

Računske vježbe 4

Programiranje I

1. Učitati string koji sadrži manje od 20 karaktera i sva mala slova u tom stringu konvertovati u velika, a ostale karaktere ne mijenjati.

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int main()
5 {
6     char str[20];
7     int i;
8     //const int diff = 'a' - 'A';
9
10    puts("Unesite jednu rijec:");
11    scanf("%s", str);
12
13    for (i = 0; i < strlen(str); i++) // uslov je mogao biti i str[i] != '\0'
14    {
15        if(str[i] >= 'a' && str[i] <= 'z')
16            str[i] -= 'a' - 'A'; // str[i] -= diff;
17    }
18    printf("Novi string je: %s", str);
19 }
```

String predstavlja niz karaktera i deklarira se kao i svaki drugi niz. Specifično za string jeste što se na njegovom kraju nalazi **terminacioni karakter**, `'\0'`, pomoću kojeg kompajler zna gdje se u memoriji završava string. Ovaj karakter se automatski dodaje na kraj stringa prilikom njegovog učitavanja, bez uticaja programera. Funkciju `strlen()`, koja je definisana u zaglavlju `string.h`, ćemo često koristiti u radu sa stringovima, jer nam omogućava da prebrojimo karaktere stringa, **ne uključujući terminacioni karakter**.

U ovom zadatku neophodno je konvertovati mala slova u velika. Uvidom u ASCII tabelu (poslednja strana dokumenta) vidimo da se velika slova nalaze u redu od A do Z, baš kao i mala od a do z. Ovu činjenicu možemo iskoristiti na dva načina. Prvo, sa sigurnošću možemo tvrditi da je karakter malo slovo ako se nalazi između a i z. Drugo, razlika između malog slova a i velikog slova A jednaka je razlici za sve ostale parove malog i velikog slova u ASCII tabeli. Drugim riječima, proizvoljno malo slovo iz stringa ćemo konvertovati u veliko na sljedeći način:

```
str[i] -= 'a' - 'A';
```

gdje u konkretnom zadatku ovu razliku možemo sačuvati u posebnu promjenljivu kako se razlika ne bi bespotrebno računala onoliko puta koliko string ima karaktera. Uočite da smo u tom slučaju koristili **const** kvalifikator tipa čime smo deklarirali da promjenljiva zapravo predstavlja konstantnu vrijednost koja se nikad neće promijeniti. Mi znamo da se ova razlika ne može, ali i **ne smije** mijenjati pa je samim tim proglašavamo konstantnom. Pokušaj promjene doveo bi do greške. Ovaj pristup predstavlja dobru programersku praksu.

2. Napisati program koji učitava string, koji predstavlja rečenicu, i koji koristeći funkciju provjerava da li je rečenica ispravno zadata. Ispravno zadata rečenica zadovoljava sljedeća pravila:
- a) Rečenica mora početi velikim slovom i završiti se tačkom.
 - b) Riječi su proizvoljni podstringovi koji mogu sadržati samo mala slova.
 - c) Riječi mogu biti razdvojene **jednim** razmakom (SPACE), zarezom ili tačka-zarezom.

U glavnom programu učitati string, pozvati funkciju i ispisati obavještenje da li je rečenica ispravno zadata.

```
1 #include <stdio.h>
2
3 int sentence(char *);
4
5 int main()
6 {
7     int p;
8     char s[20];
9     printf("Unesite recenicu: ");
10    gets(s); // koristimo gets jer se ne zaustavlja na space karakteru kao scanf
11    p = sentence(s);
12    if(p == 1) printf("Jeste");
13    else printf("Nije");
14 }
15
16 int sentence(char *s)
17 {
18     int i, length, ind = 1;
19     length = strlen(s);
20     if((s[0] < 'A' || s[0] > 'Z') || s[length - 1] != '.' ) ind = 0;
21     if(ind == 1)
22     {
23         for(i = 1; i < length-1; i++)
24         {
25             if((s[i] < 'a' || s[i] > 'z') && s[i] != ' ' && s[i] != ','
26             && s[i] != ';') ind = 0;
27             if((s[i] == ' ' || s[i] == ',' || s[i] == ';')
28             && (s[i + 1] == ' ' || s[i + 1] == ',' || s[i+1] == ';')) ind = 0;
29         }
30     }
31     return ind;
32 }
```

U programskom jeziku C navođenje prototipa funkcije nije obavezno. Međutim, da bismo osigurali provjeru poziva funkcije, odnosno da spriječimo poziv funkcije sa neočekivanim argumentima, navešćemo prototip funkcije prije njenog definisanja i prije funkcije main. Prototip funkcije je sličan njenom zaglavlju, s tim što se ne moraju navoditi imena parametara. Uopšteno:

```
tip_podatka ime_funkcije(tip_1, tip_2, ..., tip_n);
```

te sada program zna tipove ulaznih argumenata funkcije i koji tip podatka vraća. Odgovor na šta i kako funkcija radi sadržan je u njenom tijelu:

```
tip_podatka ime_funkcije(tip_1 par_1, tip_2 par_2, ..., tip_n par_n)
{
    //tijelo funkcije
}
```

gdje sa `tip_podatka` definišemo tip vrijednosti funkcije odnosno njenog rezultata. Ukoliko funkcija ima rezultat (tip vrijednosti funkcije koja nema rezultat je `void`) on se prosljeđuje upotrebom naredbe `return` koja predstavlja tačku izlaska iz funkcije. Znajući ovo, funkciju smo mogli napisati i bez pomoćne indikatorske promjenljive:

```
int sentence(char *s)
{
    int i, length;
    length = strlen(s);
    if((s[0] < 'A' || s[0] > 'Z') || s[length - 1] != '.' ) return 0;
    {
        for(i = 1; i < length-1; i++)
        {
            if((s[i] < 'a' || s[i] > 'z') && s[i] != ' ' && s[i] != ','
            && s[i] != ';') return 0;
            if((s[i] == ' ' || s[i] == ',' || s[i] == ';')
            && (s[i + 1] == ' ' || s[i+1] == ',' || s[i+1] == ';')) return 0;
        }
    }
    return 1;
}
```

Sjetimo se da je i `main` funkcija. Kako to da ona ne vraća rezultat (ne pišemo `return` na njenom kraju), a jasno smo naglasili da je tip njene vrijednosti `int`? Odgovor leži u standardu programskog jezika C koji naglašava da ako se kontrola toka nađe na kraju funkcije `main`, a da ne naiđe na ključnu riječ `return`, efektivno se izvršava `return 0`. Eto, misterija riješena.

Funkciji ne možemo proslijediti niz, ali joj možemo proslijediti pokazivač na njega. U slučaju stringa, ne moramo prosljeđivati njegovu dužinu jer je ona implicitno sadržana u njemu zbog prisustva terminacionog karaktera.

3. Napisati program koji sadrži funkciju koja određuje i štampa srednju vrijednost niza realnih brojeva. Broj članova, kao i vrijednost članova niza se zadaju po startovanju programa.

```
1 #include <stdio.h>
2
3 double mean_value(double *, int);
4
5 int main()
6 {
7     int length, i;
8     double array[20];
9     puts("Unesite broj clanova niza");
10    scanf("%d", &length);
11    puts("Unesite elemente niza");
12    for(i = 0; i < length; i++)
13        scanf("%lf", &array[i]);
14    printf("Srednja vrijednost niza je: %lf\n", mean_value(array, length));
15 }
16
17 double mean_value(double *array, int length)
18 {
19     int i;
20     double sum = 0.0;
21     for(i = 0; i < length; i++)
22         sum += array[i];
23     return sum / length;
24 }
```

4. Napisati program koji učitava prirodan broj N i koji štampa prvih N brojeva čiji je zbir cifara prost broj. Prost broj je prirodan broj veći od 1, djeljiv samo brojem 1 i samim sobom. Program sadrži dvije funkcije. Jednu koja određuje zbir cifara i drugu koja određuje da li je broj prost ili ne.

```
1 #include <stdio.h>
2
3 int prime_number(int);
4 int sum_of_digits(int);
5
6 int main()
7 {
8     int n, k = 0;
9     int i = 1;
10    printf("Unesite broj N:\n");
11    scanf("%d", &n);
12    while(k < n)
13    {
14        if(prime_number(sum_of_digits(i)))
15        {
16            k = k + 1;
17            printf("%d\n", i);
18        }
19        i = i + 1;
20    }
21 }
22
23 int sum_of_digits(int num)
24 {
25     int sum = 0;
26     while (num != 0)
27     {
28         sum += num % 10;
29         num /= 10;
30     }
31     return sum;
32 }
33
34 int prime_number(int num)
35 {
36     int i;
37     if (num < 2) return 0;
38     for(i = 2; i * i <= num; i++)
39         if (num % i == 0)
40             return 0;
41     return 1;
42 }
```

ASCII TABLE

Decimal Hex Char			Decimal Hex Char			Decimal Hex Char			Decimal Hex Char		
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]