

Základy programovania (IZP)

Reťazce (3. cvičenie)

Ing. Pavol Dubovec

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



Aký dátový typ používame pre uloženie znakov?

- Pre uloženie znakov používame dátový typ **char**.
- Každý znak je reprezentovaný **číselným kódom**.
- Najbežnejší číselný kód je **ASCII**.
- V jazyku C má typ char veľkosť 1-byte – rozsah hodnôt je **< 0; 255 >** alebo **< -128; 127 >**
- Netlačiteľné (netisknuteľné) znaky **< 0; 31 >** a **127**
- Príklad deklarácie a inicializácie a zobrazenie dátového typu char:

Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char
0	0000 0000	00	[NUL]	32	0010 0000	20	space	64	0100 0000	40	@	96	0110 0000	60	`
1	0000 0001	01	[SOH]	33	0010 0001	21	!	65	0100 0001	41	A	97	0110 0001	61	a
2	0000 0010	02	[STX]	34	0010 0010	22	"	66	0100 0010	42	B	98	0110 0010	62	b
3	0000 0011	03	[ETX]	35	0010 0011	23	#	67	0100 0011	43	C	99	0110 0011	63	c
4	0000 0100	04	[EOT]	36	0010 0100	24	\$	68	0100 0100	44	D	100	0110 0100	64	d
5	0000 0101	05	[ENQ]	37	0010 0101	25	%	69	0100 0101	45	E	101	0110 0101	65	e
6	0000 0110	06	[ACK]	38	0010 0110	26	&	70	0100 0110	46	F	102	0110 0110	66	f
7	0000 0111	07	[BEL]	39	0010 0111	27	'	71	0100 0111	47	G	103	0110 0111	67	g
8	0000 1000	08	[BS]	40	0010 1000	28	(72	0100 1000	48	H	104	0110 1000	68	h
9	0000 1001	09	[TAB]	41	0010 1001	29)	73	0100 1001	49	I	105	0110 1001	69	i
10	0000 1010	0A	[LF]	42	0010 1010	2A	*	74	0100 1010	4A	J	106	0110 1010	6A	j
11	0000 1011	0B	[VT]	43	0010 1011	2B	+	75	0100 1011	4B	K	107	0110 1011	6B	k
12	0000 1100	0C	[FF]	44	0010 1100	2C	,	76	0100 1100	4C	L	108	0110 1100	6C	l
13	0000 1101	0D	[CR]	45	0010 1101	2D	-	77	0100 1101	4D	M	109	0110 1101	6D	m
14	0000 1110	0E	[SO]	46	0010 1110	2E	.	78	0100 1110	4E	N	110	0110 1110	6E	n
15	0000 1111	0F	[SI]	47	0010 1111	2F	/	79	0100 1111	4F	O	111	0110 1111	6F	o
16	0001 0000	10	[DLE]	48	0011 0000	30	0	80	0101 0000	50	P	112	0111 0000	70	p
17	0001 0001	11	[DC1]	49	0011 0001	31	1	81	0101 0001	51	Q	113	0111 0001	71	q
18	0001 0010	12	[DC2]	50	0011 0010	32	2	82	0101 0010	52	R	114	0111 0010	72	r
19	0001 0011	13	[DC3]	51	0011 0011	33	3	83	0101 0011	53	S	115	0111 0011	73	s
20	0001 0100	14	[DC4]	52	0011 0100	34	4	84	0101 0100	54	T	116	0111 0100	74	t
21	0001 0101	15	[NAK]	53	0011 0101	35	5	85	0101 0101	55	U	117	0111 0101	75	u
22	0001 0110	16	[SYN]	54	0011 0110	36	6	86	0101 0110	56	V	118	0111 0110	76	v
23	0001 0111	17	[ETB]	55	0011 0111	37	7	87	0101 0111	57	W	119	0111 0111	77	w
24	0001 1000	18	[CAN]	56	0011 1000	38	8	88	0101 1000	58	X	120	0111 1000	78	x
25	0001 1001	19	[EM]	57	0011 1001	39	9	89	0101 1001	59	Y	121	0111 1001	79	y
26	0001 1010	1A	[SUB]	58	0011 1010	3A	:	90	0101 1010	5A	Z	122	0111 1010	7A	z
27	0001 1011	1B	[ESC]	59	0011 1011	3B	;	91	0101 1011	5B	[123	0111 1011	7B	{
28	0001 1100	1C	[FS]	60	0011 1100	3C	<	92	0101 1100	5C	\	124	0111 1100	7C	
29	0001 1101	1D	[GS]	61	0011 1101	3D	=	93	0101 1101	5D]	125	0111 1101	7D	}
30	0001 1110	1E	[RS]	62	0011 1110	3E	>	94	0101 1110	5E	^	126	0111 1110	7E	~
31	0001 1111	1F	[US]	63	0011 1111	3F	?	95	0101 1111	5F	_	127	0111 1111	7F	[DEL]

```
#include <stdio.h>
int main() {
    char znak = 'A'; // Deklarácia a inicializácia znaku
    printf("Znak: %c\n", znak); // Výpis znaku
    printf("Znak: %d\n, teda celé číslo\n", ch);
    return 0;
}
```

- Operácie s jednotlivými znakmi pomocou ich ASCII hodnôt.

```
#include <stdio.h>

int main() {
    char c;
    printf("Zadajte znak: ");
    scanf("%c", &c);
    if (c >= 'a' && c <= 'z') {
        c = c - 32; // Konverzia na veľké písmeno
    } else if (c >= 'A' && c <= 'Z') {
        c = c + 32; // Konverzia na malé písmeno
    }
    printf("Konvertovaný znak: %c\n", c); // Výpis konvertovaného znaku
    return 0;
}
```

- Znaký **128 - 255** nie sú štandardizované a môžu sa líšiť medzi rôznymi systémami.
- Použitie týchto znakov môže viesť k nepredvídateľnému správaniu na rôznych platformách!

Určte ako definujeme v jazyku c string?

- V jazyku C **neexistuje** špeciálny dátový typ pre reťazce.
- Reťazec je **pole znakov** ukončené nulovým znakom '`\0`'.
- Deklarácia: `char retazec[50];`
- Inicializácia: `char retazec[] = "Tretie cvičenie";`
- **Veľkosť poľa musí byť o jeden znak väčšia ako dĺžka reťazca kvôli nulovému znaku.**
- K znakom pristupujeme pomocou **indexu**, napr. `retazec[0]` pre prvý znak.
- Výpis reťazca: `printf("Reťazec: %s\n", retazec);`
- Vstup reťazca: `char retazec[4]; // 3 znaky + 1 pre znak \0`
`scanf("%3s", retazec); // nepíšeme & !`
- Pamäť:

R	e	t	a	z	e	c	\0
---	---	---	---	---	---	---	----

- **strlen**: Zistí dĺžku reťazca.
- **strcpy**: Kopíruje jeden reťazec do druhého. Po **inicializácii** reťazca **už nie je možné vykonať priradenie pomocou operátora =** !
- **strcat**: Pripojí jeden reťazec na koniec druhého.
- **strcmp**: Porovnáva dva reťaze.

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

int main() {
    char retazec1[50] = "Teraz vznikne";
    char retazec2[50] = " dlhšia veta.";
    strcat(retazec1, retazec2); // Pripojenie retazec2 na koniec retazec1
    printf("Spojený reťazec: %s\n", retazec1);
    return 0;
}
```

- Ak reťazec presiahne veľkosť poľa, môže dôjsť k prepisu pamäte.
- Príklad bezpečnej manipulácie:

```
strncat(retazec1, retazec2, sizeof(retazec1) -  
strlen(retazec1) - 1);
```
- Každý reťazec musí byť ukončený nulovým znakom, aby funkcie pracujúce s reťazcami vedeli, kde reťazec končí.
- Funkcia `strlen` vracia dĺžku reťazca bez nulového znaku.
- Funkcia `strcmp` porovnáva dva reťazce a vracia 0, ak sú rovnaké, záporné číslo, ak je prvý reťazec menší, a kladné číslo, ak je väčší.
- Pre prácu s reťazcami, ktorých veľkosť nie je vopred známa, je možné použiť dynamickú alokáciu pamäte pomocou `malloc` a `free` (cvičenie 9).






Akým spôsobom pracujeme v jazyku C s argumentmi príkazového riadku?

- Argumenty príkazového riadku sú hodnoty odovzdané programu **pri jeho spustení** na rozdiel od **vstupov**, ktoré si program načíta **za behu**.
- Syntax hlavičky: `int main(int argc, char *argv[])`, kde
 - **argc** – celé číslo udávajúce počet argumentov programu.
 - **argv** – pole textových reťazcov (dvojrozmerné pole znakov) obsahujúcich jednotlivé argumenty a `argv[0]` obsahuje názov programu.
- Píšeme ich pri spúšťaní za meno programu: `./main arg_1 arg_2 arg_3`
- Na prevod reťazcov (argumentov) môžeme použiť funkcie z knižnice **stdlib.h**
 - **atoi** – konvertuje reťazec na celé číslo
 - **atof** – konvertuje reťazec na `double`

```
for (int i = 0; i < argc; i++)  
    printf("%i. arg: %s\n", i, argv[i]);
```





Výsledok výpisu:

```
0. arg: {path}\excercises.exe  
1. arg: arg1  
2. arg: arg2  
3. arg: arg3
```

1. **Zadanie:** Napíšte program, ktorý načíta reťazec o maximálnej dĺžke 42 znakov zo vstupu a vypíše ho na obrazovku. **Riešenie:** ().
2. **Zadanie:** Napíšte program, ktorý načíta reťazec zo vstupu a vypočíta jeho dĺžku. **Riešenie:** ()
3. **Zadanie:** Napíšte program, ktorý načíta reťazec zo vstupu a spočíta počet abecedných a číselných znakov. **Riešenie:** ().
4. **Zadanie:** Napíšte program, ktorý načíta reťazec zo vstupu a konvertuje všetky veľké písmená na malé. **Riešenie:** ().
5. **Zadanie:** Napíšte program, ktorý načíta reťazec a znak zo vstupu a nahradí všetky výskyty tohto znaku v reťazci znakom ' _ '. **Riešenie:** ().

Ďalšie informácie – Reťazce v jazyku C sú polia znakov, ktoré končia špeciálnym znakom ' \0 '.

- Cyklus while sa dá použiť na prechod cez reťazec (test na ' \0 ').
- Dĺžka reťazca je počet znakov pred koncovým znakom ' \0 '.
- Abecedné znaky sú znaky od 'A' do 'Z' a od 'a' do 'z'. Číselné znaky sú znaky od '0' do '9'.
- ASCII tabuľka obsahuje ordinálne hodnoty znakov, ktoré môžete použiť na konverziu (rozdiel medzi 'a' a 'A').
- Nezabudnite na medzeru pred %c v scanf, ak chcete spotrebovať zvyšný znak nového riadku.

6. **Zadanie:** Napíšte program, ktorý načíta dva reťazce zo vstupu a porovná ich.
Riešenie: ().
7. **Zadanie:** Napíšte program, ktorý načíta argumenty príkazového riadku a vypíše prvý argument. **Riešenie:** ().
8. **Zadanie:** Napíšte program, ktorý načíta tri argumenty príkazového riadku a konvertuje ich na rôzne dátové typy. **Riešenie:** ().
9. **Zadanie:** Napíšte program, ktorý vypíše všetky argumenty príkazového riadku.
Riešenie: ().

Ďalšie informácie:

- Ak sú oba reťazce identické, dosiahnete koncový znak '`\0`' v oboch reťazcoch súčasne.
- `argv[0]` obsahuje cestu k programu, takže prvý argument je `argv[1]`.
- Použite funkcie `atoi` a `atof` na konverziu reťazcov na celé čísla a desatinné čísla.
- Môžete použiť cyklus `while` na prechádzanie všetkých argumentov a ich výpis.