1. Úvod do jazyka C++

Překlad (kompilace) programu

Obecně: *g++ -o spustitelny_soubor zdrojovy_soubor.c*
V praxi je vhodné používat parametry *-Wall* a *-pedantic*
příklad: *g++ -Wall -pedantic -o myprog myprog.c*

Nástroj *make*

Překlad se řídí instrukcemi zapsanými v souboru *Makefile* nebo makefile.

Jméno cíle (tj. jméno spustitelného souboru) zakončujeme dvojtečkou, pak uvádáme jména závislých souborů oddělených mezerou (jedná se zpravidla o zdrojový soubor s programem). Dále uvádíme příkazy, každý na samostatném řádku, na začátku řádku musí být znak tabulátoru.

Překlad spustíme příkazem *make jmeno_cile*, spustíme-li *make* bez uvedení názvu cíle provede se první cíl.

```
# Radky s komentarem zacinaji znakem hash. #
Kompilujeme soubor myprog.cpp a
# vytvorime spustitelny soubor myprog

myprog: myprog.cpp
  g++ -Wall -pedantic -o myprog myprog.cpp
```

Zde musí být tabulátor, ne mezery!

Struktura programu v C++

Na začátku programu se nachází direktiva pro vložení hlavičkových souborů (používají se jiné hlav. soubory než vC)

Pro použití standardních funkcí musíme deklarovat prostor jmen pomocí klíčových slov `using namespace`

Program začíná funkcí `main()` podobně jako v C

Komentáře jsou stejné jako v C, tj. dvojice /* a */ nebo //

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

int main()
{
      cout << "Hello, World!" << endl;
      return 0;
}
```

Vstup a výstup v C++

Pro vstup a výstup používáme vstupní a výstupní proudy (angl. streams) `cin` a cout.

Pro výstup používáme cout ve spojení s operátorem <<

Pro vstup používáme cin ve spojení s operátorem >>

Operátory << a >> můžeme řetězit (tj. používat je opakovaně za sebou v kombinaci s proměnnými a konstantami)

Pro použití vstupů a výstupů musí být použit hlavičkový soubor *iostream* a deklarováno using namespace std.

Chceme-li používat vstupně/výstupní funkce z C (`printf()` a pod.) deklarujeme hlavičkový soubor `cstdio` (musíme také deklarovat `using namespace std`).

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

int main()
{
  int a = 0;
  cout << "Jednoduchy text\n";
  cout << "Zretezene " << "operatory";
  cin >> a;     // Ukazka nacteni cisla
  return 0;
}
```

Výstup v C++

Za operátorem << může následovat libovolná řetězcová, znaková nebo číselná konstanta nebo proměnná (dojde k automatické konverzi na řetězec).

Konec řádku můžeme zapsat jako znak '\n' (jako v C) nebo použít `endl` (zkratka z *end of line*).

```
int a = 1;
double d = 3.14
char c = 'W';

cout << "Klasicke zakonceni radku\n";
cout << "Totez ale trochu jinak" << '\n';
cout << "Zakonceni radku" << endl;

cout << 10;      // Vypis celeho cisla
cout << 23.175; // Vypis desetinneho cisla
cout << a;       // Vypis hodnoty promenne
cout << d;       // Vypis hodnoty promenne
cout << c;       // Vypis hodnoty promenne

// Zretezeni vice operatoru vystupu:
cout << "Cele cislo: " << a
       << " nasleduje realne cislo: "
       << d << endl;
```

Vstup v C++

Za operátorem >> uvedeme jméno proměnné do které se má načíst hodnota ze vstupu (tj. zpravidla z klávesnice).

Hodnota na vstupu musí odpovídat typu proměnné (celé, reálné číslo, znak).

Při čtení se přeskakují bílé znaky (mezery, tabelátory, znak konce řádku) předcházející načítané hodnotě.

Pokud použijeme zřetězení operátoru >> hodnoty se načítají do proměnných v pořadí zleva doprava.

```
int a = 0, b1 = 0, b2 = 0, b3 = 0;
double d = 0.0;
char c = ' ';

cin >> a;  // Nacteni cisla do promenne a
cin >> d;  // Nacteni realneho cisla
cin >> c;  // Nacteni cnaku do promenne c

// Postupne se nactou celociselne hodnoty
// do promennych b1, b2, b3
cin >> b1 >> b2 >> b3;
```

Řetězce v C++

Pro práci s řetezci používáme proměnné typu `string`.

Na začátku je třeba vložit hlavičkový soubor *string* deklarovat `using namespace std`.

Řetězcovou proměnnou můžeme inicializovat pomocí řetězcové konstanty, kterou uvedeme za = nebo do závorek.

Řetězcová proměnná typu string může obsahovat řetězec libovolné velikosti, paměť pro znaky je automaticky alokována/dealokována podle délky řetězce.

```
#include <string>
using namespace std;

int main()
{
  // Ruzne zpusoby inicializace retezcu
  string s1 = "Tady je nejaky text";
  string s2("Tady je nejaky text");
  string s3;  // Prazdny retezec
  return 0;
}
```

Operace s řetězci v C++

Operátor = slouží k přiřazení hodnoty řetězci z jiné řetězcové proměnné nebo konstanty.

Pro spojování řetězců používáme operátor +

Operátor += používáme k připojení řetězce k jinému řetězci.

Operátory ==, !=, <, <=, >, >= slouží k porovnávání řetězců (lexikografickému).

Pro práci s řetězci nelze použít operátory -, *, /

```
string s1, s2;

s1 = "Skakal";
s2 = s1 + " pes pres";
s2 += " oves";

if (s2 == "Skakal pes pres oves")
  cout << "Retezce se shoduji";
```

Výstup řetězcové proměnné

Řetězcovou proměnnou můžeme zapsat na výstup pomocí operátoru <<

```
int a = 3;
string s1 = "Tady je nejaky text";

cout << "Toto je retezec s1: " << s1
       << endl << "a promenna a:" << a;

// Operator + ma prioritu pred <<
cout << s1 + " pridany text";
// Pouziti zavorek je vsak prehednejsi
cout << (s1 + " na konec");
```

Vstup do řetězcové proměnné

Řetězcovou proměnnou můžeme použít pro čtení vstupu pomocí operátoru >>.

Při načítání se ignorují počáteční mezery (resp. bílé znaky) začíná se načítat až první nebílý znak.

Při čtení se tedy vždy načítá pouze jedno slovo, tj. čtou se znaky ze vstupu tak dlouho dokud se nevyskytne bílý znak (mezera, tabelátor, znak konce řádku).

```
string s;
cout << "Zadej text: ";
cin >> s;
cout << "Nacteny text: " << s << endl;
```

Logický typ `bool`

Na rozdíl od C existuje v C++ logický typ `bool`.

Proměnná typu bool nabývá hodnot `true` nebo `false`.

```
int a = 3, b = 9;
bool val = true;
if (a < b)
  val = false;
if (val == true) // Totez jako if (val)
  a += b;
```

Knihovna g2

Při použití knihovny g2 je třeba při překaldu použít následující parametry kompilátoru:

-lg2 -lgd -lX11 (l je zde malé L)

Na začátek zdrojového souboru je třeba vložit hlavič. soubory

```
#include <g2.h>
#include <g2_X11.h>
```

```
int dev = 0;

// Otevreni okna
dev = g2_open_X11(500, 500);

// Tloustka cary bude 5 pixelu:
g2_set_line_width(dev, 5);
// Pouzije se barva cislo 1 (cerna):
g2_pen(dev, 1);
// Kruznice v bode 250, 250 s polomerem 80
g2_circle(dev, 250, 250, 80);

cout << "Stisknete Enter!" << endl;
cin.get();  // Ceka na stisknuti Enter
cin.get();  // Nekdy je treba to volat 2x

g2_close(dev);
```

Čísla barev předávané funkci `g2_pen()` mají hodnoty 0 až 26, např. 0 bílá, 1 černá, 3 modrá, 7 zelená, 19 červená, 25 žlutá.

Dokumentaci ke knihovně g2 lze nalézt na:
http://ncbr.chemi.muni.cz/~martinp/g2/modules.html

Dodržujte následující pravidla

- Pro kompilaci programu používejte nástroj *make*
- Na začátek programu nezapomeňte vložit hlavičkové soubory *iostream* (pro práci se vstupy a výstupy) a *string* (pro práci s řetězci) a deklaraci `using namespace std`
- Proměnné při deklaraci vždy inicializujte vhodnou hodnotou (zpravidla 0)

Úloha 1 — 1 bod

Vytvořte program, který si od uživatele vyžádá dvě celočíslené hodnoty oddělené mezerou a vypíše jejich součet.

Úloha 2 — 1 bod

Vytvořte program, který bude v cyklu od uživatele požadovat zadání slov, dokud uživatel nezadá slovo END. Potom vypíše všechna dosud zadaná slova oddělená mezerou (kromě slova END). Program také vypíše počet zadaných slov (bez END).

Úloha 3 — 2 body

Vytvořte program, který vykreslí na obrazovku kružnici pomocí knihovny g2. Program si na začátku vyžádá od uživatele souřadnice středu kružnice (v pixelech), poloměr kružnice (v pixelech) a číslo barvy (1, 3, 7, 19 nebo 25). Potom zobrazí okno a v něm vykreslí kružnici.

Kód pro načítání hodnot umístěte do funkce `main()`, ale kód pro vykreslení kružnice umístěte do samostané funkce, které předáte potřebné hodnoty (tj. souřadnice středu, poloměr, číslo barvy).

Úloha 4 — nepovinná, 1 bod

Předchozí úlohu modifikujte tak, že barva nebude zadána formou čísla, ale jako text (black, blue, green, red, yellow).