

Jazyk C

Začátky s jazykem C

Mgr. Markéta Trnečková, Ph.D.



Palacký University, Olomouc



- **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) ≤ sizeof(int) ≤ sizeof(long int)`
- **znaky**: `char`
- velikost `char`: 1 byte
- **desetinné**: `float` (jednoduchá přesnost), `double` (dvojitá přesnost) a `long double` (čtyřnásobná přesnost)
- `sizeof(float) ≤ sizeof(double) ≤ sizeof(long double)`
- celočíselné datové typy mohou být `signed` (znaménkové) nebo `unsigned` (neznaménkové)
- `sizeof(unsigned int) = sizeof(signed int)`



```
typedef definice_typu nazev_noveho;
```

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

- definujeme datový typ enum nazev

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

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



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"

```
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	zleva doprava
	<=	menší nebo rovno než	
	>	větší než	
	>=	větší nebo rovno než	
7	==	rovnost	zleva doprava
	!=	nerovnost	
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



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;
```



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

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 odstranem 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

```
#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;  
}
```



- += 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`



vyraz1 , vyraz2

Example

```
int i = 2, j = 3; /* Toto není operátor čárky */  
j = (i++, i - j);
```


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

Example

```
int i = 10, j = 4, k;  
float f;  
/* k i f = 2, protože i a j jsou typu int, podíl je typu int.  
Výsledek je roven celocíselnému dělení. */  
k = i / j;  
f = i / j;  
/* f bude mít hodnotu 2.5, protože číselník je float,  
podíl je také typu float.  
k bude rovno 2, i přes to, že výsledek pravé strany je  
2.5. k je typu int, dochází zde k implicitní konverzi.  
Desítná část čísla se odstraní. */  
k = (float)i / j;  
f = (float)i / j;
```



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

```
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 1 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

Example

```
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. */
```

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.

Example

```
#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;
}
```

- 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 trojčíferné číslo a poté vypíše jeho první a poslední číslici.



- 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%.