Funkce a argumenty programu

IZP-cv04

Ing. Jakub Husa

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

ihusa@fit.vut.cz

23. října 2023

Funkce

Funkce

Zdrojový kód programu se skládá z podprogramů (funkcí):

- Funkce nám umožňují část programu volat opakovaně a s různými parametry.
- Hlavička funkce obsahuje návratový typ, identifikátor a seznam parametrů.
- Funkce končí příkazem – `return` – který definuje její návratovou hodnotu.

V jazyku C u funkcí rozlišujeme mezi deklarací a definicí:

- Deklarace – říká jak funkci budeme volat, je tvořena hlavičkou a středníkem.
- Definice – říká co funkce bude dělat, je tvořena hlavičkou a tělíčkem.

```
1 //deklarace funkce
2 int soucet(int x, int y); //NavratovyTyp Identifikator (SeznamParametru);
3
4 //definice funkce
5 int soucet(int x, int y)  //NavratovyTyp Identifikator (SeznamParametru)
6 {                         //zacatek tela funkce
7     int vysledek = x + y; //vypocet
8     return vysledek;      //predani navratove hodnoty (konec funkce)
9 }                         //konec tela funkce
```

Deklarace funkcí

Funkci musíme deklarovat nebo definovat před tím než ji budeme volat:

- Pokud všechny funkce deklarujeme na začátku programu, nemusíme pak řešit jejich vzájemné volání a pořadí definic.
- Pozor – v projektech předmětu IZP je deklarace funkcí povinná!

```
1  int main()                    //hlavicka funkce main (zacatek programu)
2  {                             //"warning: implicit declaration of function"
3      int x = soucet(10,20);    //CHYBA -- volana funkce soucet jeste
4                                //  nebyla deklarovana ani definovana
5      printf("%i\n", x);        //vypis
6      return 0;                 //konec funkce main (konec programu)
7  }
8
9  int soucet(int x, int y)      //hlavicka funkce soucet
10 {
11     return x + y;             //konec funkce soucet
12 }
```

U složitějších programů sestavovaných z více souborů se funkce deklarují v samostatných hlavičkových souborech s příponou .h.

Návratový typ

Návratový typ je datový typ hodnoty kterou funkce vrací:

- Pokud funkce žádnou hodnotu nevrací, její návratový typ je – `void` –.
- U funkcí typu – `void` – můžeme vynechat příkaz – `return` –.

```
1  void prazdnaFunkce()  //funkce bez navratove hodnoty (a bez parametru)
2  {
3      //return;         //prikaz return bez navratove hodnoty lze vynechat
4  }
```

Návratovou hodnotu funkce můžeme použít i jako parametr:

```
5  int soucet(int x, int y)     //definice
6  {
7      return x + y;            //konec funkce soucet
8  }
9
10 int main()                   //hlavicka funkce main (zacatek programu)
11 {
12     printf("%i + %i = %i\n", 10, 20, soucet(10,20)); //navratova hodnota
13                              //funkce soucet je parametrem funkce printf
14     return 0;                //konec funkce main (konec programu)
15 }
```

Vyzkoušejte si:

- Deklaruje a definujte funkci prepona, která pomocí Pythagorovy věty spočítá délku přepony pravoúhlého trojúhelníku.

$$c = \sqrt{a^2 + b^2}$$

- Parametry i návratová hodnota funkce jsou desetinná čísla.
- Funkce musí ověřit platnost parametrů (obě odvěsny musejí být kladné), a pro neplané parametry vrátit hodnotu 0.0.
- V mainu načtěte dvě desetinná čísla, zavolejte funkci prepona, a vypište výsledek.

Například:

- (1.0, 1.0) => c = 1.414
- (3.0, 4.0) => c = 5.000
- (1.0, 0.0) => c = 0.000
- (3.0, -4.0) => c = 0.000

Pravdivostní hodnoty

Funkce mohou vracet i pravdivostní hodnotu datového typu – `bool` – (boolean):

- Používání tohoto datového typu vyžaduje knihovnu stdbool.h.

```
1  #include <stdbool.h>     //knihovna pro pravdivostni hodnoty
2  bool jeKladne(int x)     //definice funkce
3  {
4      if (x > 0)           //pokud je cislo vetsi jak nula
5          return true;     //vracime hodnotu TRUE
6      else                 //jinak
7          return false;    //vracime hodnotu FALSE
8  }
9
10 int main()               //hlavicka funkce main (zacatek programu)
11 {
12     int x;               //cele cislo "x"
13     scanf("%i", &x);     //ze vstupu do "x" nacteme hodnotu
14     if (jeKladne(x))     //pokud volana funkce vrati TRUE
15         printf("cislo %i je kladne", x);   //vypis
16     else                 //jinak
17         printf("cislo %i neni kladne", x); //vypis
18     return 0;            //konec funkce main (konec programu)
19 }
```

Vyzkoušejte si:

- Deklaruje a definujte funkci porovnej, která bez rozlišování velikosti písmen porovná zda se dva znaky shodují.
- Vstupem funkce jsou dva znaky.
- Výstupem funkce je pravdivostní hodnota.
- V mainu načtěte dva znaky, zavolejte funkci porovnej, a vypište výsledek.

Například:

- (A, A) => Znaky A a A jsou stejne
- (A, a) => Znaky A a a jsou stejne
- (A, !) => Znaky A a ! jsou rozdilne
- (!, !) => Znaky ! a ! jsou stejne

Pole a jeho délku funkcím předáváme jako dva samostatné parametry:

- V parametrech funkce pole označujeme prázdnými hranatými závorkami ([ ]).

```
1  int sumaPole(int delka, int pole[]) //definice funkce
2  {
3      int suma = 0;                   //pocatecni hodnota sumy je 0
4      for (int i = 0; i < delka; i++) //pro vsechny prvky pole
5          suma += pole[i];            //k sume pricteme prvek
6      return suma;                    //konec funkce, vracime sumu
7  }
8
9  int main()                          //zacatek programu
10 {
11     int x[3] = {10, 20, 30};        //pole tri celych cisel "x"
12     int suma = sumaPole(3, x);      //pole predavame BEZ hranatych zavorek
13     printf("%i\n", suma);           //vypis
14     return 0;                       //konec programu
15 }
```

Pole nemůžeme používat jako návratový typ:

```
16 int[] funkce(int delka, int pole[]);//CHYBA -- nelze vracet pole
```

Vyzkoušejte si:

- Napište funkci maCislo která rozhodne jestli pole obsahuje dané číslo.
- Napište funkci jeMnozina která rozhodne jestli je pole množina, tedy že se v něm žádná z čísla neopakují.
- Funkce jsou je deklarovány tímto způsobem:

```
1 bool maCislo(int delka, int pole[], int cislo);
2 bool jeMnozina(int delka, int pole[]);
```

- V mainu si vytvořte pole celých čísel a inicializujte ho nějakými hodnotami, vytvořte a načtěte si jedno celé číslo, a vypište výsledek obou funkcí.

Například:

- (1 2 3, 4) => Pole nema cislo
- (1 2 3) => Pole je mnozina
- (1 2 3 4 2, 4) => Pole ma cislo
- (1 2 3 4 2) => Pole neni mnozina

Argumenty programu

Argumenty programu

Program můžeme spustit s nějakými argumenty:

- V Code::Blocks je nastavíme v Project -> Set program's arguments.
- Na příkazové řádce je při spuštění píšeme za jméno programu.
- `./main ARG1 ARG2 ARG3`

Argumenty se programu předají jako parametry funkce main:

- Aby funkce main měla k parametrům přístup musíme upravit její hlavičku.

```
1 int main(int argc, char* argv[]) //upravena hlavicka funkce main
```

- argc – celé číslo udávající počet argumentů programu.
- argv – pole textových řetězců obsahující jednotlivé argumenty.

Program má vždy alespoň jeden argument – umístění spuštěného souboru:

```
2 int main(int argc, char* argv[]) //upravena hlavicka funkce main
3 {
4     printf("%i\n", argc);         //vypis pocet argumentu programu
5     return 0;                    //konec programu
6 }
```

Zpracování argumentů

Parametr – `char* argv[]` – je pole textových řetězců:

- V podstatě je to 2D pole znaků kde každý řádek může mít jinou délku.

```
1  #include <stdio.h>                  //vkladame knihovnu pro vstup a vystup
2  #include <string.h>                 //vkladame kihovnu retezcovych funkci
3
4  int main(int argc, char* argv[])    //upravena hlavicka funkce main
5  {
6      printf("Program ma %i argumentu\n", argc); //vypis
7      for (int i = 0; i < argc; i++) //pro vsechny argumenty programu
8      {
9          printf("Argument %s ma %i znaku\n", argv[i], strlen(argv[i]));
10     }                               //vypis argument a jeho delku
11     return 0;                       //konec programu
12 }
```

Argumenty programu **NEJSOU** to samé jako jeho vstup:

- Argumenty program dostává při spuštění, jednou, vždy ve funkci main.
- Vstupy si program načítá za běhu, kdykoliv, funkcemi z knihovny stdio.h.

Vyzkoušejte si:

- Napište program který bude spouštěn s alespoň jedním argumentem (kromě samotného názvu programu).
- Ze vstupu načtěte řetězec o maximální délce 100 znaků, a spočítejte kolikrát se v něm vykytuje první znak daného argumentu.
- Pokud byl program spuštěn bez argumentů, vypište hlášení nedostatek argumentu a program ukončete.

Například:

- `./main A` => (ABCD) => Retezec obsahuje 1 znaku A
  => (AAAA) => Retezec obsahuje 4 znaku A
- `./main BC` => (ABCD) => Retezec obsahuje 1 znaku B
  => (AAAA) => Retezec obsahuje 0 znaku B
- `./main` => (ABCD) => Nedostatek argumentu
  => (AAAA) => Nedostatek argumentu

Převod řetězce na číslo

Protože argumenty programu jsou řetězce nemůžeme s nimi počítat:

```
1  int x = argv[1] + argv[2];        //CHYBA -- argv[i] jsou retezce ne cisla
```

Řetězec na číslo převedeme funkcemi z knihovny stdlib.h:

- Celé číslo získáme funkcí atoi, desetinné číslo získáme funkcí atof.

```
2  #include <stdio.h>                //knihovna pro vstup a vystup
3  #include <stdlib.h>               //knihovna zakladnich funkci
4
5  int main(int argc, char* argv[]) //upravena hlavicka funkce main
6  {
7      //POZOR -- argument "0" je vzdy nazev programu
8      int x = atoi(argv[1]);        //"x" je argument "1" prevedeny na cislo
9      int y = atoi(argv[2]);        //"y" je argument "2" prevedeny na cislo
10
11     printf("%i + %i = %i\n", x, y, x+y);
12     return 0;                     //konec programu
13 }
```

Návratová hodnota programu

Návratová hodnota funkce main říká jestli program uspěl nebo selhal:

- Při úspěchu main musí vrátit hodnotu 0 (EXIT_SUCCESS).
- Při selhání main musí vrátit hodnotu jinou než 0 (EXIT_FAILURE).
- Různé typy selhání obvykle vracejí různou hodnotu – záleží na zadání projektu.

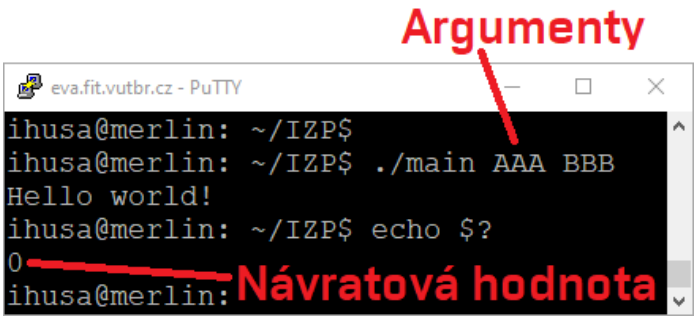
```
1  #include <stdio.h>                  //knihovna pro vstup a vystup
2  #include <stdlib.h>                 //knihovna zakladnich funkci
3
4  int main(int argc, char* argv[]) //upravena hlavicka funkce main
5  {
6      if (argc > 2)                   //overujeme jestli ma program
7      {                               //dostatecny pocet argumentu
8          int x = atoi(argv[1]) + atoi(argv[2]); //prevod a vypocet
9          printf("%s + %s = %i\n", argv[1], argv[2], x);
10         return 0;                   //konec programu -- program USPEL
11     }
12     else                            //jinak (pokud program nedostal)
13     {                               //      (dostatek argumentu)
14         return 1;                   //konec programu -- program SELHAL
15     }
16 }
```

Argumenty a návratová hodnota

V Code::Blocks argumenty uvidíme v hlavičce okna:

Na příkazové řádce návratovou hodnotu zjistíme příkazem:

- `echo $?`



Vyzkoušejte si:

- Napište program který jako argument dostane jedno desetinné číslo.
- Spočítejte jeho odmocninu, vypište ji, a program ukončete s hodnotou 0.
- Pokud se odmocnina nedá spočítat, program ukončete s hodnotou 1.
- Pokud program nedostal argument, program ukončete s hodnotou 2.
- Funkčnost programu ověřte na serveru merlin.fit.vutbr.cz.

Například:

- `./main 4` => Odmocnina z 4.000 je 2.000 => 0
- `./main 2.25` => Odmocnina z 2.250 je 1.500 => 0
- `./main -1` => => 1
- `./main` => => 2