Desetinná čísla, cykly a pole

IZP-cv02

Ing. Jakub Husa

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

23. září 2024

Desetinná čísla

Desetinná čísla

Desetinná čísla ukládáme do proměnných s pohyblivou řádovou čárkou:

- Zápis desetinných čísel v paměti počítače má omezenou přesnost.
- Jako datový typ používáme buď – `float` – (floating point, přesnost cca 8 číslic), nebo – `double` – (double precision floating point, přesnost cca 16 číslic).
- Ve zdrojovém kódu desetinná čísla vždy označujeme desetinnou tečkou.

```
1  int x = 1;          //cele cislo
2  float y = 1.0;      //desetinne cislo
3  double z = 1.0;     //desetinne cislo s dvojnasobnou presnosti
```

Desetinná čísla načítáme funkcí scanf se značkou %f (float) nebo %lf (double), a vypisujeme funkcí printf vždy se značkou %f (float i double):

```
4  scanf("%i", &x);    //nacitame int
5  scanf("%f", &y);    //nacitame float
6  scanf("%lf", &z);   //nacitame double
7
8  printf("%i\n", x);  //vypisujeme int
9  printf("%f\n", y);  //vypisujeme float nebo double
10 printf("%f\n", z);  //vypisujeme float nebo double
```

Problémy s desetinnými čísly

Proč program vypisuje hodnotu 2.0 místo 2.5?

```
1  float x = 5 / 2;     //vytvarime cislo a prirazujeme do nej hodnotu
2  printf("%f\n", x);   //vypisujeme desetinne cislo
```

Proč jsou hodnoty proměnných Y a Z různé?

```
3  float y = 0.7;       //y = 0.7
4  float z = 0.6;       //z = 0.6
5  z = z + 0.1;         //z = 0.6 + 0.1
6
7  if (y == z)          //pokud je "y" rovno "z"
8  {
9      printf("%f se rovna %f\n", y, z);    //vypis
10 }
11 else                 //jinak
12 {
13     printf("%f se nerovna %f\n", y, z); //vypis
14 }
```

Desetinná čísla jsou vypisována s přesností na šest desetinných míst:

- Formát výpisu můžeme změnit úpravou použité formátovací značky.

```
1  float a = 0.1;              //desetinne cislo
2  double b = 0.1;             //desetinne cislo s dvojnasobnou presnosti
3  printf("%.20f\n", a);       //vypis s presnosti na 20 mist
4  printf("%.20f\n", b);       //vypis s presnosti na 20 mist
```

Pro výpočet složitých matematických operací (sqrt, pow, sin, cos, log, ...) můžeme použít funkce z matematické knihovny math.h:

- Programy které tuto knihovnu používají překládáme s parametrem -lm.
- `gcc main.c -o main -lm`

```
5  #include <math.h>           //vkladame kihovnu matematickych funkci
6  int main()                  //hlavicka funkce main (zacatek programu)
7  {
8      float x = 3.0;          //vytvarime desetinne cislo x = 3.000000
9      float y = sqrt(x);      //pocitame odmocninu         y = 1.732051
10     float z = pow(x, 2.0); //pocitame druhou mocninu    z = 9.000000
11     return 0;               //prikaz ukonceni funkce (konec programu)
12 }
```

Vyzkoušejte si:

- Ze vstupu načtěte tři desetinná čísla s dvojnásobnou přesností (a, b, c).
- Spočítejte kořeny kvadratické rovnice definované vzorcem:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Například:

- (0.0, 1.0, 1.0) => Deleni nulou, rovnice nema reseni
- (1.0, 1.0, 1.0) => Odmocnina ze zaporneho cisla, rovnice nema reseni
- (1.0, 2.0, 1.0) => x1 = -1.000, x2 = -1.000
- (1.0, 3.0, 2.0) => x1 = -1.000, x2 = -2.000

Cykly

Cyklus typu while

Cyklus je řídící struktura umožňující opakování nějaké části kódu:

- Jazyk C umožňuje používat tři typy cyklů (while, do-while, a for).
- Cyklus se skládá z jednoho nebo nebo více klíčových slov, hlavičky (v kulatých závorkách) a těla (ve složených závorkách).

Cyklus s podmínkou na začátku začíná klíčovým slovem – `while` – (dokud):

- Hlavička obsahuje podmínku, při jejímž splnění se cyklus bude opakovat.
- Tělo obsahuje příkazy které se mají opakovat dokud je podmínka pravdivá.

```
1  int i = 0;                  //vytvarime cele cislo "i" s hodnotou nula
2
3  while (i < 10)              //dokud je "i" mensi jak 10
4  {
5      printf("%i\n", i);      //vypis hodnotu "i"
6      i = i + 1;              //hodnotu promenne "i" zvys o jednicku
7  }
```

Cyklus typu do-while

Cyklus s podmínkou na konci začíná klíčovým slovem – `do` – (dělej):

- Pak má tělo s příkazy které se mají opakovat dokud je podmínka pravdivá.
- A na konci je slovo – `while` – (dokud), hlavička s podmínkou a středník!

```
1  int i = 0;                  //vytvarime cele cislo "i" s hodnotou nula
2  do                          //delej
3  {
4      printf("%i\n", i);      //vypis hodnotu "i"
5      i = i + 1;                 //hodnotu promenne "i" zvys o jednicku
6  }
7  while (i < 10);             //dokud je "i" mensi jak 10
```

Pro změnu hodnoty proměnné můžeme použít zkrácený zápis:

- Pro aritmetické operace můžeme použít rozšířená přiřazení (`+=`, `-=`, `*=`, `/=`, `%=`).
- Pro inkrementaci a dekrementaci můžeme použít operátory (`++`) a (`--`).

```
8  int x = 10; //cele cislo "x" s hodnotou 10
9  x += 5;      //x = x + 5; //pricitani
10 x *= 2;      //x = x * 2; //nasobeni
11 x++;         //x = x + 1; //inkrementace
12 x--;         //x = x - 1; //dekrementace
```

Vyzkoušejte si:

- Ze vstupu načtěte jedno celé číslo (X), a dokud je kladné dělte ho dvěma a vypisujte jeho hodnotu.
- Ze vstupu načítejte desetinná čísla tak dlouho dokud nebyla zadána nula a pak vypište jejich součet.

Například:

- (10) => x = 5
  x = 2
  x = 1
  x = 0
- (1.1, 2.2, 3.3, 0.0) => Soucet cisel je 6.6

Cyklus typu for

Cyklus se známým počtem opakování začíná klíčovým slovem – `for` – (pro):

- Hlavička obsahuje inicializaci, podmínku a iteraci, oddělené středníky.
- Tělo obsahuje příkazy které se mají opakovat dokud je podmínka pravdivá.

```
1  //inicializace se provede pred prvnim provedenim tela
2  //podminka     se overuje pred kazdym provedenim tela
3  //iterace      se provede po   kazdem provedeni  tela
4  for (int i=0; i<10; i++) //for (inicializace; podminka; iterace)
5  {
6      printf("%i\n", i);   //vypis hodnotu promenne "i"
7  }
```

Proměnná vytvořená uvnitř podmínky nebo cyklu na jejich konci přestává existovat:

```
8  int soucet = 0;             //vytvarime cele cislo "soucet"
9  for (int i=1; i<=10; i++) //pro "i" od 1 do 10 (vcetne)
10 {
11     soucet += i;            //k promenne "soucet" pricitame cislo "i"
12 }
13 printf("Suma prvnich %i celych cisel %i\n", i, soucet); //vypis
14 //CHYBA -- promenna "i" vytvorena v hlavicce cyklu for uz NEEXISTUJE!
```

Vyzkoušejte si:

- Ze vstupu načtěte dvě celá čísla (X a Y).
- Vypište všechny hodnoty v rozsahu od menšího do většího z nich (včetně).

Například:

- (5, 10) => 5 6 7 8 9 10
- (10, 5) => 5 6 7 8 9 10
- (5, 5) => 5

Pole

Vytvoření a inicializace pole

Pole umožňuje vytvořit skupinu proměnných stejného datového typu:

- Velikost (počet prvků) pole musíme určit už při jeho vytvoření a nemůžeme ji později změnit (dynamická pole budeme probírat až na 7. cvičení).

```
1 int a[3];      //pole tri celych cisel "a"
2 float b[4];    //pole ctyr desetinnych cisel "b"
3 double c[5];   //pole peti desetinnych cisel s dvojnasobnou presnosti "c"
```

Počáteční hodnotu prvků můžeme inicializovat pomocí složených závorek ({ }):

- Pokud závorky použijeme, hodnoty zbývajících prvků budou vynulované.
- Pokud závorky nepoužijeme, hodnoty všech prvků budou nedefinované.

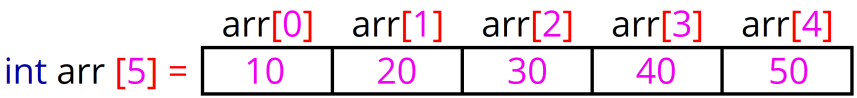
```
4 int x[5] = {10, 20, 30}; //pole peti cisel s hodnotami {10, 20, 30}
5 int y[5] = {40};         //pole peti cisel s hodnotami {40,  0,  0}
6 int z[5];                //pole peti cisel s nedefinovanymi hodnotami
7
8 z = {50, 60, 70};        //CHYBA -- slozene zavorky muzeme pouzit
9                          //pouze pri INICIALIZACI!
```

Indexace prvků pole

K jednotlivým prvkům pole přistupujeme indexem a hranatými závorkami ([ ]):

- Prvky pole jsou vždy indexovány od nuly.

```
1  int arr[5] = {10, 20, 30, 40, 50};   //pole peti celych cisel "arr"
```



Hodnoty prvků můžeme načítat funkcí scanf a vypisovat funkcí printf:

- Hranaté závorky ([ ]) mají vyšší prioritu než operátor reference (&).

```
6  int x[3];                              //pole tri celych cisel "x"
7  for (int i = 0; i < 3; i++)            //pro "i" od 0 do 2
8  {
9      scanf("%i", &x[i]);                //na index "i" pole "x" nacti cislo
10     printf("x[%i] = %i\n", i, x[i]); //vypis index a hodnotu prvku
11 }
```

Jazyk C **NEKONTROLUJE** platnost používaných indexů:

- Přístup na neplatný index způsobí neplatný přístup do paměti.
- Chyba může způsobit ztrátu dat nebo havárii celého programu!

```
1  float y[3] = {1.1, 2.2, 3.3};          //pole tri desetinnych cisel "y"
2
3  for (int i = 1; i <= 3; i++)           //pro "i" od 1 do 3 -- CHYBA
4  {                                      //idexy maji rozsah od 0 do 2
5      printf("y[%i] = %f\n", i, y[i]); //vypis index a hodnotu prvku
6  }
```

Na rozdíl od některých jiných jazyků, jazyk C nepodporuje záporné indexy:

```
7  int z[4] = {10, 20, 30, 40};          //pole ctyr celych cisel "z"
8
9  printf("z[%i] = %i\n", -1, z[-1]);    //CHYBA -- neplatny index!
```

Vyzkoušejte si:

- Vytvořte si pole pěti desetinných čísel a ze vstupu do něj načtěte hodnoty.
- Zadaná čísla vypište v obráceném pořadí.
- Vypište hodnotu maxima.
- Vypište index minima.

Například:

- (2.2, 5.5, 4.4, 1.1, 3.3) => 3.3 1.1 4.4 5.5 2.2
  => Hodnota maxima je 5.5
  => Index minima je 3