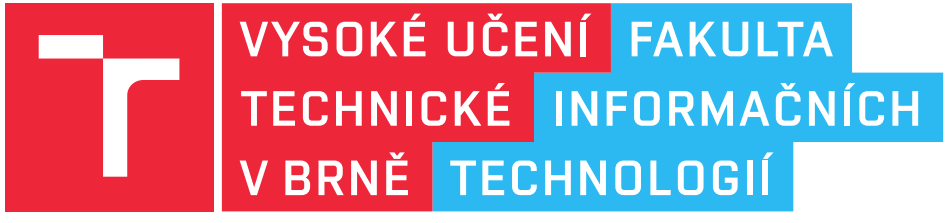
Základy programovania
Cvičenie 3

Dátové typy, argumenty programu

Cvičiaci: Ing. Magdaléna Ondrušková, (iondruskova)



1. října 2024

Integer

```c
#include <stdio.h>
int main() {
    int num;
    scanf("%d", &num);  // nacitanie celeho cisla
    printf("Zadali ste cislo: %d\n", num);  // vypis cel ho cisla
    return 0;
}
```

Float

- `float` - presnosť cca 8 číslic

```c
include <stdio.h>
int main() {
    float num;
    scanf("%f", &num);  // nacitanie desatinneho cisla (float)
    printf("Zadali ste cislo: %f\n", num);  // vypis desatinneho cisla
    return 0;
}
```

Double

- `double` - presnosť cca 16 číslic

```c
#include <stdio.h>
int main() {
    double num;
    scanf("%lf", &num);  // nacitanie desatinneho cisla (double)
    printf("Zadali ste cislo: %lf\n", num);  // vypis desatinneho cisla s vyssou presnostou
    return 0;
}
```

Char

```c
#include <stdio.h>
int main() {
    char ch;
    scanf("%c", &ch);  // nacitanie jedneho znaku
    printf("Zadali ste znak: %c\n", ch);  // vypis jedneho znaku
    return 0;
}
```

String

- Postupnosť znakov ukončená špeciálnym znakom `\0`

```c
#include <stdio.h>
int main() {
    char str[6] = "Hello";  // pole obsahujuce 5 znakov + '\0'
    char str[] = "Hello, world!"; // Ako velke bude toto pole?
    return 0;
}
```

Načítanie a vypísanie reťazcov:

```c
#include <stdio.h>
int main() {
    char str[101];  // pole pre 100 znakov + 1 pre '\0'
    printf("Zadajte retazec (max 100 znakov): ");
    scanf("%100s", str);  // nacitanie retazca, obmedzene na 100
     znakov
    printf("Zadali ste retazec: %s\n", str);  // vypis retazca

    return 0;
}
```

Tabuľka základných dátových typov

Dátový typ	PRINTF formát	SCANF formát
int	%i alebo %d	%i alebo %d
float	%f	%f
double	%lf	%lf
char	%c	%c
char()	%s	%s

Tabulka: Prehľad základných typov v jazyku C

- Obsahuje funkcie pre prácu s reťazcami

Zistenie dĺžky reťazca:

```c
#include <stdio.h>
#include <string.h>  // NEZABUDNUT nacitat kniznicu

int main() {
    char str[] = "Hello, world!"; // Zistenie dlzky retazca
    int len = strlen(str);
    int len2 = sizeof(str);

    printf("Dlzka retazca: %i\n", len);  // Vypise: 13
    printf("Dlzka retazca: %i\n", len2);  // Vypise: 14

    return 0;
}
```

Kopírovanie reťazcov

- Dva parameter: `kam` a `odkial`

```c
#include <stdio.h>
#include <string.h>

int main() {
    char src[6] = "Hello";
    char dest[50];  // Musi byt dostatocne velke

    // Kopirovanie src do dest
    strcpy(dest, src);

    printf("Dest: %s\n", dest);  // Vypise: Hello

    return 0;
}
```

Porovnanie reťazcov

- Nie je možné porovnávať pomocou ==, porovnávalo by to adresy reťazcov
- Návratové hodnoty:
  - Vráti 0 ak sa reťazce zhodujú
  - Vráti kladné číslo ak je prvý reťazec (prvý nezhodujúci sa znak v ASCII) väčší
  - Vráti záporné číslo ak je druhý reťazec (prvý nezhodujúci sa znak v ASCII) väčší

```
1  // Nacitanie kniznic
2  int main() {
3      char str1[50] = "Ahoj";
4      char str2[50] = "Ahoj";
5      int result = strcmp(str1, str2); // Porovnanie str1 a str2
6      if (result == 0) {
7          printf("Retazce su rovnake\n");
8      } else if (result > 0) {
9          printf("Prvy retazec je vacsi\n");
10     } else {
11         printf("Druhy retazec je vacsi\n");
12     }
13     return 0;
14 }
```

Úloha 1: Načítajte dva reťazce (každý max 100 znakov) a vypíšte reťazec, ktorý reťazec je dlhší.

- Rozšírenie: Vypíšte, ktorý reťazec má viac malých písmen.

Úloha 2: Načítajte reťazec (max 20 znakov), a následne vypíšte tento reťazec odzadu

- Vstup: *Dom*
- Výstup: *moD*

Úloha 2: Načítajte reťazec (max 20 znakov), a následne vypíšte tento reťazec odzadu

- Vstup: *Dom*
- Výstup: *moD*

Úloha 3: Upravte kód, aby overil, či daný reťazec je palindróm

- kajak
- krk
- oko

Argumenty programu

Program môžeme spúšťať s nejakými argumentami, ktoré píšeme za meno programu, oddelené medzerami

- `./hello argument1 argument2 argument3 ...`

Argumenty predávame do programu ako parametre funkcie `main`

- `argc` - celé číslo, udáva počet argumentov
- `argv[]` - pole textových reťazcov obsahujúce jednotlivé argumenty

```
1 int main(int argc, char *argv[]) {
2     // Telo programu
3 }
```

Program má vždy aspoň 1 argument:

- `argv[0]` - názov programu
- `argv[1], argv[2], ..., argv[argc-1]` - ďalšie argumenty zadané v príkazovom riadku

Výpis argumentov programu:

```c
#include <stdio.h>
int main(int argc, char *argv[]) {
    // argc obsahuje pocet argumentov
    printf("Pocet argumentov: %d\n", argc);
    // argv je pole retazcov (argumentov)
    for (int i = 0; i < argc; i++) {
        printf("Argument %d: %s\n", i, argv[i]);
    }
    return 0;
}
```

Ak program spustíme s argumentami:

```
./program Hello World
```

Výpis programu by bol:

```
1 Pocet argumentov: 3
2 Argument 0: ./program
3 Argument 1: Hello
4 Argument 2: World
```

Pozor!

- Argumenty != Vstup
- Vstupy sú načítavané za behu programu
- Argumenty sa načítajú raz, pri spustení (vo funkcii main)

Argumenty po načítaní sú vždy `reťazce`.

- `atoi` - funkcia, ktorá prevedie reťazec na celé číslo
- `atof` - funkcia, ktorá prevedie reťazec na desatinné číslo

Použitie:

```
1 #include <stdlib.h>  // Pre funkciu atoi
2
3 ...
4
5 int num1 = atoi(argv[1]);
```

Úloha A: Spustite program s dvoma argumentami. Program vypíše súčet tých čísiel.

Úloha B: Spustite program s dvoma argumentami. Program vypíše, ktoré číslo je väčšie (Niečo ako *Väčší argument ma hodnotu: ...*).

Úlohy na precvičenie:

Úloha: Vytvorte program, ktorý bude spúšťaný s práve jedným argumentom. V programe načítajte užívateľský vstup o dĺžke max 20 znakov. Výstup programu vypíšte.

- **Varianta A:**
  - Ak je zadaný argument `count` vypíšte počet samohlások v reťazci
  - Ak je zadaný argument `tolower` transformujte reťazec do upper case
    - Môžete použiť funkciu *toupper()*
  - Ak je zadaný argument `replace`, nahradíte znaky `a/A` v reťazci znakom `-`
- **Varianta B:**
  - Ak je zadaný argument `count` vypíšte počet znakov ktoré nie sú písmeno
  - Ak je zadaný argument `toupper` transformujte reťazec do lower case
    - Môžete použiť funkciu *tolower()*
  - Ak je zadaný argument `find`, nájdete v reťazci všetky znaky `a/A` a postupne vypíšte na ktorých indexoch (pozíciach) sa dané znaky nachádzali.