Základy jazyka C

Základy programování 1

Martin Kauer

Organizační záležitosti

- Konzultace
  - Pracovna 5.076
  - Středa 15:00 – 16:30
  - Emailem martin.kauer@upol.cz
- Web předmětu http://tux.inf.upol.cz/~kauer/index.php?content=var&class=ZP1
- Učební text http://phoenix.inf.upol.cz/~osicka/zp1-poznamky.pdf
- Sbírka úloh http://jazykc.inf.upol.cz/

Literatura

- Pavel Herout: Učebnice Jazyka C. Kopp, 2007.
- **Brian W. Kernighan, Dennis M. Ritchie: Programovací jazyk C, Computer Press, 2006.**
- Reek Kenneth: Pointers on C. Addison Wesley, 1997.
- Robert Sedgewick: Algorithms in C. Addison-Wesley Professional, 2001.
- Jeri R. Hanly, Elliot B. Koffman: Problem Solving and Program Design in C. Addison Wesley, 2006.
- Eric S. Roberts: Programming Abstractions in C. Addison Wesley, 1997.
- Eric S. Roberts: The Art and Science of C. Addison Wesley, 1994.
- a další…

Zápočet

- Alespoň 75% účast na seminářích.
- Zisk minimálně 25 bodů,
  - 2 domácí úkoly, 15 bodů za každý,
  - 10 bodů za práci na seminářích.

- Nebo možnost přímého získání zápočtu:
  - Druhý týden, programovací test na učebně.

Odevzdávání úkolů

- **Žádné plagiáty.**
- **Žádné komprimované soubory.** Jen zdrojové kódy s příponou **.c** nebo **.cpp**.
- **Rozumně formátovaný kód** - žádné dva metry dlouhé ify, pět příkazů na řádku apod.
- **Zdrojové kódy se musejí dát bez úpravy zkompilovat**.
- Programy musí splňovat **všechny povinné části** zadání.
- Deadline na odevzdání lze posunout pouze na základě opodstatněného důvodu a pouze po předchozí domluvě.
- **Po deadlinu nemusíte nic posílat,** nebude na to brán zřetel.

Jazyk C

- vytvořen 1972
- autor Dennis Ritchie
- aktuální standard C11 (2011)
- my si ale vystačíme s ANSI C (1990), příp. C99 (1999)
- inspirace pro mnoho soudobých jazyků
- vlastnosti:
  - nízkoúrovňový
  - překládaný (kompilovaný)
  - platformově nezávislý
  - staticky typovaný
  - procedurální (imperativní) paradigma

Ukázka zdrojového souboru

```c
#include <stdlib.h>
#include <stdio.h>

/*
 * Tento program nedělá nic jiného,
 * než že pozdraví celý svět.
 */

int main() { // hlavní funkce programu

    /* sem budeme zatím psát všechny příkazy */

    printf("Ahoj, světe!\n"); // napíše text do konzole

    return 0; // úspěšný konce programu
}
```

Vývojová prostředí

- MS Visual Studio / MS Visual C++ (MS Windows)
- Code::Blocks (multiplatformní)
- GNU Emacs (GNU/Linux)
- Xcode (OS X)
- Libovolný textový editor

Několik poznámek k syntaxi

- C rozlišuje velká a malá písmena (case sensitivity)
  `system ≠ System ≠ SYSTEM`
- C ignoruje většinu bílých znaků (mezer, tabulátorů, odřádkování)
  `(2*pi*r*(r+v)) = ( 2 * pi * r * (r+v) ) = (2 * pi * r * (r + v))`
- ale naopak
  `"Ahoj světe!" ≠ "Ahoj    světe!"`
  `int cislo; ≠ intcislo;`
- Příkazy jsou ukončeny středníkem

```
povrch = (2 * pi * r * (r+v));
printf("Ahoj světe!");
```

Proměnná

- Pojmenované místo v paměti
- Lze v ní uchovávat 1 hodnotu předem daného datového typu
- **Před prvním použitím je třeba nastavit její hodnotu - inicializace**
- Vytvoření proměnné:

```
typ identifikátor;
typ identifikátor = hodnota;
```

- Příklady:

```
int cislo;
double desetinne_cislo = 3.14;
char pismeno = 'A';
int logickaHodnota = 0;
```

```
int cislo1;
int cislo2 = cislo1 + 10; //CHYBA!
```

Identifikátor proměnné

- Jedinečný v rámci daného bloku / programu
- Může obsahovat písmena, číslice a podtržítko
- Musí začínat písmenem (nebo podtržítkem)
- **Identifikátor by měl být smysluplný**
  „Program častěji čteme než píšeme!“
  `obsah = pi * polomer * polomer;` vs. `xyz = ahoj * abc * abc;`
- Nedoporučuji používat háčky a čárky
- Doporučuje se underscore_case, případně camelCase

```
double polomer_kruznice;
double pi = 3.14;
int idHlavnihoUzivateleVAplikaci;
```

Základní datové typy

- `char` - obvykle pro znaky (1 byte, rozsah typicky 0 až 255)
- `int` - celé číslo (velikost přirozená pro danou platformu)
- `float` / `double` - desetinná čísla (s pohyblivou řadovou čárkou)
  (double má dvojnásobnou přesnost)
- Na typ int lze aplikovat kvalifikátory **long** a **short**:

```
short int prumerna_mzda = 25000;
int poslanecky_plat = 56000;
long int majetek_billa_gatese = 1180000000000;

short prumerna_mzda = 25000;
long majetek_billa_gatese = 1180000000000;
```

- Kvalifikátor long lze aplikovat na double:

```
long double velke_desetinne_cislo = 0.12345678902345678 93;
```

Znaménkové a neznaménkové typy

- Na celočíselné typy (int, char, ale i short, long) lze aplikovat modifikátory **signed** a **unsigned**
- `unsigned` - reprezentace pouze nezáporných čísel
- `signed` - čísla mohou být kladná i záporná
- Stejná velikost paměti
  - Rozsah signed char je od -128 do 127.
  - Rozsah unsigned char je od 0 do 255.

Logické hodnoty v C

- Logické hodnoty pravda a nepravda reprezentujeme v C čísly
- **Není zde speciální datový typ**
- Nula = nepravda
- Nenulové číslo (typicky 1) = pravda

Číselné konstanty

- Celočíselné konstanty zapisujeme číslicemi
- Záporná čísla odlišujeme znaménkem mínus
- **Implicitně mají typ int** (velké hodnoty long int)
- Desítkové: 14, 16U, -12, 5145L, 58UL
- Osmičkové: 07, 0245, 0123
- Šestnáctkové: 0xABC, 0x0, 0x12B
- U desetinných čísel používáme tečku (ne čárku)
- **Implicitně jsou tyto konstanty typu double**
- Desetinné: 12.34, 15.47F, 57.478L, 1e+6, 3.14e-3

Konstantní znaky a řetězce

- Znakové konstanty zapisujeme do apostrofů
- Znakové: 'a', '9', '\'', '\n'
- Řetězcové konstanty zapisujeme do úvozovek
- Řetězce: "Ahoj", "124", "\"Ahoj\"", "Prvni\nDruhy\nTreti\n"
- Textové řetězce jsou posloupnosti znaků ukončené '\0'

Výstup na obrazovku

- Pomocí funkce **printf** z knihovny stdio
- Obecně:
  printf(*řídící_řetězec*, *hodnota1*, *hodnota2*, ...);
- Příklady:

```
int x = 5, y = 3;
int sum = 8;
int cislo = 23;

printf("Součet je 8");
printf("Součet je %d", 8);
printf("Součet je %d", sum);
printf("Součet je %d", x + y);
printf("Součet je %d\t Součin je %d\n", x + y, x * y);
printf("Plán jsme splnili na 100 %%.");
printf("Dekadicky %d je oktalově %o a hexadecimálně %x.\n",
    cislo, cislo, cislo);
```

Vstup z klávesnice

- Pomocí funkce **scanf** z knihovny stdio
- Obecně:
  `scanf(řídící_řetězec, &promenna1, &promenna2, ...);`
- Příklady:

```
int cislo, cislo2;
unsigned int hexa_cislo;
unsigned int den, mesic, rok;

scanf("%d", &cislo);
scanf("%x", &hexa_cislo);
scanf("%d %d", &cislo, &cislo2);
scanf("%d.%d.%d", &den, &mesic, &rok);
```

Možnosti formátovacího řetězce

- %c – výpis nebo načtení znaku
- %d (%i) – celé číslo desítkově znaménkově
- %u – celé číslo desítkově neznaménkově
- %o – celé číslo osmičkově
- %x (%X) – celé číslo šestnáctkově (malá / velká písmena)
- %f – desetinné číslo (float a double při výpisu)
- %lf – načtení desetinného čísla typu double
- %e (%E) – desetinné číslo semilogaritmicky
- %g (%G) – jako %f nebo %e (%E) podle hodnoty čísla
- %s – textový řetězec

Cvičení

Úlohy k probraným tématům naleznete na webu jazykc.inf.upol.cz v části Základy jazyka C.

Časté problémy

- Pokud okno aplikace po kompilaci a spuštění okamžitě zmizí
  - Ve Visual Studiu pro spuštění použijte CTRL+F5 a ne jen F5
  - Nebo např. před návrat z vstupní funkce **`main`** vložit:

```
system("pause");
```

- Pokud se kompilace nezdaří s chybou ‘C4996’
  - Buď místo **`scanf`** použít **`scanf_s`**
  - Nebo na první řádek zdrojového souboru vložit:

```
#define _CRT_SECURE_NO_WARNINGS
```