Pracovní list 3: Sekvence a větvení

Co už máme znát

- jaké jsou základní struktury pro vytváření algoritmu;
- jak tyto prvky vzájemně kombinovat;
- jak tyto struktury zapsat v rámci programovacího jazyka C++.

Kontrolní otázky

3.1 K čemu slouží datové typy?

3.2 Jaké datové typy jsou k dispozici pro číselné hodnoty?

3.3 Jaké aritmetické operace jsou k dispozici u číselných datových typů?

3.4 Jaké operace s jednotlivými bity lze provádět u celočíselných hodnot?

3.5 Jak se deklarují proměnné a k čemu slouží?

3.6 Co je a jak se připojí k danému programu knihovna?

3.7 Co to jsou jmenné prostory v jazyce C++?

3.8 Která knihovna zahrnuje příkazy `std::cin` a `std::cout`? K čemu tyto příkazy slouží?

3.9 K čemu je v C++ konstrukce `int main()`?

3.10 K čemu slouží příkaz `return 0;` ve funkci `main`?

Příprava na cvičení

Existuje mnoho prostředí, kde lze psát příkazy jazyka C++. Na cvičeních se bude využívat on-line prostředí `https://www.onlinegdb.com/` nebo překladač jazyka C++ a libovolný editor kódu z terminálu na výukovém serveru akela.

Je tedy potřeba počítač s připojením k internetu nebo s nainstalovaným odpovídajícím překladačem jazyka C++. Hodit se vám budou i vývojové diagramy z prvního cvičení.

Řešené příklady

Příklad 3.1 Vytvořte program v jazyce C++ pro výpočet aritmetického průměru tří čísel.

Řešení: V zápisu programu si můžeme všimnout následujících prvků:
Kód `#include <iostream>` připojí knihovnu, která umožňuje číst hodnoty ze standardního vstupu a vypisovat na standardní výstup pomocí příkazů `std::cin` a `std::cout`. Tato knihovna bude připojována ve všech příkladech, které zde budou řešeny. Příkaz `using namespace std;` umožní vynechat „std::“ při používání příkazů z tohoto jmenného prostoru. Programy psané v C++ vždy začínají vykonáním funkce (podprogramu) `main`. Proto se do těla této funkce zapisují příkazy, které mají být provedeny. Následuje pak deklarace proměnných. Při deklaraci je možné proměnným rovnou přiřadit i hodnotu. To však v tomto případě není potřeba. Datový typ `int` je celočíselný datový typ. V tomto příkladu budou tohoto typu proměnné `cislo1`, `cislo2`, `cislo3` a `suma`. Datový typ `float` slouží pro práci s desetinnými čísly. Do proměnné `prumer` bude přiřazen výsledek podílu, což může být desetinné číslo. Proto je proměnná `prumer` typu `float`. Příkazy pro vypsání textu jako např. `cout<<”Zadejte 1. číslo: ”;` nejsou pro vyřešení úkolu nezbytně nutné, ale informují případného uživatele o tom, co má udělat. V těchto instrukcích, stejně jako při komentářích k výsledným hodnotám budeme vždy používat národní znaky. Příkaz `cin>>cislo1;` od uživatele načte hodnotu do proměnné `cislo1`. Obdobně je to i s dalšími příkazy pro načtení. Pomocí `suma=cislo1+cislo2+cislo3;` dojde k přiřazení součtu tří načtených hodnot uložených v proměnných `cislo1`, `cislo2` a `cislo3` do proměnné `suma`. Příkaz `prumer=float(suma)/3;` provede podíl hodnoty v proměnné `suma` a čísla `3` a přiřadí jej do proměnné `prumer`. Poněkud nejasný může být zápis `float(suma)`. Jde o tzv. přetypování. Pokud by k němu nedošlo, operátor dělení `/` by byl chápán jako celočíselné dělení, protože by se dělila dvě celá čísla. Přetypováním jednoho z operandů na desetinné číslo však sdělíme překladači úmysl vyjádřit podíl i s desetinnými místy. Následuje pak jen výpis výsledku a příkaz `return 0;`, který se standardně používá jako informace pro operační systém sdělující úspěšné vykonání programu.

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int cislo1, cislo2, cislo3, suma;
6      float prumer;
7      cout<<”Zadejte 1. číslo: ”; //tyto dialogy již nadále NEBUDEME používat
8      cin>>cislo1;
9      cout<<”Zadejte 2. číslo: ”;
10     cin>>cislo2;
11     cout<<”Zadejte 3. číslo: ”;
12     cin>>cislo3;
13     suma=cislo1+cislo2+cislo3;
14     prumer=float(suma)/3;
15     cout<<”Aritmetický průměr zadaných čísel je ”<<prumer;
16     return 0;
17 }
```

V následující variantě je ukázáno, že je možné načíst všechna čísla v rámci jednoho příkazu. Všimněte si také operace `endl`. Ta vypíše konec řádku, takže se na výstupu přesune kurzor na nový řádek.

```
18 #include <iostream>
19 using namespace std;
```

```
20
21 int main(){
22     int cislo1,cislo2,cislo3,suma;
23     float prumer;
24     cin>>cislo1>>cislo2>>cislo3;
25     suma=cislo1+cislo2+cislo3;
26     prumer=float(suma)/3;
27     cout<<"Aritmetický průměr zadaných čísel je "<<prumer;
28     return 0;
29 }
```

V mnoha případech budeme uvažovat spuštění programu v příkazovém řádku nebo v rámci dávky příkazů operačního systému. V tom případě se předpokládá, že program si bere vstupní data buď z připraveného diskového souboru, nebo ze standardního výstupu předchozího procesu v koloně příkazů na příkazovém řádku. Vypisování instrukcí pro uživatele, který u programu v tomto případě vůbec není přítomen, je tedy nejen zcela zbytečné, ale je přímo **nežádoucí**, protože výsledky a všechny výpisy jsou ukládány do jiného diskového souboru nebo posílány na vstup následujícího programu v koloně. Pokud se do výsledků budou míchat i tyto „instrukční" texty pro (nepřítomného) uživatele, může dojít ke znehodnocení nebo zmatení výstupních hodnot.

Příklad 3.2 Vytvořte program v jazyce C++ pro výpis dvou načtených čísel seřazených sestupně.

Řešení: I zde je použita knihovna `iostream` a bude se pracovat s příkazy ze jmenného prostoru `std`. Nejprve jsou nadeklarovány proměnné a následně se do nich načtou hodnoty ze vstupu. Pomocí podmíněného příkazu `if` se zjistí, zda první zadané číslo je větší než druhé zadané číslo. Pokud ano, je proveden příkaz `cout<<cislo1<<", "<<cislo2;`. V opačném případě se provede příkaz zapsaný v části `else`. Jistě jste si již všimli, že je možné kombinovat při výpisu proměnné s literály. Literál je přímý zápis určité hodnoty v programovacím jazyce. Může se jednat o čísla, znaky, řetězce, případně i jiné konstanty zapsané definovaným identifikátorem. V tomto případě je literálem řetězec `", "` vypisující čárku mezi dvěma výslednými čísly.

```
30 #include <iostream>
31 using namespace std;
32
33 int main(){
34     int cislo1, cislo2;
35     cin >> cislo1 >> cislo2;
36     if (cislo1>cislo2){
37         cout << cislo1 << ", " << cislo2;
38     }
39     else{
40         cout << cislo2 << ", " << cislo1;
41     }
42     return 0;
43 }
```

Příklad 3.3 Vytvořte program v jazyce C++ pro určení minima za tří zadaných čísel.

Řešení: Princip, jak postupovat, je popsán v prvním pracovním listu. Zde je vidět, jak správně příklad přepsat pomocí jazyka C++. Povšimněte si zanoření jednotlivých podmíněných příkazů. Doporučuje se dodržovat odsazení, neboť to výrazně zvyšuje čitelnost a přehlednost kódu.

```
44 #include <iostream>
45 using namespace std;
46
47 int main(){
48     int cislo1,cislo2,cislo3;
49     cin>>cislo1>>cislo2>>cislo3;
50     if (cislo1>cislo2){
51         if (cislo2>cislo3){
52             cout<<”Nejmenší je: ”<<cislo3;
53         }
54         else{
55             cout<<”Nejmenší je: ”<<cislo2;
56         }
57     }
58     else{
59         if (cislo1>cislo3){
60             cout<<”Nejmenší je: ”<<cislo3;
61         }
62         else{
63             cout<<”Nejmenší je: ”<<cislo1;
64         }
65     }
66     return 0;
67 }
```

Příklad 3.4 Vytvořte program v jazyce C++, který provede převod zadaného celého čísla na číslo opačné prostřednictvím binárních operací v doplňkovém kódu.

Řešení: Číslo opačné (ke kladnému je záporné, k zápornému je kladné) získáme v doplňkovém kódu tak, že provedeme inverzi všech bitů a k výsledku přičteme jedničku. Výsledný program může mít následující podobu:

```
68 #include <iostream>
69 using namespace std;
70
71 int main(){
72     int cele;
73     cin >> cele;  //přečtení celého čísla ze vstupu
74     cele =   cele + 1; //přepočet na opačné číslo
75     cout << ”Opačné číslo je: ” << cele << endl;
```

```
76      return 0;
77  }
```

Příklad 3.5 Vytvořte program v jazyce C++ pro zjištění, zda zadané číslo je, či není sudé.

Řešení: Sudé číslo poznáme podle toho, že je beze zbytku dělitelné dvěma (tj. zbytek po dělení je roven nule). Pokud uvažujeme celá čísla, kladná i záporná, nesmíme zapomenout, že zbytek po dělení může být i `1` nebo `-1`, takže *netestujeme vůči jedničce*.

Pro zjištění sudosti můžeme však využít i skutečnosti, že sudé číslo v binární podobě má na nejnižším bitu nulu a tu zjistíme maskováním a bitovým součinem.

Ukážeme si obě varianty řešení tohoto úkolu, nejprve zjištění zbytku po dělení dvěma:

```
78  #include <iostream>
79  using namespace std;
80
81  int main(){
82      int cele;
83      cin >> cele;  //přečtení celého čísla ze vstupu
84      if (cele % 2 == 0) //Pozor, testujeme vždy na rovnost nule!
85         cout << "Číslo je sudé." << endl;
86      else cout << "Číslo je liché." << endl;
87      return 0;
88  }
```

Druhá varianta využívá možnosti práce s jednotlivými bity (uvádíme jen jádro programu):

```
89      int cele;
90      cin >> cele;  //přečtení celého čísla ze vstupu
91      if (cele & 1 == 0) //nejnižší bit byl nulový
92         cout << "Číslo je sudé." << endl;
93      else cout << "Číslo je liché." << endl;
```

Příklady

Příklad 3.6 Vytvořte program v jazyce C++ pro výpočet součinu tří zadaných čísel.

Příklad 3.7 Vytvořte program v jazyce C++ pro výpočet obsahu a obvodu obdélníka zadaného dvěma stranami.

Příklad 3.8 Vytvořte program v jazyce C++ pro výpočet počtu hlav a nohou všech zvířat na farmě, kde chovají husy a ovce. Vstupem jsou počty jednotlivých zvířat podle druhu. Uvažujte pouze zdravá zvířata.

Příklad 3.9 Vytvořte program v jazyce C++ pro výpočet kořene lineární rovnice $ax + b = 0$ (tedy hodnoty x). Koeficienty a a b načte algoritmus ze vstupu.

Příklad 3.10 Vytvořte program v jazyce C++ pro určení minima ze dvou zadaných čísel.

Příklad 3.11 Vytvořte program v jazyce C++ pro zjištění hodnot tří nejnižších bitů vnitřní reprezentace zadaného celého čísla.

Příklad 3.12 Vytvořte program v jazyce C++ pro zjištění hodnot čtyř nejvyšších bitů vnitřní reprezentace zadaného znaku.

Příklad 3.13 Vytvořte program v jazyce C++ pro zjištění, kolik bajtů má znak v kódování UTF-8 zadaný ze vstupu.

Příklad 3.14 Vytvořte program v jazyce C++ pro absolutní hodnoty rozdílu dvou zadaných čísel.

Příklad 3.15 Vytvořte program v jazyce C++ pro určení, zda zadané číslo je kladné, záporné či nula.

Příklad 3.16 Vytvořte program v jazyce C++ pro výpočet obvodu trojúhelníka zadaného délkami jeho tří stran.

Příklad 3.17 Vytvořte program v jazyce C++ pro výpočet obsahu pravoúhlého trojúhelníka zadaného délkami jeho odvěsen.

Příklad 3.18 Vytvořte program v jazyce C++ pro výpočet obvodu pravoúhlého trojúhelníka zadaného délkami jeho odvěsen.

Příklad 3.19 Vytvořte program v jazyce C++ pro určení, zda je trojúhelník zadaný svými délkami stran rovnostranný, rovnoramenný nebo pravoúhlý.

Příklad 3.20 Vytvořte program v jazyce C++ pro načtení tří hodnot a jejich vzestupný výpis (výpis od nejmenší po největší).

Příklad 3.21 Vytvořte program v jazyce C++ pro výpočet kořenů kvadratické rovnice na základě zadaných koeficientů a, b, c, kde platí $ax^2 + bx + c = 0$.

Co máme po cvičení umět

- Jak napsat základní prvky programu v jazyce C++: připojení knihovny, práci se jmenným prostorem, hlavní funkci s příkazem `return`.
- Jak v C++ deklarovat proměnné daného datového typu.
- Jak v C++ načíst vstupní data do proměnných.

- Jak v C++ provádět výpis.
- Jak v C++ používat sekvenci a podmíněné příkazy.
- Jak v C++ do sebe podmíněné příkazy zanořovat.

Kontrolní otázky

3.11 Co je klíčové slovo?

3.12 Co je to literál?

3.13 Jak se změní zápis programů, neuvedeme-li příkaz `using namespace std;`?

3.14 Jak se zapisuje podmíněný příkaz?

3.15 Jak se zapisuje podmíněný výraz?

3.16 Kolik podmíněných příkazů můžeme do sebe maximálně zanořit?

3.17 Jak lze jedním příkazem přečíst několik hodnot ze standardního vstupu?

3.18 Jak se do výpisu výstupních hodnot mohou vkládat formátovací znaky a konce řádků?

3.19 Čemu se vyhýbáme, pokud píšeme program spouštěný z příkazového řádku nebo skriptu operačního systému?