PA2 1.test
progtest

Martin Franc

4. března 2014

Otázka 1: Předpokládejme následujcící kód. Jaký bude výstup?

```
int * a = new int[69];
cout << *a;
delete [] a;
```

Program půjde zkompilovat, nespadne, ale výsledek není definovaný.

Otázka 2: Předpokládejme následujcící kód. Jaký bude výstup?

```
int * a = new int(65);
cout << *a;
delete [] a;
```

Program jde zkompilovat, ale pracuje špatně s pamětí a může spadnout.

Otázka 3: Předpokládejme následujcící kód. Jaký bude výstup?

```
int * a = new int(73);
cout << *a;
delete a;
```

Výsledek je 73.

Otázka 4: Předpokládejme následujcící kód. Jaký bude výstup?

```
int i, * a = new int[26];
for (i = 0; i < 26; i ++)
   a[i] = i;
delete a;
cout << i;
```

Program jde zkompilovat, ale pracuje špatně s pamětí a může spadnout.

Otázka 5: Předpokládejme následujcící kód. Jaký bude výstup?

```
int i, * a = new int[37];
for (i = 0; i < 37; i ++)
   a[i] = i;
delete [] a;
cout << i;
```

Výsledek je 37.

Otázka 6: Předpokládejme následujcící kód. Jaký bude výstup?

```
int i, * a = new int(114);
for (i = 0; i < 114; i ++)
   a[i] = i;
delete a;
cout << i;
```

Program jde zkompilovat, ale pracuje špatně s pamětí a může spadnout

---

Otázka 7: Předpokládejme následujcící kód. Jaký bude výstup?

```
int & foo(int * a){
   return a[3];
}
...
int y[10], * x = y;
foo(x) = 78;
x++;
cout << y[3];
```

Výsledek je 78

Otázka 8: Předpokládejme následujcící kód. Jaký bude výstup?

```
int & foo(int * a){
   return a[3];
}
...
int * x = new int[10];
int & y = foo(x);
delete [] x;
y = 14;
cout << y;
```

Program jde zkompilovat, ale po spuštění může spadnout/spadne.

Otázka 9: Předpokládejme následujcící kód. Jaký bude výstup?

```
int & foo (int * a){
   return a[3];
}
...
int y[10];
foo(y) = 55;
cout << y[3];
```

Výsledek je 55.

Otázka 10: Co bude výsledkem následujícího kódu (kompilace podle ISO/IEC 14882:1998)?

```
int foo(int a){
   return 10;
}
const char * foo (float b){
   return "Hello";
}
...
cout << foo('a');
```

Výsledek je 10.

Otázka 11: Předpokládejme následujcící kód. Jaký bude výstup?

```
int foo(int a){
   return 10;
}
const char * foo (int b){
   return "Hello";
}
...
int x = foo(10);
cout << x;
```

Kód nepůjde zkompilovat.

Otázka 12: Co bude výsledkem následujícího kódu (kompilace podle ISO/IEC 14882:1998)?

```
int foo(int a){
   return 20;
}
const char foo(double b){
   return -3;
}
int x = foo(6.0);
cout << x;
```

Výsledek záleží na konkrétním kompilátoru nebo platformě.

Otázka 13: Co bude výsledkem následujícího kódu (kompilace podle ISO/IEC 14882:1998)?

```
int foo(int a){
   return 10;
}
const char * foo(float b){
   return "Hello";
}
...
cout << foo(12.5);
```

Kód nepůjde zkompilovat.

Otázka 14: Co bude výsledkem následujícího kódu (kompilace podle ISO/IEC 14882:1998)?

```
int foo (char a, double b){
   return 10;
}
int foo (int a, int b){
   return 20;
}
...
cout << foo('a', 12);
```

Kód nepůjde zkompilovat.

Otázka 15: Co bude výsledkem následujícího kódu (kompilace podle ISO/IEC 14882:1998)?

```
int foo(char a, double b){
   return 10;
}
int foo (int a, int b){
   return 20;
}
...
cout << foo ('a', 'b');
```

Kód nepůjde zkompilovat.

Otázka 16: Co bude výsledkem následujícího kódu (kompilace podle ISO/IEC 14882:1998)?

```
int foo(char a, double b){
   return 10;
}
int foo(int a, int b){
   return 20;
}
...
cout << foo(12.5, 10);
```

Výsledek je 20.

Otázka 17: Předpokládejme následujcící kód. Jaký bude výstup?

```
int & foo(int b){
   int * a = new int(99);
   (*a) += b;
   return *a;
}
...
int x = foo(0);
int y = foo(1);
cout << (x + y);
```

Program půjde zkompilovat, nespadne, ale neuvolní po sobě alokované prostředky (paměť).

Otázka 18: Předpokládejme následujcící kód. Jaký bude výstup?

```
int & foo(int b){
   int a = 86;
   a += b;
   return a;
}
...
int & x = foo(0);
int & y = foo(1);
cout << (x + y);
```

Program jde zkompilovat, ale po spuštění může spadnout/spadne.

Otázka 19: Předpokládejme následujcící kód. Jaký bude výstup?

```
int & foo(int b){
   static int a = 61;
   a += b;
   return a;
}
...
int & x = foo(0);
int & y = foo(1);
cout << (x + y);
```

Výsledek je 124.

Otázka 20: Předpokládejme následujcící kód. Jaký bude výstup?

```
int foo (int b){
   int a = 91;
   a += b;
   return a;
}
...
int & x = foo(0);
int & y = foo(1);
cout << (x + y);
```

Program nepůjde zkompilovat.

Otázka 21: Předpokládejme následujcící kód. Jaký bude výstup?

```
int & foo(int b){
   int * a = new int(56);
   (*a) += b;
   return *a;
 }
...
int x = foo(0);
int y = foo(1);
cout << (x + y);
```

Program půjde zkompilovat, nespadne, ale neuvolní po sobě alokované prostředky (paměť).

Otázka 22: Předpokládejme následujcící kód. Jaký bude výstup?

```
int & foo(int b){
   static int a = 35;
   a += b;
   return a;
}
...
int x = foo(0);
int y = foo(1);
cout << (x + y);
```

Výsledek je 71.

Otázka 23: Předpokládejte následující kód. Jaká bude hodnota proměnné x?

```
int bar(int a){
   return 2 * a;
}
int foo(const int & a){
   return bar ( a + 2 );
}
...
int DATA = 61;
int x = foo(DATA);
```

Výsledkem je hodnota 126.

Otázka 24: Předpokládejte následující kód. Jaká bude hodnota proměnné x?

```
int bar(int a){
   return 2 * a;
}
int foo(int & a){
   return bar(a + 2);
}
...
int DATA = 20;
int x = foo(DATA);
```

Výsledkem je hodnota 44.

Otázka 25: Předpokládejte následující kód. Jaká bude hodnota proměnné x?

```
int bar(int a){
   return 2 * a;
}
int foo(int & a){
   return bar(a + 2);
}
...
const int DATA = 44;
int x = foo(DATA);
```

Program nepůjde zkompilovat.

Otázka 26: Předpokládejte následující kód. Jaká bude hodnota proměnné x?

```
int bar(int a){
   return 2 * a;
}
int foo (const int & a){
   return bar (a + 2);
}
...
const int DATA = 39;
int x = foo(DATA);
```

Výsledkem je hodnota 82.

Otázka 27: Předpokládejte následující kód. Jaká bude hodnota proměnné x?

```
int bar(int a){
   return 2 * a;
}
int foo(int & a){
   return bar(a + 2);
}
...
int x = foo(74);
```

Program nepůjde zkompilovat.

Otázka 28: Předpokládejte následující kód. Jaká bude hodnota proměnné x?

```
int bar(int a){
   return 2 * a;
}
int foo(const int & a){
   return bar(a + 2);
}
...
int x = foo(84);
```

Výsledek je 172.

Otázka 29: Předpokládejme následujcící kód. Jaký bude výstup?

```
void foo(int a){
   a += 2;
}
void bar(int & a){
   a += 2;
   foo(a);
   a += 2;
}
...
int x = 27;
bar(x);
```

Výsledek je 31.

Otázka 30: Předpokládejme následujcící kód. Jaká bude hodnota proměnné x?

```
void foo(int & a){
   a += 2;
}
void bar(int a){
   a += 2;
   foo (a);
   a += 2;
}
...
int x = 73;
bar(x);
```

Výsledek je 73.

Otázka 31: Předpokládejme následujcící kód. Jaká bude hodnota proměnné x?

```
void foo(int & a){
   a += 2;
}
void bar(int & a){
   a += 2;
   foo(a);
   a += 2;
}
...
int x = 4;
bar(x);
```

Výsledek je 10.

Otázka 32: Předpokládejme následujcící kód. Jaká bude hodnota proměnné x?

```
void foo(int & a){
   a += 2;
}
void bar(int & a){
   a += 2;
   foo(a);
   a += 2;
}
...
int x = 88;
int & y = x;
bar(y);
```

Výsledek je 94.

Otázka 33: Předpokládejme následujcící kód. Co zobrazí?

```
void foo(const char * a){
   a += 2;
}
void bar(const char * & a){
   a ++;
   foo(a);
   a ++;
}
...
const char * x = "To iterate is human, to recurse divine.";
bar(x);
cout << x;
```

Zobrazí se "[mezera]iterate is human, to recurse divine.".

Otázka 34: Předpokládejme následujcící kód. Co zobrazí?

```
void foo(const char * & a){
   a += 2;
}
void bar(const char * a){
   a ++;
   foo(a);
   a ++;
}
...
const char * x = "Java is, in many ways, C++-.";
bar(x);
cout << x;
```

Zobrazí se "Java is, in many ways, C++-.".

Otázka 35: Předpokládejme následujcící kód. Jaký bude výstup?

```
void foo(const char * & a){
   a += 2;
}
void bar(const char * & a){
   a ++;
   foo(a);
   a ++;
}
...
const char * x = "Java is, in many ways, C++-.";
const char * & y = x;
bar(y);
cout << x;
```

Zobrazí se "[mezera]is, in many ways, C++-.".

Otázka 36: Předpokládejme následujcící kód. Co zobrazí?

```
void foo (const char * & a){
   a += 2;
}
void bar (const char * & a){
   a ++;
   foo(a);
   a ++;
}
...
const char * x = "Java is good, C++ better";
bar(x);
cout << x;
```

Zobrazí se "[mezera]is good, C++ better".
