Jazyk C

Začátky s jazykem C

Mgr. Markéta Trnečková, Ph.D.

Palacký University, Olomouc

Základní datové typy

- **celočíselné**: `int`, `short int` (zkráceně `short`) a `long int` (zkráceně `long`)
- velikost `int`: dána počítačem
- nový standard: celočíselné typy s přesně daným počtem bitů.
- `sizeof(short int)` $\leq$ `sizeof(int)` $\leq$ `sizeof(long int)`
- **znaky**: `char`
- velikost `char`: 1 byte
- **desetinné**: `float` (jednoduchá přesnost), `double` (dvojitá přesnost) a `long double` (čtyřnásobná přesnost)
- `sizeof (float)` $\leq$ `sizeof(double)` $\leq$ `sizeof(long double)`
- celočíselné datové typy mohou být `signed` (znaménkové) nebo `unsigned` (neznaménkové)
- `sizeof(unsigned int)` $=$ `sizeof(signed int)`

Uživatelské datové typy

```
typedef definice_typu nazev_noveho;
```

Výčtový typ

```
enum nazev {
    HODNOTA1, HODNOTA2, ..., HODNOTAN};
```

- definujeme datový typ `enum nazev`

```
enum nazev {
    HODNOTA1 = hodnota1,
    HODNOTA2 = hodnota2,
    ...,
    HODNOTAN = hodnotan};

typedef enum {
    HODNOTA1, HODNOTA2, ..., HODNOTAN
    } NAZEV;
```

Výčtový typ

Example

```
typedef enum {
    FALSE, TRUE
} BOOLEAN;
```

Literály

- **celočíselné**:
  - desítková soustava: 1, 42, 123
  - osmičková soustava: 01, 052, 0173
  - šestnáctková soustava: 0x1, 0x2a, 0X7B
- implicitně `int`
- přípony: L (l) – `long`, U (u) – `unsigned` 123LU
- záporné hodnoty -
- **reálné**:
  - s desetinnou tečkou: .123, 123., 1.23
  - exponenciální tvar: 1e23, 12E3
- implicitně `double`
- přípony: F (f) – `float`, L (l) – `long double`
- **znaky**:
  - apostrofy: 'a', '*', '2'
  - escape sekvence: '\n', '\'', '\0'
  - pomocí čísel: '\ddd', '\xhh'
- **řetězce**:
  - v uvozovkách: ''Ja jsem \'' retezcova \'' hodnota \n''

Proměnné

```
typ jmeno;

int cislo_1, cislo_2, cislo_3;
```

Example

```
int i;
int main()
{
    int j;
    return 0;
}
```

Operátory: priority operátorů

priorita	operátor	význam	asociativita
1	`()`	volání funkce	zleva doprava
	`[]`	index do pole	
	`->`	přístup k prvku struktury	
	`.`	přístup k prvku struktury	
2	`!`	negace logického výrazu	zprava doleva
	`~`	jedničkový doplněk	
	`++`	inkrementace	
	`--`	dekrementace	
	`+`	unární plus (+1)	
	`-`	unární mínus (-1)	
	`(typ)`	přetypování	
	`*`	dereference	
	`&`	reference (adresový operátor)	
	`sizeof`	velikost typu	
3	`*`	součin	zleva doprava
	`/`	dělení	
	`%`	dělení modulo	
4	`+`	součet	zleva doprava
	`-`	rozdíl	
5	`<<`	bitový posun doleva	zleva doprava
	`>>`	bitový posun doprava	

Operátory: priority operátorů

priorita	operátor	význam	asociativita
6	< <= > >=	menší než menší nebo rovno než větší než větší nebo rovno než	zleva doprava
7	== !=	rovnost nerovnost	zleva doprava
8	&	bitový součin	zleva doprava
9	^	exkluzivní bitový součin	zleva doprava
10	|	bitový součet	zleva doprava
11	&&	logický součin	zleva doprava
12	||	logický součet	zleva doprava
13	? :	ternární (podmínkový) operátor	zprava doleva
14	= >>= <<= += -= *= /= %= &= |= ^=	přiřazovací operátory	zprava doleva
15	,	operátor čárky	zleva doprava

Operátory: priority

Example

```
a = 2;
b = 4;
c = a++ - b / 2;
```

Example

```
a = 2;
b = 4;
c = a + 2 - b;
```

Operátory: operátor přiřazení

```
l_value = r_value;
```

Example

```
int a = 5, b = 6, c, d;

c = 7;
c = b;
b = 3;
c = c - b;

/* nekolikanasobne prirazeni
   a = (b = (c - 2));
*/
a = b = c = 2;
```

Operátory: aritmetické

- **unární**: `+`, `-`
- **binární**: `+`, `-`, `*`, `/`, `%`

Operátory: aritmetické

Example

```
int i = 10, j = 4, k;
float f, g = 10.0, h;

/* k i f jsou rovny 2, protoze i a j jsou typu int, podil je
 typu int. Vysledek dostaneme odstranenim  desetinne casti */
k = i / j;
f = i / j;

/* f bude mit hodnotu 2.5, protoze g je float,
 podil je take typu float. */
f = g / j;

/* k bude rovno 2, ztrati se desetinna cast. */
k = f;

/* h bude rovno 0.5. Proc? */
h = f - k;
```

Operátory - ++ a --

- operand musí být `l_value`
- **inkrementace**: `a++`, `++a`
- **dekrementace**: `a--`, `--a`

Example

```c
#include <stdio.h>

int main()
{
    int a = 10;
    int b = 10;
    a = b++;
    printf("hodnota a je %d a hodnota b je %d \n", a, b);
    a = ++b;
    printf("hodnota a je %d a hodnota b je %d \n", a, b);
    return 0;
}
```

Operátory - přiřazovací

- += přiřazení sčítání, zkratka pro
  `l_value = l_value + r_value`
- −= přiřazení odečítání, zkratka pro
  `l_value = l_value - r_value`
- *= přiřazení násobení, zkratka pro
  `l_value = l_value * r_value`
- /= přiřazení dělení, zkratka pro
  `l_value = l_value / r_value`

Operátory - operátor čárky

```
vyraz1, vyraz2
```

Example

```
int i = 2, j = 3; /* Toto neni operator carky */

j = (i++, i - j);
```

Typová konverze

- **implicitní** - aplikace operátorů
- **explicitní** - `(typ) vyraz`

Example

```
int i = 10, j = 4, k;
float f;
/* k i f = 2, protoze i a j jsou typu int, podil je typu int.
Vysledek je roven celociselnemu deleni. */
k = i / j;
f = i / j;
/* f bude mit hodnotu 2.5, protoze citatel je float,
 podil je take typu float.
   k bude rovno 2, i pres to, ze vysledek prave strany je
   2.5 . k je typu int, dochazi zde k implicitni konverzi.
   Desetinna cast cisla se odstrani. */
k = (float)i / j;
f = (float)i / j;
```

Vstup a výstup

- `stdio.h`
- **Výstup**:
  - `putchar()`
  - `printf()`
- **Vstup**:
  - `getchar()`
  - `scanf()`

printf()

```
printf(format, h1, ..., hn);
```

- **Formátovací instrukce**:
  `%[příznak][šířka][.přesnost][modifikátor]konverze`
  - `%c` znak
  - `%i` číslo `int`
  - `%d` číslo `int` v desítkové soustavě (znaménkově)
  - `%u` číslo `int` v desítkové soustavě (neznaménkově)
  - `%x` číslo `int` v šestnáctkové soustavě (malými písmeny, např.: `1a2b`)
  - `%X` číslo `int` v šestnáctkové soustavě (velkými písmeny, např.: `1A2B`)
  - `%o` číslo `int` v osmičkové soustavě
    (Přidáním `l` před znaky `d`, `u`, `x`, `X`, `o` pracujeme s číslem typu `long`)
  - `%f` desetinné číslo typu `float`
    (Přidáme-li před `f` znak `l` (`%lf`), vypisujeme desetinné číslo typu `double`. Přidáme-li `L` (`%Lf`), vypisujeme desetinné číslo typu `long double`.)
  - `%e` desetinné číslo v exponenciálním tvaru
  - `%s` řetězec

printf()

Example

```c
printf("Soucet je %d\n", i + j);
/* Vypise: Soucet je 11 (v zavislosti na hodnotach i a j) a odradkuje */

printf("Soucet je %d\tSoucin je %d\n", i + j, i * j);
/* Vypise: Soucet je 11   Soucin je 28 a odradkuje */

printf("Potrebuji %s jeden priklad.", "vymyslet");
/* Vypise: Potrebuji vymyslet jeden priklad. */
```

Úkol

Example

Vytvořte proměnnou `znak` typu `char`, přiřaďte jí nějakou hodnotu. Pomocí funkce `printf()` vypište tuto proměnnou a ASCII hodnotu příslušnou této hodnotě. Například pro proměnnou `znak = 'A'` program vypíše: `Znak 'A' ma ASCII hodnotu 65.`

getchar()

Example

```c
#include <stdio.h>

int main()
{
    int znak;
    znak = getchar();
    /* Vypis nacteneho znaku.*/
    putchar(znak);
    return 0;
}
```

scanf()

```
scanf(format, p1, ..., pn);
```

Example

```
#include <stdio.h>

int main()
{
    int i;
    /* Nacteni celeho cisla a ulozeni do promenne i. */
    scanf("%d", &i);
    return 0;
}
```

Cvičení

1. Napište program, který vypočte hodnotu matematického výrazu

$$\frac{3}{2} + 12 + \frac{5 * 5 - 2 * 2}{6}$$

   a výsledek vypíše na obrazovku. Správnost výsledku ověřte výpočtem.

2. Pro `a = 2, b = 2, c = 0, d = 1, e = 4` odhadněte výsledné hodnoty výrazů a zkontrolujte tím, že napíšete program, který výsledky spočítá.
   - `a++ / b-- * d++`
   - `b-- * c++ + e`
   - `-b + --c`
   - `++a + a--`
   - `c / a * d++ / c--`
   - `a %= b = d = 1 + e / 2`

3. Napište program, který načte stranu čtverce a vypíše jeho obvod a obsah.

4. Napište program, který načte velké písmeno, a vypíše jej jako malé. Využijte znalosti ASCII tabulky.

5. Načtěte ze vstupu trojciferné číslo a poté vypište jeho první a poslední číslici.

Cvičení

6. Načtěte ze vstupu reálné číslo a vypište jeho celou část.
7. Napište program, který na obrazovku vypíše přesně text:

```
Uspesnost predmetu "Diskretni struktury 1", ktery ma zkratku
KMI/DISK1, je 50%.
```