# Znaky, řetězce a argumenty programu

## IZP-cv03

**Ing. Jakub Husa**

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


30. září 2024

# Znaky

# Znaky

Znaky ukládáme do proměnných datového typu – `char` – (character):

- Ve zdrojovém kódu znaky ohraničujeme obyčejnými uvozovkami ('A', 'B', ...).
- Znaky načítáme a vypisujeme knihovními funkcemi se značkou %c:

```
1 char x;                //promenna typu znak, jmenem "x"
2 scanf("%c", &x);       //ze vstupu do ni nacitame znak
3 printf("%c\n",x);      //nacteny znak vypisujeme na vystup
```

V paměti počítače se znaky ukládají jako celá čísla v rozsahu od 0 do 255:

- Každému z čísel je přiřazen nějaký znak podle kódové tabulky.

```
4 printf("%c\n", 65);   //vypis znak cislo 65
5 printf("%i\n", 'A'); //vypis ciselnou hodnotu znaku 'A'
```

Znaky od 0 do 127 jsou vždy kódovány tabulkou ASCII:

- Znaky od 32 do 126 představují standardní abecedu.
- Znaky od 0 do 31 a 127 jsou netisknutelné řídící znaky (null, escape, delete, ... ).

# Tabulka ASCII

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | 16 | DLE | 32 | | 48 | 0 | 64 | @ | 80 | P | 96 | ` | 112 | p |
| 1 | SOH | 17 | DC1 | 33 | ! | 49 | 1 | 65 | A | 81 | Q | 97 | a | 113 | q |
| 2 | STX | 18 | DC2 | 34 | " | 50 | 2 | 66 | B | 82 | R | 98 | b | 114 | r |
| 3 | ETX | 19 | DC3 | 35 | # | 51 | 3 | 67 | C | 83 | S | 99 | c | 115 | s |
| 4 | EOT | 20 | DC4 | 36 | $ | 52 | 4 | 68 | D | 84 | T | 100 | d | 116 | t |
| 5 | ENQ | 21 | NAK | 37 | % | 53 | 5 | 69 | E | 85 | U | 101 | e | 117 | u |
| 6 | ACK | 22 | SYN | 38 | & | 54 | 6 | 70 | F | 86 | V | 102 | f | 118 | v |
| 7 | BEL | 23 | ETB | 39 | ' | 55 | 7 | 71 | G | 87 | W | 103 | g | 119 | w |
| 8 | BS | 24 | CAN | 40 | ( | 56 | 8 | 72 | H | 88 | X | 104 | h | 120 | x |
| 9 | HT | 25 | EM | 41 | ) | 57 | 9 | 73 | I | 89 | Y | 105 | i | 121 | y |
| 10 | LF | 26 | SUB | 42 | * | 58 | : | 74 | J | 90 | Z | 106 | j | 122 | z |
| 11 | VT | 27 | ESC | 43 | + | 59 | ; | 75 | K | 91 | [ | 107 | k | 123 | { |
| 12 | FF | 28 | FS | 44 | , | 60 | < | 76 | L | 92 | \ | 108 | l | 124 | \| |
| 13 | CR | 29 | GS | 45 | - | 61 | = | 77 | M | 93 | ] | 109 | m | 125 | } |
| 14 | SO | 30 | RS | 46 | . | 62 | > | 78 | N | 94 | ^ | 110 | n | 126 | ~ |
| 15 | SI | 31 | US | 47 | / | 63 | ? | 79 | O | 95 | _ | 111 | o | 127 | DEL |

Vyzkoušejte si:

```
1 for (int i = 32; i < 127; i++)   //pro vsechny tisknutelne znaky
2     printf("%i = %c\n", i, i);   //vypis cislo a odpovidajici znak
```

Znaky od 128 do 255 jsou kódovány jinými tabulkami:

- Jejich interpretace závisí na nastavení vypisujícího programu.
- Ve zdrojových kódech tyto znaky **NEPOUŽÍVEJTE**, jinak váš kód nepůjde přečíst!

```
1  printf("ěščřžýáíé\n");     //to jestli se vam text zobrazi spravne
2                             //zavisi na nastaveni vasi konzole a editoru
```

Znaky můžeme porovnávat a provádět s nimi aritmetické operace:

```
3  char x;                    //promenna typu znak, jmenem "x"
4  scanf("%c", &x);           //ze vstupu do ni nacitame znak
5
6  if(x >= 'A' && x <= 'Z') //pokud je znak velke pismeno
7  {                          //prictenim konstanty ho prevedeme na male
8      x = x + 32;            //(v tabulce ASCII se mala a velka
9  }                          //  pismena lisi presne o 32 pozic)
10
11 printf("%c\n", x);         //vypiseme upraveny znak
```

Vyzkoušejte si:

- Ze vstupu si načtěte dva znaky a ověřte jestli platí následující podmínky.
- První znak je číslice.
- Druhý znak není písmeno.
- Alespoň jeden znak není ani číslice ani písmeno.
- Oba znaky jsou písmena a bez ohledu na velikost jsou stejné písmeno.
- Práci si můžete usnadnit funkcemi z knihovny ctype.h.

Například:

- (1, A) => Prvni znak je cislice
- (A, 1) => Druhy znak neni pismeno
- (+, a) => Alespon jeden znak neni ani cislice ani pismeno
- (A, a) => Oba znaky jsou stejne pismeno, bez ohledu na velikost

# Řetězce

# Řetězce

Textový řetězec (string) je posloupnost znaků zakončená řídícím znakem NUL ('\0'):

```
1  char x[6] = "Hello";   //pole obsahujici retezec sesti znaku
2                         //{'H', 'e', 'l', 'l', 'o', '\0' }
```

Řetězce načítáme a vypisujeme knihovními funkcemi se značkou %s:

- Aby při načítání nedošlo k neplatnému přístupu do paměti počet načítaných znaků vždy omezujeme na hodnotu o jedna menší než je velikost pole!
- Při načítání řetězců nepoužíváme dereferenční operátor (&)!

```
3  char y[10];             //vytvarime pole o velikosti DESET znaku
4  scanf("%9s", y);        //do pole nacitame maximalne DEVET znaku
5  printf("%s\n", y);      //vypisujeme nacteny retezec
```

Značka %s načítá řetězec ukončený bílým znakem (mezera, tabulátor, konec řádku):

- Celý řádek, včetně bílých znaků, načteme složitější formátovací značkou.

```
6  scanf("%9[^\n]", y);    //do pole nacitame 9 znaku zakoncenych koncem radku
7  printf("%s\n", y);      //vypisujeme nacteny retezec
```

# Knihovna řetězcových funkcí

Funkce pro práci s řetězci poskytuje knihovna string.h:

- Délku řetězce (bez ukončujícího znaku NUL) zjistíme funkcí strlen.
- Řetězec bude vždy kratší něž pole ve kterém je uložen.

```
1  #include <stdio.h>           //vkladame knihovnu pro vstup a vystup
2  #include <string.h>          //vkladame kihovnu retezcovych funkci
3
4  int main()                   //hlavicka funkce main (zacatek programu)
5  {
6      char x[80] = "Hello";    //pole znaku "x" obsahujici retezec "Hello"
7      int delkaX = strlen(x); //delka retezce v poli "x"
8      printf("Delka retezce %s je %i\n", x, delkaX); //vypis
9
10     char y[80];              //neinicializovane pole znaku "y"
11     scanf("%79s", y);        //do pole "y" nacitame retezec
12     int delkaY = strlen(y); //delka retezce v poli "y"
13     printf("Delka retezce %s je %i\n", y, delkaY); //vypis
14
15     return 0;                //konec funkce main (konec programu)
16 }
```

# Porovnávání a kopírování řetězců

Řetězce porovnáme funkcí strcmp která vrací hodnotu 0 pokud se texty shodují:

- Pozor – řetězce nejsou čísla a proto je nelze porovnávat relačními operátory, operátor == by porovnával jestli jsou řetězce na stejné adrese!

```
1  char a[80] = "Hello";  //pole "a" obsahujici retezec "Hello"
2  char b[80] = "Hello";  //pole "b" obsahujici retezec "Hello"
3  if (strcmp(a, b) == 0) //pokud se texty retezcu shoduji
4      printf("Retezce %s a %s se shoduji\n", a, b);
5  else                   //jinak
6      printf("Retezce %s a %s se odlisuji\n", a, b);
7  //if (a == b)          //CHYBA -- operator "==" by porovnal adresy
```

Řetězce kopírujeme funkcí strcpy s parametry kam a odkud:

- Po inicializaci už řetězce nelze přiřazovat operátorem = .

```
8  char c[80] = "Ahoj"; //pole "c" obsahujici retezec "Ahoj"
9  char d[80];          //neinicializovane pole "d"
10 strcpy(d, c);        //do pole "d" zkopiruj retezc z pole "c"
11 printf("%s\n", d);   //vypis obsah pole "d"
12 //d = c;             //CHYBA -- "assignment to expression with array type"
```

Vyzkoušejte si:

- Vytvořte si dostatečně velké pole a načtěte do něj řetězec až sto znaků.
- Spočítejte kolik řetězec obsahuje malých a velkých písmen.
- Všechny ostatní znaky v řetězci nahraďte pomlčkou ('–').
- Vypište upravený řetězec a kolik obsahuje jakých písmen.

Například:

- (Ahoj) => Retezec Ahoj obsahuje 3 malych a 1 velkych pismen
- (1A2B3C) => Retezec –A–B–C obsahuje 0 malych a 3 velkych pismen
- (12345678) => Retezec –––––––– obsahuje 0 malych a 0 velkych pismen

# Argumenty programu

# Argumenty programu

Program může být spuštěn s nějakými argumenty:

- Argumenty píšeme při spuštění za jméno programu, oddělené mezerami.
- `./main argument_1 argument_2 argument_3`

Argumenty se do programu předají jako parametry funkce main:

- Aby k nim měl main přístup musíme upravit jeho 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 – jméno spuštěného programu:

```
2 int main(int argc, char* argv[]) //upravena hlavicka funkce main
3 {
4     printf("%i\n", argc);          //vypis pocet argumentu programu
5     printf("%s\n", argv[0]);       //vypis jmeno programu
6     return 0;                      //konec programu
7 }
```

# Zpracování argumentů

Datový typ parametru – `char* argv[]` – je pole textových řetězců:

- V podstatě je to dvourozměrné 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.

# 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 můžeme převést 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 }
```

Vyzkoušejte si:

- Vytvořte program který jako argumenty dostane několik desetinných čísel.
- Ze zadaných argumentů počítejte maximum a aritmetický průměr.
- Platnost argumentů neřešte (dá se ověřit funkcí strtof).

Například:

- `./main` => Nedostatek argumentu
- `./main 1.0` => Maximum je 1.0, prumer je 1.0
- `./main 1.0 2.0` => Maximum je 2.0, prumer je 1.5
- `./main -1.0 -2.0 -3.0` => Maximum je -1.0, prumer je -2.0

# Na procvičení

Vyzkoušejte si:

- Ze vstupu si načtěte jeden řetězec o maximální délce deset znaků.
- Podle toho s jakým argumentem byl program spuštěn a proveďte jednu z operací.
- tolower – všechna velká písmena v řetězci převeďte na malá.
- notalnum – z řetězce vypište všechny znaky které nejsou ani číslice ani písmena.
- palindrom – vypište jestli načtený řetězec je nebo není palindrom.

Například:

- `./main tolower` => ahoj_svete
  `Ahoj_Svete`
- `./main notalnum` => +=
  `A+1=B`
- `./main palindrom` => alice neni palindrom
  `alice`
- `./main palindrom` => bob je palindrom
  `bob`