Základy programování (v jazyku C)

EDU-IZP

Ing. Jakub Husa Ph.D.

Vysoké Učení Technické v Brně, Fakulta informačních technologií
Božetěchova 1/2. 612 66 Brno - Královo Pole

ihusa@fit.vut.cz

23. října 2025

Jazyk C je imperativní a kompilovaný programovací jazyk:

- Imperativní – program je tvořen posloupností příkazů definujících postup řešení nějakého problému (algoritmus).
- Kompilovaný – zdrojový kód programu nemůžeme spustit na přímo, ale musíme ho vždy nejprve přeložit na spustitelný soubor.

K programování potřebujeme vývojové prostředí obsahující editor a překladač:

- Editor – slouží k editování souboru do kterého budeme psát zdrojový kód.
- Překladač – slouží k přeložení zdrojového kódu na spustitelný soubor.

V EDU dílně si vystačíme s některým z on-line vývojových prostředí, například:

- https://www.onlinegdb.com/online_c_compiler
- https://www.tutorialspoint.com/compilers/online-c-compiler.htm
- https://onecompiler.com/c

Zdrojový kód programu se skládá z podprogramů zvaných funkce:

- Vykonávání programu vždy začíná hlavní funkcí jménem main (viz. cv03).
- Další funkce si můžeme napsat sami (viz. cv04), nebo si je můžeme naimportovat ze standardních knihoven jazyka C.
- Seznam standardních knihoven a jejich funkcí najdeme na referenci jazyka C.

Ukázkový program – Hello, World!:

```c
#include <stdio.h>                  //vlozeni knihovny pro vstup a vystup
                                    //prazdny radek (jen pro prehlednost)
int main()                          //hlavicka funkce main (zacatek programu)
{                                   //zacatek tela funkce main
    printf("Hello, World!\n");      //knihovni funkce pro vypis textu
    return 0;                       //prikaz ukonceni funkce (konec programu)
}                                   //konec tela funkce main
```

Dvojité lomítko (//) v jazyku C označuje jednořádkový komentář:

- Komentáře vysvětlují chování kódu, ale na jeho funkcionalitu nemají vliv.
- Víceřádkové komentáře ohraničujeme značkami (/*) a (*/).

Proměnné

Data v programu ukládáme do proměnných:

- Každá proměnná musí mít určen nějaký datový typ a identifikátor.
- Datový typ – udává rozsah možných hodnot a dostupné operace.
- Identifikátor – označuje místo v paměti na kterém jsou data uložena.

Identifikátory musejí začínat písmenem nebo podtržítkem (a-z, A-Z, _) a mohou pokračovat libovolnou kombinací písmen, číslic a podtržítek (a-z, A-Z, 0-9, _):

- V jednom bloku kódu nelze mít dvě proměnné se stejným identifikátorem.
- Identifikátor se nesmí shodovat s žádným z klíčových slov.
- Identifikátory začínající podtržítkem mají speciální význam (nepoužívat)!
- Celá čísla v rozsahu od -2^{31} do $2^{31}-1$ ukládáme do datového typu – `int` – (integer).

```
int x;      //cele cislo jmenem "x"
int x1;     //cele cislo jmenem "x1"
int x_1;    //cele cislo jmenem "x_1"
int x 1;    //CHYBA -- jmeno promenne nesmi obsahovat mezery
int 1x;     //CHYBA -- jmeno promenne nesmi zacinat cislici
int int;    //CHYBA -- jmeno promenne nesmi byt klicovym slovem
int x;      //CHYBA -- promenna jmenem "x" jiz existuje
int _x;     //VAROVANI -- jmeno promenne zacina podtrzitkem
```

Hodnotu proměnné definujeme pomocí operátoru přiřazení (=):

- Hodnotu proměnné můžeme definovat už při jejím vytvoření.

```
int a = 10;             //vytvor cele cislo "a" s hodnotou 10
a = 20;                 //hodnotu promenne "a" zmen na 20
```

Hodnoty zpracováváme pomocí výrazů:

- Celá čísla počítáme pomocí aritmetických operátorů (+, -, *, /, %).

```
int b = 1 + 1;          //b =  2 //scitani
int c = 2 - 3;          //c = -1 //odcitani
int d = b * c;          //d = -2 //nasobeni
int e = 5 / 2;          //e =  2 //celociselne deleni
int f = 5 % 2;          //f =  1 //zbytek po celociselnem deleni (modulo)
```

Pořadí aplikace operátorů se řídí jejich prioritou:

- Změnu pořadí můžeme vynutit pomocí kulatých závorek (( )).

```
int g =  1 + 2  * 3; //g =  7 //nasobeni ma vetsi prioritu nez scitani
int h = (1 + 2) * 3; //h =  9 //nejvetsi prioritu maji zavorky
```

Vstup a výstup programu

Vstup a výstup provádíme funkcemi z knihovny stdio.h:

- Knihovny vkládáme na začátku souboru direktivou – `#include` –
- Vstupy načítáme funkcí scanf, výstup vypisujeme funkcí printf.

Prvním parametrem funkce scanf je formátovaný textový řetězec se značkou načítaného datového typu, druhým parametrem je adresa načítané proměnné:

- Celá čísla označujeme symbolem (`%i`).
- Adresu proměnné (viz. cv06) získáme operátorem reference (&, zkratka AltGr+C).

```
int x;                  //vytvarime promennou jmenem "x"
scanf("%i", &x);        //ze vstupu do ni nacitame hodnotu
```

Funkce printf vypisuje formátovaný řetězec, který značek může obsahovat více:

- Značky se při výpisu nahradí čísly, v pořadí ve kterém jsme je funkci předali.
- Řádky ukončujeme speciálním znakem nového řádku (`\n`, viz. cv03).

```
int y = 10;             //vytvarime cele cislo "y" s hodnotou 10
int z = 20;             //vytvarime cele cislo "z" s hodnotou 20
printf("%i + %i = %i\n", y, z, y+z); //vypisujeme "10 + 20 = 30"
```

Vyzkoušejte si:

- Vytvořte si dvě celá čísla (výšku a šířku) a ze vstupu do nich načtěte hodnoty.
- Vypište obsah a obvod odpovídajícího obdélníku.

$$S = a * b \qquad O = 2 * (a + b)$$

Například:

- (2, 3) => Obsah je 6
  => Obvod je 10
- (10, 20) => Obsah je 200
  => Obvod je 60

Nápověda:

- Pokud chceme uživatele vyzvat k zadání hodnoty, musíme použít funkci printf.
- V řetězci funkce scanf nepoužíváme speciální znak konce řádku (`\n`).