STRINGOVI I FUNKCIJE ČIJI SU ARGUMENTI STRINGOVI

Stringovi su zapravo nizovi koji su tipa ***char*** i deklarisanje stringova će zapravo biti vrlo slično deklarisanju nizova tipa ***int*** i ***float,*** samo se jednom malom razlikom**.** Prilikom deklaracije stringova nećemo zadavati tačnu dužinu nizova, već ćemo zadati fiksnu dimenziju (najčešće 100) i time rezervisati da dužina stringa bude najviše 100 karaktera. Na kraju samog kraj stringa određen je karakterom **'\0'** i pod dužinom stringa podrazumevaćemo broj karaktera do **'\0'.** Pomoću sledećeg koda deklarišemo jedan string za koji je retervisano najviše 100 mesta.

```
char string[100];
```

Za razliku od nizova tipa ***int*** ili ***float***, gde smo koristili ***for*** petlju i ***scanf*** funkciju, kod stringova ćemo i učitavanje i štampanje vršiti znatno lakše. Za učitavanje stringa pod nazivom koristićemo ***string1*** funkciju ***gets()*** i kod

```
gets(string1);
```

Da bismo štampali string možemo koristiti ***printf ()*** funkciju rezervisanu oznaku ***s*** za string prilikom štampanja. Recimo kod koji nam je potreban da bismo štapali prethodno uneti string pod nazivom koristićemo ***string1*** koristićemo kod

```
printf("%s",string1);
```

Da bismo izvršili rekapitulaciju prethodno navedenog rešavamo naredni zadatak.

1. **Napisati program pomoću koga učitavamo string i štampamo.**

Rešenje:

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


int main(){
    char str[100];
    printf("Unesite string: ");
    gets(str);
    printf("Uneti string: %s",str);
    return 0;
}
```

.

Takođe, bitno je napomenuti da je tip ***char*** takođe celobrojni tip promenljive gde se svaki karakter poistovećuje sa odgovorajućim svojim ASCII kodom. U nastavku možemo videte tabelu u kojoj su dati karakteri i njihovi odgovarajući ASCII kodovi (posmatramo kolone sa crvenim ivicama).

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
nul	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
soh	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
stx	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
etx	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
eot	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
enq	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
ack	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
bel	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
bs	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
tab	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
lf	10	0012	0x0A	*	42	0052	0x2A	J	74	0112	0x4A	j	106	0152	0x6A
vt	11	0013	0x0B	+	43	0053	0x2B	K	75	0113	0x4B	k	107	0153	0x6B
ff	12	0014	0x0C	,	44	0054	0x2C	L	76	0114	0x4C	l	108	0154	0x6C
cr	13	0015	0x0D	-	45	0055	0x2D	M	77	0115	0x4D	m	109	0155	0x6D
so	14	0016	0x0E	.	46	0056	0x2E	N	78	0116	0x4E	n	110	0156	0x6E
si	15	0017	0x0F	/	47	0057	0x2F	O	79	0117	0x4F	o	111	0157	0x6F
dle	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
dc1	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
dc2	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
dc3	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
dc4	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
nak	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
syn	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
etb	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
can	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
em	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
sub	26	0032	0x1A	:	58	0072	0x3A	Z	90	0132	0x5A	z	122	0172	0x7A
esc	27	0033	0x1B	;	59	0073	0x3B	[	91	0133	0x5B	{	123	0173	0x7B
fs	28	0034	0x1C	<	60	0074	0x3C	\	92	0134	0x5C	|	124	0174	0x7C
gs	29	0035	0x1D	=	61	0075	0x3D	]	93	0135	0x5D	}	125	0175	0x7D
rs	30	0036	0x1E	>	62	0076	0x3E	^	94	0136	0x5E	~	126	0176	0x7E
us	31	0037	0x1F	?	63	0077	0x3F	_	95	0137	0x5F	(del)	127	0177	0x7F

Tabela sa ASCII kodovima

U narednim zadacima videćemo upravo interpretaciju navedenih tvrdnji.

2. **Napisati funkciju koja sva slova u stringu pretvara u mala. Zatim je testirati u glavnom delu programa.**

Rešenje:

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

void toLower(char str[100])
{
    int i = 0;
    while(str[i]!='\0'){
      if('A'<=str[i] && str[i]<='Z')
{
          str[i] = str[i]+32;
      }
      i++;
   }
```

```
}
int main()
{   printf("Unesite string: \n");
    char unos[100];
    gets(unos);
    toLower(unos);
    printf("Uneti string posle poziva funkcije toLower: %s", unos);
    return 0;
}
```

3. **Napisati funkciju koja sva slova u stringu pretvara u velika. Zatim je testirati u glavnom delu programa.**

Rešenje:

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

void toUpper(int n, char str[n])
{
    int i = 0;
    while(str[i]!='\0'){
       if('a'<=str[i] && str[i]<='z')
      {
           str[i] = str[i]-32;
       }
       i++;
    }
}
int main()
{   printf("Unesite string: \n");
    char unos[100];
    gets(unos);
    toUpper(unos);
    printf("Uneti string posle poziva funkcije toUpper: %s\n", unos);
    return 0;
}
```

Obratite pažnju da prilikom deklarisanja funkcija u prethodna dva zadatka nigde nismo iskoristili dužinu stringa, već smo koristili da se na kraju stringa nalazi karakter **'\0'.** Tačnije, mi smo koristili petlju i uz navedeni uslov proveravali da li član niza koji želimo trenutno da modifikujemo zapravo kraj stringa. Na ovaj način izbegavamo da koristimo tačnu dužinu stringa koju još uvek ne znamo odrediti.

Pre nego definišemo funkciju pomoću koje ćemo određivati dužinu stringa uradićemo još jedan zadatak.

4. **Napisati funkciju koja modifikuje string tako da svako slovo na neparnom mestu menja u veliko, a svako slovo na parnom mestu u stringu u malo. Zatim je testirati u glavnom delu programa.**

Rešenje:

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

void toLower(char str[100])
{
    int i=0;
    while(str[i]!='\0'){
       if('A'<=str[i] && str[i]<='Z'){
            str[i]=str[i]+32;
       }
       i++;
    }
}
void toUpper(char str[100])
{
    int i=0;
    while(str[i]!='\0'){
       if('a'<=str[i] && str[i]<='z')
      {
            str[i]=str[i]-32;
       }
       i++;
    }
}

void toUpperOdd(char str[100])
```

```
{
    toLower(str);
    int i = 0;
    while(str[i]!='\0'){
       if('a'<=str[i] && str[i]<='z'){
            str[i] = str[i]-32;
       }
       i=i+2;
    }

}
int main()
{   printf("Unesite string: \n");
    char unos[100];
    gets(unos);
    toUpperOdd(unos);
    printf("Uneti string posle poziva funkcije toUpperOdd: %s\n", unos);
    return 0;
}
```

Sada ćemo kreirati funkciju koja određuje dužinu stringa. Potreba za ovakvom funkcijom leži u činjenici da jedan string možemo deklarisati tako da za njega bude rezervisan određeni broj karaktera (na primer 100), ali se zapravo ne moraju uneti tačno toliko karaktera, već određeni deo može ostati prazan. Samim tim pod dužina stringa podrazumevaćemo broj karaktera do **'\0'**, kao što smo već i naveli na početku.

5. **Napisati funkciju koja određuje dužinu stringa, zatim je testirati u glavnom delu programa.**

Rešenje:

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


int strlen(char s[100])
{ int i=0;
while( s[i]!='\0' )
i++;
return i;}
int main()
{   printf("Unesite string: \n");
    char unos[100];
```

```
    gets(unos);
    printf("Duzina stringa je %d.\n", strlen(unos));
    return 0;
}
```

Za kraj uradićemo još jedan zadatak gde ćemo iskoristiti funkciju ***strlen***.

6. **Napisati funkciju koja string okreće naopačke. Zatim je testirati u glavnom delu programa.**

Rešenje:

```
#include <stdio.h>
#include <stdlib.h>
int strlen(char s[100])
{ int i=0;
while( s[i]!='\0' )
i++;
return i;}
void Naopacke(char str[100])
{   int i;
    for(i=strlen(str)-1;i>=0;i--)
        s[strlen(str)-1-i]=str[i];
    for(i=0;i<=strlen(str);i++)
        str[i]=s[i];
}

int main()
{
    printf("Unesite string: \n");
    char unos[100] ;
    gets(unos);
   Naopacke(unos);
    printf("Uneti string posle poziva funkcije Naopacke: %s\n", unos);
    return 0;
}
```