Pokročilé programování v jazyce C pro chemiky

(C3220)

Úvod do jazyka C++

Objektové programování

Objektový přístup – data a metody pro manipulaci s nimi jsou sdruženy společně v rámci jedné entity

Objektové programování – pracuje s objekty = speciální proměnné obsahující datové položky (podobně jako struktury v C) a zároveň funkce pro manipulaci s nimi

Simula - první objektově orientovaný jazyk (1967) – ovlivnila mnoho dalších jazyků

Další objektové jazyky: Smalltalk, C++, Java, C#, Objective-C, Python

Novější verze jazyků COBOL, Pascal (Turbo Pascal, Delphi), Fortran (Fortran 2003), BASIC (Visual BASIC), Ada (Ada 95), Perl (Perl 5), PHP

Jazyk C++

Výhody:

- Široce používaný
- Vychází z prověřeného jazyka C
- Výhody objektového programování (vč. pokročilých technik)
- Rychlost (jedná se o kompilovaný jazyk)
- Široká dostupnost kvalitních překladačů pro různé hardwarové platformy, vývojových a ladících nástrojů a knihoven

Nevýhody:

- Relativní složitost
- Chybí systém pro automatickou správu paměti (garbage collection)
- Některé knihovny nejsou standardizované (multithreading, network I/O, GUI) – omezuje portabilitu
- Změna zdrojového kódu vyžaduje rekompilaci (jedná se o kompilovaný jazyk)

Relativní nevýhody:

- Práce s ukazateli (efektivní, ale málo bezpečná)

Historie jazyka C++

1969 - 1973 (nejvíce 1972) vyvinut jazyk C (Dennis Ritchie v AT&T Bell Labs)

1979 – započal vývoj *C with classes* (AT&T Bell Labs, B. Stroustrup) – rozšíření C o vlastnosti jazyka Simula

1983 – jméno změněno na C++

1986 – Bjarne Stroustrup publikoval knihu “The C++ Programming Language”

1992 – v Sun Microsystems vyvinuli jazyk Java (inspirován C++)

2000 – v Microsoft vyvinuli v rámci projektu .NET jazyk C# (inspirován C, C++, Java)

1998 – standard ISO/IEC 14882:1998 - **C++98**

2003 – standard ISO/IEC 14882:2003 - **C++03**

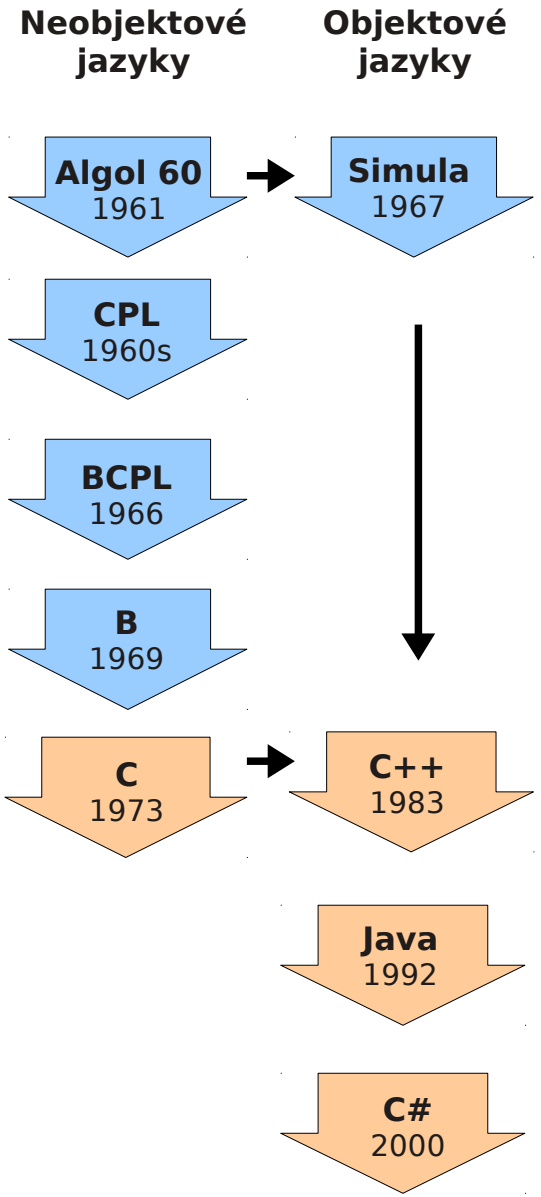
2011 – standard ISO/IEC 14882:2011 - **C++11**

Zkratky:

ANSI - American National Standards Institute

ISO - International Organization for Standardization

IEC - International Electrotechnical Commission



Učebnice jazyka C++

Základy Jazyka C++

Miroslav Virius: **Od C k C++**. KOPP, 2002. 232 s. ISBN 80-7232-110-2. (cca 199 Kč)

Petr Šaloun: **Programovací jazyk C++ pro zelenáče**. Neocortex, 2005. 252 s., ISBN 80-86330-18-4

Podrobná učebnice jazyka C++

Jesse Liberty, Bradley L. Jones: **Naučte se C++ za 21 dní.** Computer Press, 2007. 800 s., ISBN 978-80-251-1583-1 (790 Kč)

Pokročilejší témata jazyka C++

Miroslav Virius: Pasti a propasti jazyka C++. Computer Press, 2005. 376 s., ISBN 80-251-0509-1

Pokročilé techniky objektového programování

Rudolf Pecinovský: Návrhové vzory - 33 vzorových postupů pro objektové programování. Computer Press, 2007. 527 s., ISBN 80-251-1582-8

Steve McConnell: Dokonalý kód – umění programování a techniky tvorby software. Computer Press, 2006. 894 s., ISBN 80-251-0849-X

Kompilace programu v C++

- Zdrojové soubory s kódem v C++ mývají příponu *.cpp* (někdy se používají *.C*, *.CC*, *.cc*)
- Hlavičkové soubory se používají podobně jako v C, mívají příponu *.h*, *.hh*, *.hpp nebo .H, často* však nemají žádnou příponu
- Kompilace se provádí podobně jako při programování v C, místo překladače *gcc* se používí ***g++*** (pod Linuxem):

  g++ -o spustitelny_soubor zdrojovy_soubor.cpp

- V praxi je vhodné používat parametry -Wall a -pedantic

  Příklad: *g++ -Wall -pedantic -o myprog myprog.cpp*

- Pro psaní kódu programu lze použít např. editor *Kate*, který umožňuje automatické odsazování textu, barevné označení klíčových slov jazyka, vyznačení párování závorek a pod.

Make

- Nástroj *make* slouží k automatizovanému překladu programů
- 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

```
# Soubor muze obsahovat komentare, ktere zacinaji znakem hash
# Budeme kompilovat 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 hlavičkové soubory než v C)
- 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 //

```c
// Program v jazyce C
#include <stdio.h>


int main()
{
   printf("Hello, World!\n"));
   return 0;
}
```

```c
// Program v jazyce C++
#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`

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

int main()
{
  int a = 0;
  cout << "Jednoduchy text\n";
  cout << "Operatory " << "lze " << "zretezit";
  cin >> a;    // Ukazka nacteni celociselne hodnoty
}
```

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 << "Klasicke zakonceni radku trochu jinak" << '\n';
cout << "Alternativni zpusob zakonceni radku" << endl;
cout << 10;          // Vypis celeho cisla
cout << 23.175;      // Vypis desetinneho cisla
cout << a;           // Vypis hodnoty celociselne promenne
cout << d;           // Vypis hodnoty realne promenne
cout << c;           // Vypis hodnoty znakove promenne
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 číslo, 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;   // Do promenne a se nacte celociselna hodnota
cin >> d;   // Do promenne d se nacte realna hodnota
cin >> c;   // Do promenne c se nacte znak

// 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()
{
  string s1 = "Tady je nejaky text"; // Prvni zpusob inicializace
  string s2("Tady je nejaky text");  // Jiny zpusob inicializace
  string s3;         // Prazdny retezec
}
```

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 jiného řetězce k danému řetězci
- Operátory **==**, **!=**, **<**, **<=**, **>**, **>=** slouží k porovnávání řetězců (lexikografickému)
- Pro práci s řetězci nelze použít operátory **-**, *, /

```
int main()
{
  string s1, s2;

  s1 = "Skakal"; // Prirazeni retezcove konstanty do retezcove promennne
  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 main()
{
  int a = 3;
  string s1 = "Tady je nejaky text";

  cout << "Toto je retezec s1: " << s1 << endl << "a promenna a:" << a;
  cout << s1 + " pridany text";  // Operator + ma prioritu pred <<
  cout << (s1 + " na konec");    // Pouziti zavorek je vsak prehednejsi
}
```

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)

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

int main()
{
  string s;
  cout << "Zadej text: ";
  cin >> s;

  cout << "Nacteny text: " << s << endl;

  return 0;
}
```

Logický typ `bool`

- Na rozdíl od C existuje v C++ logický typ **`bool`**
- Proměnná typu bool nabývá hodnot **`true`** nebo **`false`**

```
// Ukazka pouziti typu bool a hodnot true a false

int a = 0, b = 0;
bool val = true;

// ... tady muze byt nejaky kod nastavujici hodnoty promennych a, b

if (a < b)
  val = false;

// Take lze napsat nasledujici, je to ale totez jako if (val)
if (val == true)
  a += b;

// Do logicke promenne lze ulozit vysledek logickeho vyrazu
val = a < b || b > 0;
if (val)
  a = b*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 dát
  `#include <g2.h>`
  `#include <g2_X11.h>`
- Čísla barev předávané funkci `g2_pen()` mají hodnoty 0 - 26, např. 0 bílá, 1 černá, 3 modrá, 7 zelená, 19 červená, 25 žlutá

```
int dev = 0;
dev = g2_open_X11(500, 500);  // Otevreni okna

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

cout << "Pro ukonceni programu stisknete Enter!" << endl;
cin.get();
cin.get();

g2_close(dev);
```

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)

Cvičení

1. Vytvořte program, který si od uživatele vyžádá dvě celočíslené hodnoty oddělené mezerou a vypíše jejich součet. **1 bod**
2. 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). **1 bod**
3. 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). **2 body**
4. Předchozí úlohu modifikujte tak, že barva nebude zadána formou čísla, ale jako text (black, blue, green, red, yellow).

   nepovinná, 1 bod