Znaky, řetězce, zanořené cykly a vícerozměrná pole

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

23. října 2023

Znaky

Znaky

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

- Ve zdrojovém kódu znaky ohraničujeme jednoduchými 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 jde porovnávat relačními operátory 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 načtěte jeden znak, a vypište jestli jde o číslici, písmeno, netisknutelný řídící znak nebo nějaký jiný tisknutelný znak.

Například:

- (1) => 1 je cislice
- (A) => A je pismeno
- (z) => z je pismeno
- (ENTER) =>
  je netisknutelny ridici znak
- (=) => = je jiny tisknutelny znak

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 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 dvě pole znaků (X a Y).
- Do pole X načtěte řetězec a zkopírujte ho do pole Y.
- Poslední znak řetězce v poli Y (před znakem NUL) nahraďte podtržítkem ('_').
- Řetězce X a Y spolu porovnejte a vypište jestli se shodují.

Například:

- (abcd) => Retezce abcd a abc_ se odlisuji
- (abc_) => Retezce abc_ a abc_ se shoduji
- (ab) => Retezce ab a a_ se odlisuji
- (a_) => Retezce a_ a a_ se shoduji

Vyzkoušejte si:

- Vytvořte si dostatečně velké pole a načtěte do něj řetězec až osmi znaků.
- Spočítejte kolik řetězec obsahuje malých písmen.
- Všechny ostatní znaky v řetězci nahraďte pomlčkami (’–’).
- Vypište počet malých písmen a upravený řetězec.

Například:

- (abcdefgh) => Retezec abcdefgh obsahuje 8 malych pismen
- (AbCdEfGh) => Retezec –b–d–f–h obsahuje 4 malych pismen
- (01234567abcd) => Retezec –––––––– obsahuje 0 malych pismen
- (a) => Retezec a obsahuje 1 malych pismen

Zanořené cykly

Zanořené cykly

Cykly můžeme vzájemně zanořovat (stejně jako podmínky):

- V každém cyklu obvykle iterujeme přes jinou proměnnou.

```
1  for (int i = 0; i < 4; i++)            //pro "i" od 0 do 3
2  {
3      for (int j = 0; j < 4; j++)        //pro "j" od 0 do 3
4      {
5          printf("i=%i, j=%i\n", i, j); //vypis hodnotu "i" a "j"
6      }
7  }
```

Pokud tělo cyklu obsahuje jen jeden příkaz tak jeho závorky můžeme vynechat:

- Zanořený cyklus nebo podmínka jsou považovány za jeden strukturovaný příkaz.

```
8  for (int i = 0; i < 4; i++)            //pro promennou "i" od 0 do 3
9      for (int j = 0; j < 4; j++)        //pro promennou "j" od 0 do 3
10         printf("i=%i, j=%i\n", i, j); //vypis hodnotu "i" a "j"
```

Cyklus můžeme předčasně ukončit příkazem – `break` – (zlom):

```
1  for (int i = 0; i < 5; i++) //pro "i" od 0 do 4
2  {
3      if (i == 2)             //pokud je "i" rovno 2
4          break;              //ukonci cyklus
5      printf("%i ", i);       //vypis hodnotu "i"
6  }
7  printf("\n");               //vypis konec radku
```

Zbytek těla cyklu můžeme přeskočit příkazem – `continue` – (pokračuj):

```
8  for (int i = 0; i < 5; i++) //pro "i" od 0 do 4
9  {
10     if (i == 2)             //pokud je "i" rovno 2
11         continue;           //preskoc zbytek tela cyklu
12     printf("%i ", i);       //vypis hodnotu "i"
13 }
14 printf("\n");               //vypis konec radku
```

Přerušení zanořeného cyklu

Pokud – `break` – nebo – `continue` – použijeme uvnitř zanořeného cyklu tak se vnitřní cyklus přeruší ale vnější cyklus nebude ovlivněn:

```
1  #include <stdio.h>                    //knihovna pro vstup a vystup
2
3  int main()                            //hlavicka funkce main
4  {
5      for(int i = 0; i < 10; i++)       //pro "i" od 0 do 9
6      {
7          for(int j = 0; j < 10; j++)   //pro "j" od 0 do 9
8          {
9              printf("X");              //vypis znak "X"
10             if (i == j)               //pokud se "i" rovna "j"
11             {
12                 break;                //ukonci cyklus
13             }
14         }
15         printf("\n");                 //vypis konec radku
16     }
17     return 0;                         //konec funkce main
18 }
```

Vyzkoušejte si:

- Vytvořte si pole znaků a načtěte do něj řetězec.
- Vypište všechny tisknutelné znaky které mají v řetězci alespoň jeden výskyt.
- Na pořadí znaků nezáleží, každý ale musí být vypsán pouze jednou.

Například:

- (ABCD) => A B C D
- (DABA) => A B D
- (aAaA) => A a
- (1+1=2) => + 1 2 =

Vícerozměrná pole

Pole mohou mít v jazyku C několik rozměrů (dimenzí):

- Vícerozměrné pole vytvoříme použitím několika hranatých závorek ([] []).

```
1  int a;           //cele cislo
2  int b[4];        //1D pole  4 celych cisel (                  4 sloupce)
3  int c[3][4];     //2D pole 12 celych cisel (         3 radky, 4 sloupce)
4  int d[2][3][4]; //3D pole 24 celych cisel (2 patra, 3 radky, 4 sloupce)
```

K prvkům pole přistupujeme několika indexy a hranatými závorkami ([] []):

- Pole můžeme inicializovat zanořenými složenými závorkami ({{}}):

```
5  int x[2][3] = {{0,1,2}, {3,4,5}}; //dvourozmerne pole cisel
6
7  for(int i = 0; i < 2; i++)         //pro kazdy radek
8  {
9      for(int j = 0; j < 3; j++)     //pro kazdy sloupec
10     {
11         printf("%i ", x[i][j]);    //vypis jeden prvek pole
12     }
13     printf("\n");                  //vypis znak konce radku
14 }
```

Vyzkoušejte si:

- Vytvořte si pole čísel s 5 řádky a 5 sloupci, a inicializujte ho čísly od 0 do 24.
- Vypište první sloupec.
- Vypište poslední řádek.
- Vypište hlavní diagonálu.
- Vypište všechny prvky pod vedlejší diagonálou.

Například:

- Prvni sloupec : 0 5 10 15 20
  Posledni radek: 20 21 22 23 24
  Na hlavni : 0 6 12 18 24
  Pod vedlejsi : 9 13 14 17 18 19 21 22 23 24