec "Hellrld"  
**F** Vypíše řetězec "Hell"

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**8**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 64 prvků typu int
- B** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C** žádná z odpovědí není správná
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F** způsobí zápis za konec přidělené paměti

**9**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** memory leak o velikosti 10 bajtů
- C** nezpůsobí žádný memory leak
- D** vypíše Hello
- E** vypíše World

**10**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** žádná z odpovědí není správná
- B** pád programu
- C** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- D** memory leak o velikosti 120 bajtů
- E** memory leak o velikosti 60 bajtů
- F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 67           |

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - C** Typová konverze na sémantické úrovni je typicky časově náročnější
  - D** Typová konverze na bitové úrovni je typicky časově náročnější
  - E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

- 2**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:
- A** 3.950000
  - B** nelze přeložit
  - C** žádná z odpovědí není správná
  - D** 0.050000
  - E** 0.000000

- 3**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše řetězec "Helrld"
  - B** Nelze přeložit
  - C** Vypíše řetězec "Hellorld"
  - D** Vypíše řetězec "Helord"
  - E** Vypíše řetězec "Hell"
  - F** Vypíše řetězec "Hellrld"

- 4**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(5);foo(2);foo(5);foo(10);foo(7);
  - B** foo(1);foo(4);foo(6);foo(10);foo(8);
  - C** žádná z odpovědí není správná
  - D** foo(2);foo(3);foo(10);foo(7);
  - E** foo(2);foo(0);foo(5);foo(7);foo(8);

- 5**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:
- A** memory leak o velikosti 60 bajtů
  - B** žádná z odpovědí není správná
  - C** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - D** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - E** memory leak o velikosti 120 bajtů
  - F** pád programu

**6** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** World Hello World Hello
- B** Nelze přeložit
- C** World Hello Hello Hello
- D** Hello Hello Hello Hello
- E** Žádná z odpovědí není správná
- F** Hello World Hello World

**7** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B** Vypíše hodnoty '3 3 7'
- C** Vypíše hodnoty '3 3 6'
- D** Žádná z odpovědí není správná
- E** Nelze přeložit
- F** Vypíše hodnoty '1 1 6'

**8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**9** `int main() {`  
 `int array[7][7];`  
 `for (int i = 1; i <= 7; i++) {`  
 `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
 `}`  
 `return 0;`  
`}`

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 49 prvků typu int
- B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C** vytvoří pole o celkovém počtu 64 prvků typu int
- D** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E** způsobí zápis za konec přidělené paměti
- F** žádná z odpovědí není správná

**10** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** vypíše World
- B** vypíše Hello
- C** nezpůsobí žádný memory leak
- D** memory leak o velikosti 10 bajtů
- E** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 68           |

**1** #include <stdio.h>

```
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A nelze přeložit
- B žádná z odpovědí není správná
- C 3.950000
- D 0.000000
- E 0.050000

**2** int main() {  
int array[7][7];  
for (int i = 1; i <= 7; i++) {  
for (int j = 1; j <= 7; j++) array[i][j] = i;  
}  
return 0;  
}

Výše uvedený program:

- A vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B způsobí zápis za konec přidělené paměti
- C žádná z odpovědí není správná
- D vytvoří pole o celkovém počtu 64 prvků typu int
- E vytvoří pole o celkovém počtu 49 prvků typu int
- F vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B Typová konverze na sémantické úrovni je typicky časově náročnější
- C Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D Typová konverze na bitové úrovni je typicky časově náročnější
- E Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- F Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

**4** #include <stdlib.h>

```
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B memory leak o velikosti 120 bajtů
- C žádná z odpovědí není správná
- D memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E memory leak o velikosti 60 bajtů
- F pád programu

**5** #include <stdio.h>

```
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A World Hello World Hello
- B Hello World Hello World
- C Žádná z odpovědí není správná
- D Nelze přeložit
- E World Hello Hello Hello
- F Hello Hello Hello Hello

**6** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** memory leak o velikosti 10 bajtů
- C** žádná z odpovědí není správná
- D** vypíše Hello
- E** vypíše World

**7** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše hodnoty '1 1 6'
- B** Žádná z odpovědí není správná
- C** Vypíše hodnoty '3 3 7'
- D** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- E** Vypíše hodnoty '3 3 6'
- F** Nelze přeložit

**8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**9** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`  
 Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(5);foo(2);foo(5);foo(10);foo(7);
- B** žádná z odpovědí není správná
- C** foo(2);foo(3);foo(10);foo(7);
- D** foo(1);foo(4);foo(6);foo(10);foo(8);
- E** foo(2);foo(0);foo(5);foo(7);foo(8);

**10** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`  
 Výše uvedený program:

- A** Vypíše řetězec "Hellrld"
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Helrld"
- D** Nelze přeložit
- E** Vypíše řetězec "Hell"
- F** Vypíše řetězec "Hellorld"



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 69           |

- 1**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(5);foo(2);foo(5);foo(10);foo(7);
  - B** foo(1);foo(4);foo(6);foo(10);foo(8);
  - C** žádná z odpovědí není správná
  - D** foo(2);foo(0);foo(5);foo(7);foo(8);
  - E** foo(2);foo(3);foo(10);foo(7);

- 2**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše řetězec "Helord"
  - B** Vypíše řetězec "Hellrld"
  - C** Vypíše řetězec "Helloworld"
  - D** Nelze přeložit
  - E** Vypíše řetězec "Hell"
  - F** Vypíše řetězec "Helrld"

- 3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - C** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - D** Typová konverze na sémantické úrovni je typicky časově náročnější
  - E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - F** Typová konverze na bitové úrovni je typicky časově náročnější

- 4**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Nelze přeložit
  - B** Žádná z odpovědí není správná
  - C** Vypíše hodnoty '1 1 6'
  - D** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
  - E** Vypíše hodnoty '3 3 6'
  - F** Vypíše hodnoty '3 3 7'

- 5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** žádná z odpovědí není správná
  - B** vytvoří pole o celkovém počtu 49 prvků typu int
  - C** způsobí zápis za konec přidělené paměti
  - D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - F** vytvoří pole o celkovém počtu 64 prvků typu int

- 6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

```

7 #include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}

```

Výše uvedený program:

- A vypíše World
- B memory leak o velikosti 10 bajtů
- C žádná z odpovědí není správná
- D vypíše Hello
- E nezpůsobí žádný memory leak

```

8 #include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}

```

Spuštění programu způsobí:

- A memory leak o velikosti 60 bajtů
- B pád programu
- C žádná z odpovědí není správná
- D memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F memory leak o velikosti 120 bajtů

```

9 #include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}

```

Výše uvedený program vypíše:

- A Žádná z odpovědí není správná
- B World Hello World Hello
- C World Hello Hello Hello
- D Hello World Hello World
- E Hello Hello Hello Hello
- F Nelze přeložit

```

10 #include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}

```

Výše uvedený program vypíše:

- A žádná z odpovědí není správná
- B nelze přeložit
- C 3.950000
- D 0.050000
- E 0.000000

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 70           |

**1**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '1 1 6'
- B** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C** Nelze přeložit
- D** Žádná z odpovědí není správná
- E** Vypíše hodnoty '3 3 7'
- F** Vypíše hodnoty '3 3 6'

**2**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- B** způsobí zápis za konec přidělené paměti
- C** žádná z odpovědí není správná
- D** vytvoří pole o celkovém počtu 64 prvků typu int
- E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- F** vytvoří pole o celkovém počtu 49 prvků typu int

**3**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 60 bajtů
- B** memory leak o velikosti 120 bajtů
- C** pád programu
- D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** žádná z odpovědí není správná
- F** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)

**4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**5**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** World Hello World Hello
- C** Nelze přeložit
- D** World Hello Hello Hello
- E** Hello Hello Hello Hello
- F** Hello World Hello World

- 6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - E** Typová konverze na bitové úrovni je typicky časově náročnější
  - F** Typová konverze na sémantické úrovni je typicky časově náročnější

- 7**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

 Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(2);foo(3);foo(10);foo(7);
  - B** foo(2);foo(0);foo(5);foo(7);foo(8);
  - C** foo(5);foo(2);foo(5);foo(10);foo(7);
  - D** foo(1);foo(4);foo(6);foo(10);foo(8);
  - E** žádná z odpovědí není správná

- 8**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

 Výše uvedený program:
- A** Vypíše řetězec "Hell"
  - B** Nelze přeložit
  - C** Vypíše řetězec "Hellorld"
  - D** Vypíše řetězec "HeLrld"
  - E** Vypíše řetězec "Hellrld"
  - F** Vypíše řetězec "HeLord"

- 9**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

 Výše uvedený program:
- A** nezpůsobí žádný memory leak
  - B** memory leak o velikosti 10 bajtů
  - C** vypíše Hello
  - D** vypíše World
  - E** žádná z odpovědí není správná

- 10**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

 Výše uvedený program vypíše:
- A** žádná z odpovědí není správná
  - B** 0.000000
  - C** 3.950000
  - D** 0.050000
  - E** nelze přeložit

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 71           |

**1** `#include <stdio.h>
#include <string.h>
int main() {
 char str1[20];
 strcpy(str1, "Hello world");
 str1[4] = 0;
 printf("%s", str1);
 char* str2 = str1 + 8;
 printf("%s", str2);
 return 0;
}`

Výše uvedený program:

- A** Vypíše řetězec "Helord"
- B** Vypíše řetězec "Helrld"
- C** Vypíše řetězec "Hell"
- D** Nelze přeložit
- E** Vypíše řetězec "Hellorld"
- F** Vypíše řetězec "Hellrld"

**2** `#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
 *array = malloc(10);
 array = NULL;
}
int main() {
 int* array = NULL;
 foo(&array);
 if (array != NULL) printf("Hello");
 else printf("World");
 free(array);
 return 0;
}`

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** memory leak o velikosti 10 bajtů
- C** žádná z odpovědí není správná
- D** vypíše World
- E** vypíše Hello

**3** `#include <stdio.h>
void foo(int a) {
 switch (a) {
 case 0: break;
 case 1: printf("Fr"); break;
 case 2: printf("F");
 case 3: printf("re"); break;
 case 4: printf("e");
 case 5:
 case 6: break;
 case 7:
 case 8: printf("m"); break;
 default: printf("edo");
 }
}`

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** foo(2);foo(0);foo(5);foo(7);foo(8);
- C** foo(1);foo(4);foo(6);foo(10);foo(8);
- D** foo(5);foo(2);foo(5);foo(10);foo(7);
- E** foo(2);foo(3);foo(10);foo(7);

**4** `#include <stdlib.h>
int main() {
 int* pArray1 = NULL;
 int* pArray2 = NULL;
 int* pArray3 = NULL;
 pArray1 = malloc(60);
 pArray2 = pArray1;
 pArray1 += 8;
 pArray3 = malloc(60);
 free(pArray2);
 pArray2 = pArray3;
 free(pArray2);
 return 0;
}`

Spuštění programu způsobí:

- A** memory leak o velikosti 60 bajtů
- B** žádná z odpovědí není správná
- C** memory leak o velikosti 120 bajtů
- D** pád programu
- E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni je typicky časově náročnější
- B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E** Typová konverze na bitové úrovni je typicky časově náročnější
- F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

**6** `#include <stdio.h>
int main() {
 float a = (int) 3.95;
 int x = a;
 printf("%f", x - a);
 return 0;
}`

Výše uvedený program vypíše:

- A** 0.050000
- B** 0.000000
- C** nelze přeložit
- D** 3.950000
- E** žádná z odpovědí není správná

**7** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** World Hello Hello Hello
- B** Nelze přeložit
- C** Hello Hello Hello Hello
- D** Žádná z odpovědí není správná
- E** Hello World Hello World
- F** World Hello World Hello

**8** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B** Vypíše hodnoty '3 3 6'
- C** Vypíše hodnoty '3 3 7'
- D** Vypíše hodnoty '1 1 6'
- E** Žádná z odpovědí není správná
- F** Nelze přeložit

**9** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 49 prvků typu int
- B** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C** způsobí zápis za konec přidělené paměti
- D** vytvoří pole o celkovém počtu 64 prvků typu int
- E** žádná z odpovědí není správná
- F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

**10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 72           |

- 1**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše hodnoty '1 1 6'  
**B** Nelze přeložit  
**C** Žádná z odpovědí není správná  
**D** Vypíše hodnoty '3 3 6'  
**E** Vypíše hodnoty '3 3 7'  
**F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

- 2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind  
**B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel  
**C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel  
**D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

- 3**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7  
**B** způsobí zápis za konec přidělené paměti  
**C** vytvoří pole o celkovém počtu 64 prvků typu int  
**D** vytvoří pole o celkovém počtu 49 prvků typu int  
**E** vyplní všechny položky pole hodnotou z intervalu 1 do 7  
**F** žádná z odpovědí není správná

- 4**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:
- A** World Hello Hello Hello  
**B** World Hello World Hello  
**C** Nelze přeložit  
**D** Hello Hello Hello Hello  
**E** Hello World Hello World  
**F** Žádná z odpovědí není správná

- 5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem  
**B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**C** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ  
**D** Typová konverze na sémantické úrovni je typicky časově náročnější  
**E** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**F** Typová konverze na bitové úrovni je typicky časově náročnější

- 6**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(2);foo(3);foo(10);foo(7);  
**B** žádná z odpovědí není správná  
**C** foo(2);foo(0);foo(5);foo(7);foo(8);  
**D** foo(1);foo(4);foo(6);foo(10);foo(8);  
**E** foo(5);foo(2);foo(5);foo(10);foo(7);

```
7 #include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A Nelze přeložit
- B Vypíše řetězec "Hellrld"
- C Vypíše řetězec "Hell"
- D Vypíše řetězec "Helrld"
- E Vypíše řetězec "Hellorld"
- F Vypíše řetězec "Helord"

```
10 #include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A nezpůsobí žádný memory leak
- B žádná z odpovědí není správná
- C memory leak o velikosti 10 bajtů
- D vypíše Hello
- E vypíše World

```
8 #include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A nelze přeložit
- B 3.950000
- C 0.000000
- D žádná z odpovědí není správná
- E 0.050000

```
9 #include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A pád programu
- B memory leak o velikosti 120 bajtů
- C žádná z odpovědí není správná
- D memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F memory leak o velikosti 60 bajtů



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 73           |

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

- 2**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** žádná z odpovědí není správná
  - B** foo(1);foo(4);foo(6);foo(10);foo(8);
  - C** foo(2);foo(0);foo(5);foo(7);foo(8);
  - D** foo(2);foo(3);foo(10);foo(7);
  - E** foo(5);foo(2);foo(5);foo(10);foo(7);

- 3**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:
- A** žádná z odpovědí není správná
  - B** 3.950000
  - C** 0.000000
  - D** 0.050000
  - E** nelze přeložit

- 4**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:
- A** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
  - B** žádná z odpovědí není správná
  - C** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
  - D** pád programu
  - E** memory leak o velikosti 60 bajtů
  - F** memory leak o velikosti 120 bajtů

- 5**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:
- A** World Hello World Hello
  - B** Žádná z odpovědí není správná
  - C** Hello Hello Hello Hello
  - D** Nelze přeložit
  - E** Hello World Hello World
  - F** World Hello Hello Hello

- 6**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** vytvoří pole o celkovém počtu 64 prvků typu int
  - B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - C** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - D** vytvoří pole o celkovém počtu 49 prvků typu int
  - E** způsobí zápis za konec přidělené paměti
  - F** žádná z odpovědí není správná

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na bitové úrovni je typicky časově náročnější
  - C** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - D** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - E** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - F** Typová konverze na sémantické úrovni je typicky časově náročnější

**8**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše hodnoty '1 1 6'
- C** Vypíše hodnoty '3 3 7'
- D** Vypíše hodnoty '3 3 6'
- E** Žádná z odpovědí není správná
- F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

**9**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** vypíše Hello
- C** žádná z odpovědí není správná
- D** memory leak o velikosti 10 bajtů
- E** vypíše World

**10**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Helord"
- B** Vypíše řetězec "Hell"
- C** Vypíše řetězec "Hellorld"
- D** Vypíše řetězec "Hellrld"
- E** Vypíše řetězec "Helrld"
- F** Nelze přeložit

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 74           |

**1**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Helrld"
- D** Vypíše řetězec "Hell"
- E** Vypíše řetězec "Hellrld"
- F** Vypíše řetězec "Hellorld"

**2**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 60 bajtů
- B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** memory leak o velikosti 120 bajtů
- D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** žádná z odpovědí není správná
- F** pád programu

**3**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** Hello Hello Hello Hello
- C** World Hello Hello Hello
- D** Nelze přeložit
- E** Hello World Hello World
- F** World Hello World Hello

**4**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** žádná z odpovědí není správná
- C** memory leak o velikosti 10 bajtů
- D** vypíše World
- E** vypíše Hello

**5**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** foo(5);foo(2);foo(5);foo(10);foo(7);
- C** foo(2);foo(0);foo(5);foo(7);foo(8);
- D** foo(2);foo(3);foo(10);foo(7);
- E** foo(1);foo(4);foo(6);foo(10);foo(8);

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni je typicky časově náročnější
- B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- E** Typová konverze na bitové úrovni je typicky časově náročnější
- F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

**7**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B** vytvoří pole o celkovém počtu 49 prvků typu int
- C** vytvoří pole o celkovém počtu 64 prvků typu int
- D** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- E** žádná z odpovědí není správná
- F** způsobí zápis za konec přidělené paměti

**8**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 0.050000
- B** 3.950000
- C** nelze přeložit
- D** 0.000000
- E** žádná z odpovědí není správná

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**10**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '3 3 7'
- B** Vypíše hodnoty '1 1 6'
- C** Žádná z odpovědí není správná
- D** Vypíše hodnoty '3 3 6'
- E** Nelze přeložit
- F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 75           |

- 1**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(1);foo(4);foo(6);foo(10);foo(8);  
**B** žádná z odpovědí není správná  
**C** foo(5);foo(2);foo(5);foo(10);foo(7);  
**D** foo(2);foo(3);foo(10);foo(7);  
**E** foo(2);foo(0);foo(5);foo(7);foo(8);

- 2**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Vypíše hodnoty '1 1 6'  
**B** Nelze přeložit  
**C** Vypíše hodnoty '3 3 6'  
**D** Vypíše hodnoty '3 3 7'  
**E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**F** Žádná z odpovědí není správná

- 3**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:
- A** 0.000000  
**B** žádná z odpovědí není správná  
**C** 0.050000  
**D** nelze přeložit  
**E** 3.950000

- 4**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:
- A** World Hello World Hello  
**B** Hello Hello Hello Hello  
**C** World Hello Hello Hello  
**D** Hello World Hello World  
**E** Žádná z odpovědí není správná  
**F** Nelze přeložit

- 5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:
- A** způsobí zápis za konec přidělené paměti  
**B** žádná z odpovědí není správná  
**C** vytvoří pole o celkovém počtu 64 prvků typu int  
**D** vyplní všechny položky pole hodnotou z intervalu 1 do 7  
**E** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7  
**F** vytvoří pole o celkovém počtu 49 prvků typu int

- 6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**B** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem  
**C** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ  
**D** Typová konverze na sémantické úrovni je typicky časově náročnější  
**E** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**F** Typová konverze na bitové úrovni je typicky časově náročnější

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**8**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hellorld"
- B** Vypíše řetězec "Hellrld"
- C** Vypíše řetězec "Helord"
- D** Nelze přeložit
- E** Vypíše řetězec "Hell"
- F** Vypíše řetězec "Helrld"

**9**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** vypíše Hello
- C** nezpůsobí žádný memory leak
- D** memory leak o velikosti 10 bajtů
- E** vypíše World

**10**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** pád programu
- B** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** memory leak o velikosti 120 bajtů
- D** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** memory leak o velikosti 60 bajtů
- F** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 76           |

**1**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Nelze přeložit
- B** Hello World Hello World
- C** World Hello World Hello
- D** Hello Hello Hello Hello
- E** World Hello Hello Hello
- F** Žádná z odpovědí není správná

**2**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 120 bajtů
- B** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** žádná z odpovědí není správná
- D** pád programu
- E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** memory leak o velikosti 60 bajtů

**3**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(2);foo(0);foo(5);foo(7);foo(8);
- B** foo(5);foo(2);foo(5);foo(10);foo(7);
- C** žádná z odpovědí není správná
- D** foo(1);foo(4);foo(6);foo(10);foo(8);
- E** foo(2);foo(3);foo(10);foo(7);

**4**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Helord"
- C** Vypíše řetězec "Helrld"
- D** Vypíše řetězec "Helloworld"
- E** Nelze přeložit
- F** Vypíše řetězec "Hellrld"

**5**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** memory leak o velikosti 10 bajtů
- C** žádná z odpovědí není správná
- D** vypíše World
- E** vypíše Hello

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**7**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B** způsobí zápis za konec přidělené paměti
- C** žádná z odpovědí není správná
- D** vytvoří pole o celkovém počtu 64 prvků typu int
- E** vytvoří pole o celkovém počtu 49 prvků typu int
- F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

**8**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 3.950000
- B** nelze přeložit
- C** 0.050000
- D** 0.000000
- E** žádná z odpovědí není správná

**9**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše hodnoty '1 1 6'
- C** Žádná z odpovědí není správná
- D** Vypíše hodnoty '3 3 7'
- E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- F** Vypíše hodnoty '3 3 6'

**10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na bitové úrovni je typicky časově náročnější
- D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Typová konverze na sémantické úrovni je typicky časově náročnější



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 77           |

**1**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Hello Hello Hello Hello
- B Hello World Hello World
- C Nelze přeložit
- D Žádná z odpovědí není správná
- E World Hello World Hello
- F World Hello Hello Hello

**2**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A Vypíše hodnoty '3 3 6'
- B Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C Žádná z odpovědí není správná
- D Vypíše hodnoty '1 1 6'
- E Vypíše hodnoty '3 3 7'
- F Nelze přeložit

**3**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(5);foo(2);foo(5);foo(10);foo(7);
- B žádná z odpovědí není správná
- C foo(2);foo(0);foo(5);foo(7);foo(8);
- D foo(1);foo(4);foo(6);foo(10);foo(8);
- E foo(2);foo(3);foo(10);foo(7);

**4**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A žádná z odpovědí není správná
- B 3.950000
- C nelze přeložit
- D 0.050000
- E 0.000000

**5**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B memory leak o velikosti 60 bajtů
- C pád programu
- D žádná z odpovědí není správná
- E memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F memory leak o velikosti 120 bajtů

**6** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Hellrld"
- C** Vypíše řetězec "Helord"
- D** Vypíše řetězec "Helrld"
- E** Vypíše řetězec "Hellorld"
- F** Nelze přeložit

**7** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** vypíše Hello
- C** nezpůsobí žádný memory leak
- D** memory leak o velikosti 10 bajtů
- E** vypíše World

- 8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
  - B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**9** `int main() {`  
 `int array[7][7];`  
 `for (int i = 1; i <= 7; i++) {`  
 `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
 `}`  
 `return 0;`  
`}`

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 49 prvků typu int
- B** způsobí zápis za konec přidělené paměti
- C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- D** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- E** vytvoří pole o celkovém počtu 64 prvků typu int
- F** žádná z odpovědí není správná

- 10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- C** Typová konverze na bitové úrovni je typicky časově náročnější
- D** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- E** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Typová konverze na sémantické úrovni je typicky časově náročnější

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 78           |

- 1** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Nelze přeložit  
**B** Vypíše hodnoty '1 1 6'  
**C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**D** Vypíše hodnoty '3 3 6'  
**E** Žádná z odpovědí není správná  
**F** Vypíše hodnoty '3 3 7'

- 2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel  
**B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel  
**C** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel  
**D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

- 3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem  
**B** Typová konverze na sémantické úrovni je typicky časově náročnější  
**C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace  
**D** Typová konverze na bitové úrovni je typicky časově náročnější  
**E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ  
**F** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace

- 4** `#include <stdio.h>`  
`int main() {`  
    `float a = (int) 3.95;`  
    `int x = a;`  
    `printf("%f", x - a);`  
    `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** nelze přeložit  
**B** 0.000000  
**C** 0.050000  
**D** 3.950000  
**E** žádná z odpovědí není správná

- 5** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** žádná z odpovědí není správná  
**B** vytvoří pole o celkovém počtu 64 prvků typu int  
**C** vytvoří pole o celkovém počtu 49 prvků typu int  
**D** způsobí zápis za konec přidělené paměti  
**E** vyplní všechny položky pole hodnotou z intervalu 1 do 7  
**F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

- 6** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Vypíše řetězec "Helrld"  
**B** Vypíše řetězec "Hellerld"  
**C** Vypíše řetězec "Hellorld"  
**D** Vypíše řetězec "Hell"  
**E** Vypíše řetězec "Helord"  
**F** Nelze přeložit

**7** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** World Hello Hello Hello
- B** Žádná z odpovědí není správná
- C** Hello World Hello World
- D** World Hello World Hello
- E** Nelze přeložit
- F** Hello Hello Hello Hello

**8** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** vypíše Hello
- B** nezpůsobí žádný memory leak
- C** vypíše World
- D** žádná z odpovědí není správná
- E** memory leak o velikosti 10 bajtů

**9** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`

Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":

- A** `foo(2);foo(0);foo(5);foo(7);foo(8);`
- B** `foo(5);foo(2);foo(5);foo(10);foo(7);`
- C** `foo(1);foo(4);foo(6);foo(10);foo(8);`
- D** `foo(2);foo(3);foo(10);foo(7);`
- E** žádná z odpovědí není správná

**10** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- B** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- C** žádná z odpovědí není správná
- D** memory leak o velikosti 60 bajtů
- E** pád programu
- F** memory leak o velikosti 120 bajtů

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 79           |

**1**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hell"
- B** Vypíše řetězec "Hellorld"
- C** Vypíše řetězec "Hellrld"
- D** Nelze přeložit
- E** Vypíše řetězec "Helrld"
- F** Vypíše řetězec "Helord"

**2**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** pád programu
- C** memory leak o velikosti 60 bajtů
- D** memory leak o velikosti 120 bajtů
- E** žádná z odpovědí není správná
- F** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)

**3**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** World Hello World Hello
- B** Nelze přeložit
- C** World Hello Hello Hello
- D** Hello World Hello World
- E** Žádná z odpovědí není správná
- F** Hello Hello Hello Hello

**4**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- D** vytvoří pole o celkovém počtu 64 prvků typu int
- E** vytvoří pole o celkovém počtu 49 prvků typu int
- F** způsobí zápis za konec přidělené paměti

**5**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Žádná z odpovědí není správná
- B** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C** Vypíše hodnoty '1 1 6'
- D** Vypíše hodnoty '3 3 7'
- E** Nelze přeložit
- F** Vypíše hodnoty '3 3 6'

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- D** Typová konverze na sémantické úrovni je typicky časově náročnější
- E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Typová konverze na bitové úrovni je typicky časově náročnější

**7**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** nelze přeložit
- B** 0.050000
- C** žádná z odpovědí není správná
- D** 3.950000
- E** 0.000000

**8**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(5);foo(2);foo(5);foo(10);foo(7);
- B** foo(2);foo(0);foo(5);foo(7);foo(8);
- C** foo(2);foo(3);foo(10);foo(7);
- D** žádná z odpovědí není správná
- E** foo(1);foo(4);foo(6);foo(10);foo(8);

**9**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** vypíše World
- C** nezpůsobí žádný memory leak
- D** žádná z odpovědí není správná
- E** vypíše Hello

**10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 80           |

**1**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** nezpůsobí žádný memory leak
- C** vypíše World
- D** vypíše Hello
- E** memory leak o velikosti 10 bajtů

**2**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A** Hello World Hello World
- B** Žádná z odpovědí není správná
- C** Nelze přeložit
- D** World Hello World Hello
- E** World Hello Hello Hello
- F** Hello Hello Hello Hello

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**4**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše řetězec "Hell"
- C** Vypíše řetězec "Hellorld"
- D** Vypíše řetězec "Helord"
- E** Vypíše řetězec "Hellrld"
- F** Vypíše řetězec "Helrld"

**5**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** žádná z odpovědí není správná
- B** memory leak o velikosti 120 bajtů
- C** memory leak o velikosti 60 bajtů
- D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** pád programu

**6** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`

Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":

- A** `foo(1);foo(4);foo(6);foo(10);foo(8);`
- B** `foo(5);foo(2);foo(5);foo(10);foo(7);`
- C** `foo(2);foo(0);foo(5);foo(7);foo(8);`
- D** `foo(2);foo(3);foo(10);foo(7);`
- E** žádná z odpovědí není správná

**7** `int main() {`  
 `int array[7][7];`  
 `for (int i = 1; i <= 7; i++) {`  
 `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
 `}`  
 `return 0;`  
`}`

Výše uvedený program:

- A** způsobí zápis za konec přidělené paměti
- B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- D** vytvoří pole o celkovém počtu 49 prvků typu `int`
- E** žádná z odpovědí není správná
- F** vytvoří pole o celkovém počtu 64 prvků typu `int`

- 8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - C** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - E** Typová konverze na bitové úrovni je typicky časově náročnější
  - F** Typová konverze na sémantické úrovni je typicky časově náročnější

**9** `#include <stdio.h>`  
`int main() {`  
 `float a = (int) 3.95;`  
 `int x = a;`  
 `printf("%f", x - a);`  
 `return 0;`  
`}`  
 Výše uvedený program vypíše:

- A** 3.950000
- B** nelze přeložit
- C** 0.000000
- D** 0.050000
- E** žádná z odpovědí není správná

**10** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
 `pArray[0] += 2;`  
 `pArray[value] += 2;`  
 `value++;`  
 `pArray[value] += 2;`  
`}`  
`int main(void) {`  
 `unsigned char array[10];`  
 `int value = 6;`  
 `memset(array, 1, sizeof(array));`  
 `foo(array, value);`  
 `printf("%d %d %d", array[0], array[6], value);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Nelze přeložit
- B** Žádná z odpovědí není správná
- C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti `array`.
- D** Vypíše hodnoty '3 3 7'
- E** Vypíše hodnoty '3 3 6'
- F** Vypíše hodnoty '1 1 6'



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 81           |

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

- 2**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:
- A** nelze přeložit
  - B** 0.050000
  - C** žádná z odpovědí není správná
  - D** 0.000000
  - E** 3.950000

- 3**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:
- A** Žádná z odpovědí není správná
  - B** Vypíše hodnoty '3 3 7'
  - C** Vypíše hodnoty '1 1 6'
  - D** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
  - E** Nelze přeložit
  - F** Vypíše hodnoty '3 3 6'

- 4**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše řetězec "Hellrld"
- B** Vypíše řetězec "Hellorld"
- C** Vypíše řetězec "Hell"
- D** Vypíše řetězec "Helord"
- E** Nelze přeložit
- F** Vypíše řetězec "Helrld"

- 5**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":
- A** foo(2);foo(0);foo(5);foo(7);foo(8);
  - B** foo(5);foo(2);foo(5);foo(10);foo(7);
  - C** žádná z odpovědí není správná
  - D** foo(2);foo(3);foo(10);foo(7);
  - E** foo(1);foo(4);foo(6);foo(10);foo(8);

**6** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** pád programu
- B** memory leak o velikosti 120 bajtů
- C** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- D** memory leak o velikosti 60 bajtů
- E** žádná z odpovědí není správná
- F** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)

**7** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** vypíše Hello
- C** vypíše World
- D** memory leak o velikosti 10 bajtů
- E** žádná z odpovědí není správná

**8** `int main() {`  
 `int array[7][7];`  
 `for (int i = 1; i <= 7; i++) {`  
 `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
 `}`  
 `return 0;`  
`}`

Výše uvedený program:

- A** způsobí zápis za konec přidělené paměti
- B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C** vytvoří pole o celkovém počtu 49 prvků typu `int`
- D** vytvoří pole o celkovém počtu 64 prvků typu `int`
- E** žádná z odpovědí není správná
- F** vyplní všechny položky pole hodnotou z intervalu 1 do 7

**9** `#include <stdio.h>`  
`int main() {`  
 `unsigned char value1 = 0x55;`  
 `unsigned char value2 = 0xAA;`  
 `if (value1 & value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 && value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 | value2) printf("Hello ");`  
 `else printf("World ");`  
 `if (value1 || value2) printf("Hello ");`  
 `else printf("World ");`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Hello Hello Hello Hello
- B** Hello World Hello World
- C** Nelze přeložit
- D** World Hello Hello Hello
- E** World Hello World Hello
- F** Žádná z odpovědí není správná

**10** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na bitové úrovni je typicky časově náročnější
- B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- D** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- E** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Typová konverze na sémantické úrovni je typicky časově náročnější

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 82           |

- 1** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - C** Typová konverze na sémantické úrovni je typicky časově náročnější
  - D** Typová konverze na bitové úrovni je typicky časově náročnější
  - E** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
  - F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

- 2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
  - B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
  - C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
  - D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

- 3**
- ```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```
- Výše uvedený program:
- A** způsobí zápis za konec přidělené paměti
  - B** vytvoří pole o celkovém počtu 64 prvků typu int
  - C** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
  - D** žádná z odpovědí není správná
  - E** vyplní všechny položky pole hodnotou z intervalu 1 do 7
  - F** vytvoří pole o celkovém počtu 49 prvků typu int

- 4**
- ```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```
- Výše uvedený program vypíše:
- A** 3.950000
  - B** nelze přeložit
  - C** žádná z odpovědí není správná
  - D** 0.050000
  - E** 0.000000

- 5**
- ```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```
- Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** nezpůsobí žádný memory leak
- C** vypíše World
- D** žádná z odpovědí není správná
- E** vypíše Hello

- 6**
- ```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```
- Výše uvedený program:
- A** Vypíše řetězec "Hell"
  - B** Vypíše řetězec "Hellorld"
  - C** Nelze přeložit
  - D** Vypíše řetězec "Helrld"
  - E** Vypíše řetězec "Hellrld"
  - F** Vypíše řetězec "Helord"

```

7 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}

```

Výše uvedený program:

- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B** Žádná z odpovědí není správná
- C** Vypíše hodnoty '3 3 7'
- D** Nelze přeložit
- E** Vypíše hodnoty '1 1 6'
- F** Vypíše hodnoty '3 3 6'

```

8 #include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}

```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(5);foo(2);foo(5);foo(10);foo(7);
- B** žádná z odpovědí není správná
- C** foo(1);foo(4);foo(6);foo(10);foo(8);
- D** foo(2);foo(0);foo(5);foo(7);foo(8);
- E** foo(2);foo(3);foo(10);foo(7);

```

9 #include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}

```

Spuštění programu způsobí:

- A** memory leak o velikosti 60 bajtů
- B** žádná z odpovědí není správná
- C** memory leak o velikosti 120 bajtů
- D** pád programu
- E** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)

```

10 #include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}

```

Výše uvedený program vypíše:

- A** World Hello World Hello
- B** Hello World Hello World
- C** Žádná z odpovědí není správná
- D** Hello Hello Hello Hello
- E** Nelze přeložit
- F** World Hello Hello Hello

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 83           |

**1** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Nelze přeložit
- B** World Hello Hello Hello
- C** World Hello World Hello
- D** Žádná z odpovědí není správná
- E** Hello World Hello World
- F** Hello Hello Hello Hello

**2** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Hellrld"
- B** Vypíše řetězec "Hellorld"
- C** Vypíše řetězec "Helrld"
- D** Vypíše řetězec "Helord"
- E** Nelze přeložit
- F** Vypíše řetězec "Hell"

**3** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- B** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- C** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

**4** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
    `*array = malloc(10);`  
    `array = NULL;`  
`}`  
`int main() {`  
    `int* array = NULL;`  
    `foo(&array);`  
    `if (array != NULL) printf("Hello");`  
    `else printf("World");`  
    `free(array);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** nezpůsobí žádný memory leak
- B** vypíše World
- C** memory leak o velikosti 10 bajtů
- D** vypíše Hello
- E** žádná z odpovědí není správná

**5** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** pád programu
- C** memory leak o velikosti 120 bajtů
- D** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** memory leak o velikosti 60 bajtů
- F** žádná z odpovědí není správná

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na bitové úrovni je typicky časově náročnější
- B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E** Typová konverze na sémantické úrovni je typicky časově náročnější
- F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

```

7 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}

```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše hodnoty '3 3 7'
- C** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- D** Vypíše hodnoty '1 1 6'
- E** Žádná z odpovědí není správná
- F** Vypíše hodnoty '3 3 6'

```

8 #include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}

```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(2);foo(0);foo(5);foo(7);foo(8);
- B** foo(5);foo(2);foo(5);foo(10);foo(7);
- C** žádná z odpovědí není správná
- D** foo(1);foo(4);foo(6);foo(10);foo(8);
- E** foo(2);foo(3);foo(10);foo(7);

```

9 #include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}

```

Výše uvedený program vypíše:

- A** žádná z odpovědí není správná
- B** nelze přeložit
- C** 0.000000
- D** 0.050000
- E** 3.950000

```

10 int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}

```

Výše uvedený program:

- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B** žádná z odpovědí není správná
- C** vytvoří pole o celkovém počtu 49 prvků typu int
- D** způsobí zápis za konec přidělené paměti
- E** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- F** vytvoří pole o celkovém počtu 64 prvků typu int

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 84           |

**1**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A vypíše Hello
- B žádná z odpovědí není správná
- C nezpůsobí žádný memory leak
- D vypíše World
- E memory leak o velikosti 10 bajtů

**2**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A žádná z odpovědí není správná
- B 0.000000
- C 3.950000
- D nelze přeložit
- E 0.050000

**3**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- B memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C memory leak o velikosti 120 bajtů
- D pád programu
- E žádná z odpovědí není správná
- F memory leak o velikosti 60 bajtů

**4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- C Typová konverze na sémantické úrovni je typicky časově náročnější
- D Typová konverze na bitové úrovni je typicky časově náročnější
- E Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- F Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace

**5**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A World Hello Hello Hello
- B World Hello World Hello
- C Hello Hello Hello Hello
- D Hello World Hello World
- E Žádná z odpovědí není správná
- F Nelze přeložit

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- C Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- D Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel

```

7 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}

```

Výše uvedený program:

- A** Vypíše hodnoty '3 3 6'
- B** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- C** Žádná z odpovědí není správná
- D** Nelze přeložit
- E** Vypíše hodnoty '1 1 6'
- F** Vypíše hodnoty '3 3 7'

```

8 #include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}

```

Výše uvedený program:

- A** Vypíše řetězec "Hellrld"
- B** Nelze přeložit
- C** Vypíše řetězec "Helord"
- D** Vypíše řetězec "Hellorld"
- E** Vypíše řetězec "Hell"
- F** Vypíše řetězec "Helrld"

```

9 #include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}

```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** foo(1);foo(4);foo(6);foo(10);foo(8);
- C** foo(2);foo(3);foo(10);foo(7);
- D** foo(2);foo(0);foo(5);foo(7);foo(8);
- E** foo(5);foo(2);foo(5);foo(10);foo(7);

```

10 int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}

```

Výše uvedený program:

- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B** způsobí zápis za konec přidělené paměti
- C** vytvoří pole o celkovém počtu 64 prvků typu int
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** žádná z odpovědí není správná
- F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 85           |

**1**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B** Žádná z odpovědí není správná
- C** Vypíše hodnoty '1 1 6'
- D** Nelze přeložit
- E** Vypíše hodnoty '3 3 6'
- F** Vypíše hodnoty '3 3 7'

**2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D** Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**3**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** foo(2);foo(3);foo(10);foo(7);
- B** žádná z odpovědí není správná
- C** foo(1);foo(4);foo(6);foo(10);foo(8);
- D** foo(2);foo(0);foo(5);foo(7);foo(8);
- E** foo(5);foo(2);foo(5);foo(10);foo(7);

**4**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** vypíše Hello
- C** vypíše World
- D** nezpůsobí žádný memory leak
- E** memory leak o velikosti 10 bajtů

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na bitové úrovni je typicky časově náročnější
- B** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- F** Typová konverze na sémantické úrovni je typicky časově náročnější

**6**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 60 bajtů
- B** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** žádná z odpovědí není správná
- D** memory leak o velikosti 120 bajtů
- E** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- F** pád programu

```
7 #include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Hello Hello Hello Hello
- B World Hello Hello Hello
- C Žádná z odpovědí není správná
- D Nelze přeložit
- E Hello World Hello World
- F World Hello World Hello

```
8 #include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A Vypíše řetězec "Hellorld"
- B Vypíše řetězec "Hellrld"
- C Vypíše řetězec "Helord"
- D Nelze přeložit
- E Vypíše řetězec "Helrld"
- F Vypíše řetězec "Hell"

```
9 #include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A nelze přeložit
- B 0.050000
- C 0.000000
- D 3.950000
- E žádná z odpovědí není správná

```
10 int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A způsobí zápis za konec přidělené paměti
- B vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C vytvoří pole o celkovém počtu 64 prvků typu int
- D vytvoří pole o celkovém počtu 49 prvků typu int
- E žádná z odpovědí není správná
- F vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 86           |

- 1** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Vypíše hodnoty '3 3 6'  
**B** Vypíše hodnoty '3 3 7'  
**C** Nelze přeložit  
**D** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.  
**E** Vypíše hodnoty '1 1 6'  
**F** Žádná z odpovědí není správná

- 2** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`  
Výše uvedený program:
- A** Vypíše řetězec "Hell"  
**B** Vypíše řetězec "Helord"  
**C** Vypíše řetězec "Hellorld"  
**D** Nelze přeložit  
**E** Vypíše řetězec "Helrld"  
**F** Vypíše řetězec "Hellrld"

- 3** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** Žádná z odpovědí není správná  
**B** Nelze přeložit  
**C** World Hello World Hello  
**D** Hello World Hello World  
**E** World Hello Hello Hello  
**F** Hello Hello Hello Hello

- 4** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel  
**B** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel  
**C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind  
**D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

- 5** `#include <stdio.h>`  
`int main() {`  
    `float a = (int) 3.95;`  
    `int x = a;`  
    `printf("%f", x - a);`  
    `return 0;`  
`}`  
Výše uvedený program vypíše:
- A** nelze přeložit  
**B** žádná z odpovědí není správná  
**C** 0.000000  
**D** 3.950000  
**E** 0.050000

**6**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

 Výše uvedený program:

- A** žádná z odpovědí není správná
- B** vypíše Hello
- C** memory leak o velikosti 10 bajtů
- D** nezpůsobí žádný memory leak
- E** vypíše World

**7**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

 Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A** žádná z odpovědí není správná
- B** foo(1);foo(4);foo(6);foo(10);foo(8);
- C** foo(2);foo(3);foo(10);foo(7);
- D** foo(5);foo(2);foo(5);foo(10);foo(7);
- E** foo(2);foo(0);foo(5);foo(7);foo(8);

**8**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

 Výše uvedený program:

- A** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- B** žádná z odpovědí není správná
- C** vytvoří pole o celkovém počtu 64 prvků typu int
- D** vytvoří pole o celkovém počtu 49 prvků typu int
- E** způsobí zápis za konec přidělené paměti
- F** vyplní všechny položky pole hodnotou z intervalu 1 do 7

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- D** Typová konverze na sémantické úrovni je typicky časově náročnější
- E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- F** Typová konverze na bitové úrovni je typicky časově náročnější

**10**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** pád programu
- B** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- C** žádná z odpovědí není správná
- D** memory leak o velikosti 120 bajtů
- E** memory leak o velikosti 60 bajtů
- F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 87           |

**1** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Hello World Hello World
- B** Hello Hello Hello Hello
- C** Žádná z odpovědí není správná
- D** Nelze přeložit
- E** World Hello World Hello
- F** World Hello Hello Hello

**2** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
    `*array = malloc(10);`  
    `array = NULL;`  
`}`  
`int main() {`  
    `int* array = NULL;`  
    `foo(&array);`  
    `if (array != NULL) printf("Hello");`  
    `else printf("World");`  
    `free(array);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** vypíše Hello
- B** žádná z odpovědí není správná
- C** memory leak o velikosti 10 bajtů
- D** nezpůsobí žádný memory leak
- E** vypíše World

**3** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
    `char str1[20];`  
    `strcpy(str1, "Hello world");`  
    `str1[4] = 0;`  
    `printf("%s", str1);`  
    `char* str2 = str1 + 8;`  
    `printf("%s", str2);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Helord"
- B** Vypíše řetězec "Hell"
- C** Vypíše řetězec "Hellorld"
- D** Vypíše řetězec "HeLrld"
- E** Vypíše řetězec "Hellrld"
- F** Nelze přeložit

**4** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- B** Vypíše hodnoty '3 3 6'
- C** Nelze přeložit
- D** Vypíše hodnoty '3 3 7'
- E** Vypíše hodnoty '1 1 6'
- F** Žádná z odpovědí není správná

**5** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**6** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`

Výše uvedený program:

- A** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C** způsobí zápis za konec přidělené paměti
- D** vytvoří pole o celkovém počtu 64 prvků typu int
- E** žádná z odpovědí není správná
- F** vytvoří pole o celkovém počtu 49 prvků typu int

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - C** Typová konverze na bitové úrovni je typicky časově náročnější
  - D** Typová konverze na sémantické úrovni je typicky časově náročnější
  - E** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

- 8** `#include <stdio.h>`  
`int main() {`  
 `float a = (int) 3.95;`  
 `int x = a;`  
 `printf("%f", x - a);`  
 `return 0;`  
`}`  
 Výše uvedený program vypíše:
- A** 0.000000
  - B** nelze přeložit
  - C** 3.950000
  - D** žádná z odpovědí není správná
  - E** 0.050000

- 9** `#include <stdio.h>`  
`void foo(int a) {`  
 `switch (a) {`  
 `case 0: break;`  
 `case 1: printf("Fr"); break;`  
 `case 2: printf("F");`  
 `case 3: printf("re"); break;`  
 `case 4: printf("e");`  
 `case 5:`  
 `case 6: break;`  
 `case 7:`  
 `case 8: printf("m"); break;`  
 `default: printf("edo");`  
 `}`  
`}`  
 Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":
- A** `foo(5);foo(2);foo(5);foo(10);foo(7);`
  - B** `foo(1);foo(4);foo(6);foo(10);foo(8);`
  - C** `foo(2);foo(0);foo(5);foo(7);foo(8);`
  - D** žádná z odpovědí není správná
  - E** `foo(2);foo(3);foo(10);foo(7);`

- 10** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`  
 Spuštění programu způsobí:
- A** memory leak o velikosti 60 bajtů
  - B** pád programu
  - C** memory leak o velikosti 120 bajtů
  - D** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)
  - E** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)
  - F** žádná z odpovědí není správná

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 88           |

**1** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** World Hello Hello Hello
- B** Žádná z odpovědí není správná
- C** Nelze přeložit
- D** World Hello World Hello
- E** Hello Hello Hello Hello
- F** Hello World Hello World

**2** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`  
Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":

- A** `foo(1);foo(4);foo(6);foo(10);foo(8);`
- B** `foo(2);foo(3);foo(10);foo(7);`
- C** `foo(2);foo(0);foo(5);foo(7);foo(8);`
- D** žádná z odpovědí není správná
- E** `foo(5);foo(2);foo(5);foo(10);foo(7);`

**3** `#include <stdlib.h>`  
`int main() {`  
    `int* pArray1 = NULL;`  
    `int* pArray2 = NULL;`  
    `int* pArray3 = NULL;`  
    `pArray1 = malloc(60);`  
    `pArray2 = pArray1;`  
    `pArray1 += 8;`  
    `pArray3 = malloc(60);`  
    `free(pArray2);`  
    `pArray2 = pArray3;`  
    `free(pArray2);`  
    `return 0;`  
`}`  
Spuštění programu způsobí:

- A** memory leak o velikosti 480 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- B** memory leak o velikosti 240 bajtů (za předpokladu, že `int` zabírá 4 bajty)
- C** memory leak o velikosti 120 bajtů
- D** memory leak o velikosti 60 bajtů
- E** žádná z odpovědí není správná
- F** pád programu

**4** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
    `*array = malloc(10);`  
    `array = NULL;`  
`}`  
`int main() {`  
    `int* array = NULL;`  
    `foo(&array);`  
    `if (array != NULL) printf("Hello");`  
    `else printf("World");`  
    `free(array);`  
    `return 0;`  
`}`  
Výše uvedený program:

- A** memory leak o velikosti 10 bajtů
- B** vypíše Hello
- C** vypíše World
- D** nezpůsobí žádný memory leak
- E** žádná z odpovědí není správná

**5**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A** Nelze přeložit
- B** Vypíše řetězec "Hell"
- C** Vypíše řetězec "Hellrld"
- D** Vypíše řetězec "Helrld"
- E** Vypíše řetězec "Helord"
- F** Vypíše řetězec "Hellorld"

**6** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na bitové úrovni je typicky časově náročnější
- B** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- E** Typová konverze na sémantické úrovni je typicky časově náročnější
- F** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ

**7**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 49 prvků typu int
- B** vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C** způsobí zápis za konec přidělené paměti
- D** vytvoří pole o celkovém počtu 64 prvků typu int
- E** žádná z odpovědí není správná
- F** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7

**8** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- D** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel

**9**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '3 3 7'
- B** Nelze přeložit
- C** Žádná z odpovědí není správná
- D** Vypíše hodnoty '1 1 6'
- E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- F** Vypíše hodnoty '3 3 6'

**10**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A** 0.050000
- B** 3.950000
- C** 0.000000
- D** nelze přeložit
- E** žádná z odpovědí není správná



# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 89           |

**1** `#include <stdio.h>`  
`int main() {`  
    `unsigned char value1 = 0x55;`  
    `unsigned char value2 = 0xAA;`  
    `if (value1 & value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 && value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 | value2) printf("Hello ");`  
    `else printf("World ");`  
    `if (value1 || value2) printf("Hello ");`  
    `else printf("World ");`  
    `return 0;`  
`}`

Výše uvedený program vypíše:

- A** Žádná z odpovědí není správná
- B** Hello World Hello World
- C** World Hello World Hello
- D** Hello Hello Hello Hello
- E** Nelze přeložit
- F** World Hello Hello Hello

**2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Typová konverze na sémantické úrovni je typicky časově náročnější
- B** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
- C** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
- D** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
- E** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem
- F** Typová konverze na bitové úrovni je typicky časově náročnější

**3** `int main() {`  
    `int array[7][7];`  
    `for (int i = 1; i <= 7; i++) {`  
        `for (int j = 1; j <= 7; j++) array[i][j] = i;`  
    `}`  
    `return 0;`  
`}`

Výše uvedený program:

- A** vytvoří pole o celkovém počtu 64 prvků typu `int`
- B** vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- C** vytvoří pole o celkovém počtu 49 prvků typu `int`
- D** žádná z odpovědí není správná
- E** způsobí zápis za konec přidělené paměti
- F** vyplní všechny položky pole hodnotou z intervalu 1 do 7

**4** `#include <stdio.h>`  
`#include <stdlib.h>`  
`#include <string.h>`  
`void foo(unsigned char* pArray, int value) {`  
    `pArray[0] += 2;`  
    `pArray[value] += 2;`  
    `value++;`  
    `pArray[value] += 2;`  
`}`  
`int main(void) {`  
    `unsigned char array[10];`  
    `int value = 6;`  
    `memset(array, 1, sizeof(array));`  
    `foo(array, value);`  
    `printf("%d %d %d", array[0], array[6], value);`  
    `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše hodnoty '3 3 7'
- B** Žádná z odpovědí není správná
- C** Vypíše hodnoty '1 1 6'
- D** Vypíše hodnoty '3 3 6'
- E** Nelze přeložit
- F** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti `array`.

**5** `#include <stdio.h>`  
`void foo(int a) {`  
    `switch (a) {`  
        `case 0: break;`  
        `case 1: printf("Fr"); break;`  
        `case 2: printf("F");`  
        `case 3: printf("re"); break;`  
        `case 4: printf("e");`  
        `case 5:`  
        `case 6: break;`  
        `case 7:`  
        `case 8: printf("m"); break;`  
        `default: printf("edo");`  
    `}`  
`}`  
Která sekvence volání funkce `foo()` vypíše řetězec "Freedom":

- A** `foo(2);foo(0);foo(5);foo(7);foo(8);`
- B** `foo(1);foo(4);foo(6);foo(10);foo(8);`
- C** žádná z odpovědí není správná
- D** `foo(2);foo(3);foo(10);foo(7);`
- E** `foo(5);foo(2);foo(5);foo(10);foo(7);`

**6** `#include <stdlib.h>`  
`#include <stdio.h>`  
`void foo(int** array) {`  
 `*array = malloc(10);`  
 `array = NULL;`  
`}`  
`int main() {`  
 `int* array = NULL;`  
 `foo(&array);`  
 `if (array != NULL) printf("Hello");`  
 `else printf("World");`  
 `free(array);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** vypíše Hello
- B** nezpůsobí žádný memory leak
- C** vypíše World
- D** žádná z odpovědí není správná
- E** memory leak o velikosti 10 bajtů

**7** `#include <stdio.h>`  
`#include <string.h>`  
`int main() {`  
 `char str1[20];`  
 `strcpy(str1, "Hello world");`  
 `str1[4] = 0;`  
 `printf("%s", str1);`  
 `char* str2 = str1 + 8;`  
 `printf("%s", str2);`  
 `return 0;`  
`}`

Výše uvedený program:

- A** Vypíše řetězec "Helord"
- B** Vypíše řetězec "Hellorld"
- C** Nelze přeložit
- D** Vypíše řetězec "Helrld"
- E** Vypíše řetězec "Hellrld"
- F** Vypíše řetězec "Hell"

**8** `#include <stdlib.h>`  
`int main() {`  
 `int* pArray1 = NULL;`  
 `int* pArray2 = NULL;`  
 `int* pArray3 = NULL;`  
 `pArray1 = malloc(60);`  
 `pArray2 = pArray1;`  
 `pArray1 += 8;`  
 `pArray3 = malloc(60);`  
 `free(pArray2);`  
 `pArray2 = pArray3;`  
 `free(pArray2);`  
 `return 0;`  
`}`

Spuštění programu způsobí:

- A** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** pád programu
- C** memory leak o velikosti 120 bajtů
- D** memory leak o velikosti 60 bajtů
- E** žádná z odpovědí není správná
- F** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)

**9** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A** Memory leak lze detekovat s využitím nástrojů, např. Valgrind
- B** Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- C** Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D** Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel

**10** `#include <stdio.h>`  
`int main() {`  
 `float a = (int) 3.95;`  
 `int x = a;`  
 `printf("%f", x - a);`  
 `return 0;`  
`}`

Výše uvedený program vypíše:

- A** 0.050000
- B** nelze přeložit
- C** 3.950000
- D** žádná z odpovědí není správná
- E** 0.000000

# PB071: Prubezna\_20111024

| Jméno a příjmení – pište do okénka | UČO | Číslo zadání |
|------------------------------------|-----|--------------|
|                                    |     | 90           |

**1**

```
#include <stdio.h>
int main() {
    unsigned char value1 = 0x55;
    unsigned char value2 = 0xAA;
    if (value1 & value2) printf("Hello ");
    else printf("World ");
    if (value1 && value2) printf("Hello ");
    else printf("World ");
    if (value1 | value2) printf("Hello ");
    else printf("World ");
    if (value1 || value2) printf("Hello ");
    else printf("World ");
    return 0;
}
```

Výše uvedený program vypíše:

- A Žádná z odpovědí není správná
- B World Hello Hello Hello
- C Hello Hello Hello Hello
- D Nelze přeložit
- E Hello World Hello World
- F World Hello World Hello

**2** Která z uvedených tvrzení jsou pro jazyk C pravdivá?

- A Memory leak označuje paměť na haldě na kterou již neexistuje ukazatel
- B Memory leak označuje paměť na zásobníku nebo haldě na kterou již neexistuje ukazatel
- C Memory leak označuje paměť na zásobníku na kterou již neexistuje ukazatel
- D Memory leak lze detekovat s využitím nástrojů, např. Valgrind

**3**

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20];
    strcpy(str1, "Hello world");
    str1[4] = 0;
    printf("%s", str1);
    char* str2 = str1 + 8;
    printf("%s", str2);
    return 0;
}
```

Výše uvedený program:

- A Vypíše řetězec "Helrld"
- B Vypíše řetězec "Helord"
- C Nelze přeložit
- D Vypíše řetězec "Hell"
- E Vypíše řetězec "Hellrld"
- F Vypíše řetězec "Helloworld"

**4**

```
#include <stdio.h>
int main() {
    float a = (int) 3.95;
    int x = a;
    printf("%f", x - a);
    return 0;
}
```

Výše uvedený program vypíše:

- A žádná z odpovědí není správná
- B nelze přeložit
- C 3.950000
- D 0.000000
- E 0.050000

**5**

```
int main() {
    int array[7][7];
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 7; j++) array[i][j] = i;
    }
    return 0;
}
```

Výše uvedený program:

- A vyplní všechny položky na diagonále hodnotou z intervalu 1 do 7
- B vyplní všechny položky pole hodnotou z intervalu 1 do 7
- C způsobí zápis za konec přidělené paměti
- D žádná z odpovědí není správná
- E vytvoří pole o celkovém počtu 49 prvků typu int
- F vytvoří pole o celkovém počtu 64 prvků typu int

**6**

```
#include <stdio.h>
void foo(int a) {
    switch (a) {
        case 0: break;
        case 1: printf("Fr"); break;
        case 2: printf("F");
        case 3: printf("re"); break;
        case 4: printf("e");
        case 5:
        case 6: break;
        case 7:
        case 8: printf("m"); break;
        default: printf("edo");
    }
}
```

Která sekvence volání funkce foo() vypíše řetězec "Freedom":

- A foo(2);foo(3);foo(10);foo(7);
- B foo(1);foo(4);foo(6);foo(10);foo(8);
- C foo(2);foo(0);foo(5);foo(7);foo(8);
- D foo(5);foo(2);foo(5);foo(10);foo(7);
- E žádná z odpovědí není správná

- 7** Která z uvedených tvrzení jsou pro jazyk C pravdivá?
- A** Typová konverze na sémantické úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - B** Typová konverze na sémantické úrovni je typicky časově náročnější
  - C** Typová konverze na bitové úrovni je typicky časově náročnější
  - D** Pokud nedojde ke zápisu dat do paměti, tak lze typovou konverzi na bitové úrovni změnit na předchozí datový typ
  - E** Typová konverze na bitové úrovni nemění obsah paměti, pouze mění způsob její interpretace
  - F** Typová konverze na sémantické úrovni může být ztrátová, pokud přetypováváme z typu s větším rozsahem na typ s menším rozsahem

**8**

```
#include <stdlib.h>
#include <stdio.h>
void foo(int** array) {
    *array = malloc(10);
    array = NULL;
}
int main() {
    int* array = NULL;
    foo(&array);
    if (array != NULL) printf("Hello");
    else printf("World");
    free(array);
    return 0;
}
```

Výše uvedený program:

- A** žádná z odpovědí není správná
- B** memory leak o velikosti 10 bajtů
- C** vypíše World
- D** vypíše Hello
- E** nezpůsobí žádný memory leak

**9**

```
#include <stdlib.h>
int main() {
    int* pArray1 = NULL;
    int* pArray2 = NULL;
    int* pArray3 = NULL;
    pArray1 = malloc(60);
    pArray2 = pArray1;
    pArray1 += 8;
    pArray3 = malloc(60);
    free(pArray2);
    pArray2 = pArray3;
    free(pArray2);
    return 0;
}
```

Spuštění programu způsobí:

- A** memory leak o velikosti 480 bajtů (za předpokladu, že int zabírá 4 bajty)
- B** pád programu
- C** žádná z odpovědí není správná
- D** memory leak o velikosti 240 bajtů (za předpokladu, že int zabírá 4 bajty)
- E** memory leak o velikosti 120 bajtů
- F** memory leak o velikosti 60 bajtů

**10**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void foo(unsigned char* pArray, int value) {
    pArray[0] += 2;
    pArray[value] += 2;
    value++;
    pArray[value] += 2;
}
int main(void) {
    unsigned char array[10];
    int value = 6;
    memset(array, 1, sizeof(array));
    foo(array, value);
    printf("%d %d %d", array[0], array[6], value);
    return 0;
}
```

Výše uvedený program:

- A** Vypíše hodnoty '3 3 6'
- B** Nelze přeložit
- C** Vypíše hodnoty '3 3 7'
- D** Vypíše hodnoty '1 1 6'
- E** Nelze předem určit, co vypíše. Závisí na hodnotách v paměti array.
- F** Žádná z odpovědí není správná