KATEDRA INFORMATIKY
PŘÍRODOVĚDECKÁ FAKULTA
UNIVERZITA PALACKÉHO V OLOMOUCI

Typy, operátory a výrazy

2. cvičení

Jiří Zacpal

KMI/ZP1 – Základy programování 1

Základní datové typy

- Celočíselné datové typy
  - znaménkové (signed)
  - neznaménkové (unsigned)

```
char, short, int, long
```

- Reálné číselné typy – s pohyblivou řádovou čárkou

```
float, double, long double
```

- Textové řetězce nejsou základní datový typ

Celočíselné datové typy

- `char` (běžně pro uložení znaku)
- `short int` (zkráceně `short`)
- `int`
- `long int` (zkráceně `long`)
- reprezentace logických hodnot typem `int`
  - nula = nepravda
  - nenulové číslo = pravda

Reálné číselné datové typy

- `float`
- `double`
- `long double`

Textové řetězce

- nejsou v jazyku C základní datový typ
- pracuje se s nimi jako s poli typu `char`
  nebo jako s ukazateli na typ `char`
- tvoří je posloupnost znaků ukončených nulovým znakem (`'\0'`)

Konstanty

- Celočíselné
  - Desítkové: `10, 1234589, 15u, 1366L, -56, 42LU`
  - Osmičkové: `07, 0124, 073`
  - Šestnáctkové: `0xA1B, 0x0, 0X1d3, 0xac`
  - Znakové: `'a', '*', '3', '\''`
- Reálné

  `123.56, 15. , .86, -13.2f, 1.23F, 23.128L`

  `2.1425e-3, 2.1e+4L`

Konstantní výrazy

- výraz obsahující pouze konstanty
- lze je vyčíslit během kompilace

```
#define MAXRADEK 100
char radek[MAXRADEK+1];
```

Řetězcové konstanty

```
"Ahoj Pepo!"
```

- Speciální znaky:
  `\n, \t, \a, \\, \", \', \0`
- Pomocí ASCII hodnoty
  - osmičkové: `\OOO`
  - šestnáctkové: `\xHH` nebo `\XHH`
- Příklad:
  `"\tMáma volá: \n\"Ahoj Pepo\041\""`

Proměnné

- definice proměnné

  ```
  typ indentifikátor = hodnota;
  ```

- identifikátor
  - je tvořen písmeny, číslicemi a znakem podtržítko
  - měl by začínat písmenem
- proměnné musí být před použitím deklarovány
- příklady:

  ```
  int moje_cislo;
  char *str = "Ahoj Světe!";
  char z = '*';
  int c1, c2=3, c3;
  ```
  (definice více proměnných)

Operátory - základní pojmy

- Priorita
  - udává pořadí, ve kterém se operace provádějí
  - pořadí lze pozměnit použitím kulatých závorek (závorky je ovšem dobré používat i tam, kde nejsou nutné, protože zvyšují přehlednost kódu)
- Asociativita
  - udává „směr“, ve kterém se vyhodnocují operace se stejnou prioritou (zleva nebo zprava)
  - lze opět ovlivnit použitím závorek
- Arita

Přehled operátorů jazyka C

Pr.	Operátory	Asoc.	Arita
1	`() [] -> .`	Zleva	
2	`! ~ ++ -- + - (`*typ*`) * & sizeof`	Zprava	Unární
3	`* / %`	Zleva	Binární
4	`+ -`	Zleva	Binární
5	`<< >>`	Zleva	Binární
6	`< > <= >=`	Zleva	Binární
7	`== !=`	Zleva	Binární
8	`&`	Zleva	Binární
9	`^`	Zleva	Binární
10	`|`	Zleva	Binární
11	`&&`	Zleva	Binární
12	`||`	Zleva	Binární
13	`? :`	Zprava	Ternární
14	`= += -= *= /= %= >>= <<= &= |= ^=`	Zprava	Binární
15	`,`	Zleva	Binární

Přiřazení

- je operátor (vyhodnocuje se na přiřazovanou hodnotu)
- obecný tvar: L-hodnota = výraz
- L-hodnota (L-value) je výraz, kterému odpovídá adresa v paměti (např. identifikátor proměnné)
- další operátory přiřazení: `+=, -=, *=, /=, %=, ...`
  (např. `x += 2` odpovídá `x = x + 2`)
- příklady:

```
i = j = k = 2;
x = 2 + (y = 3 * z * z + 2 * y);
```

Aritmetické operátory 1/2

- unární plus a mínus

  ```
  např. +2,  -cislo
  ```

- sčítání a odčítání (binární plus a mínus)

  ```
  i+3, a-b
  ```

- dekrementace a inkrementace

  ```
  i--, ++c
  ```

  - použití prefixové a postfixové

    ```
    int i = 1, j, k;
    j = i++;
    ```
    (j má hodnotu 1, i má hodnotu 2)

    ```
    k = ++i;
    ```
    (k má hodnotu 3, i má hodnotu 3)

Aritmetické operátory 2/2

- násobení, dělení, modulo

```
2*a, 4/3, 5%b
```

- typ výsledku závisí na typech operandů:

```
celé */% celé = celé
celé */ reálné = reálné
realne */ cele = reálné
reálné */ reálné = reálné
```

Logické operátory a porovnání

- logický součin, logický součet, negace

```
a&&b, a||b, !a
```

- porovnání
  (je menší, je větší, je menší nebo rovno,
  je větší nebo rovno, rovná se, nerovná se)

```
a<1, 2>b, 2<=3, a>=b, a==1, b!=2
```

Příklad 1

```
#include<stdio.h>

main()
{
        char r[]="Ahoj svete";
        char c ='e';

        int i,j;

        for(i=j=0;r[i]!='\0';i++)
                if (r[i]!=c)
                        r[j++]=r[i];
        r[j]='\0';

        printf("%s\n",r);

}
```

Implicitní přetypování

- Jsou-li operandy různých typů
- „Užší“ typ se převede na „širší“
- Výrazy nedávající smysl (float v indexu pole) nejsou povoleny
- Převod „širšího na „užší“ je povolen, může ale vyvolat varovnou hlášku
- Implicitní přetypování proběhne:
  1. Pokud nemají operandy stejný typ
  2. V přiřazovacích příkazech se typ výrazu vpravo konvertuje na typ levé strany, což je i typ výsledku
  3. Při předávání argumentů funkcí

Explicitní (požadované, vynucené) přetypování

- Obecně: (nový_typ)výraz

  příklad 1:

```
int i = 10;
double f;
f = sqrt((double)i);
```

  příklad 2:

```
float f = 10.23;
int i;
i = (int)f;
```

- Operátor sizeof
  - vrací velikost datového typu nebo proměnné v bytech

  např. `sizeof(int), sizeof(moje_cislo)`

Příklad 2

```
#include<stdio.h>
#include<math.h>

main()
{
        int a,b,c,d;
        double x1,x2;

        printf("Zadej koeficienty a,b,c:");
        scanf("%d%d%d",&a,&b,&c);

        d=(int)pow((double)b,2)-4*a*c;
        if(d>0)
        {
                x1=(-b+sqrt((double)d))/2;
                x2=(-b-sqrt((double)d))/2;
                printf("Koreny jsou %.2f a %.2f\n",x1,x2);
        }
        else
                printf("Rovnice nema reseni v oboru realnych cisel.");
}
```

Podmínkový operátor

- syntaxe:

```
podmínka ? výraz_1 : výraz_2
```

- pokud je podmínka splněna, pak se vyhodnotí na výraz_1 jinak na výraz_2
- používá se zřídka, většinou jej lze nahradit podmínkovým příkazem if, který je přehlednější (viz příští seminář)

Operátor čárka

- syntaxe:

```
výraz_1, výraz_2
```

- při vyhodnocování operátoru se nejprve vyhodnotí výraz_1 a pak výraz_2; jako výsledek vyhodnocení operátoru se pak použije výsledek vyhodnocení výrazu výraz_2
- používá se téměř jen v řídících řádcích cyklů (viz seminář 4)

Bodovaný úkol

- Napište v jazyku C program, který ověří, jestli lze sestrojit trojúhelník se zadanými velikostmi stran. Vstupem programu jsou velikosti stran trojúhelníku. V příkladu použijte podmínkový operátor.
- **Příklad výstupu:**

```
Zadejte stranu a: 8
Zadejte stranu b: 5
Zadejte stranu c: 20
Trojuhelnik nelze sestrojit.
```

- **Povolené knihovny:** stdio.h