5. Tip podatka `char i string`

Bez obzira na jezik u kojem je pisan, svaki program sastoji se od niza naredbi koje mijenjaju vrijednost objekata pohranjenih u memoriji računala. Računalo dijelove memorije u kojim a su smješteni objekti razlikuje pomoću pripadajuće memorijske adrese. Da programer tijekom pisanja programa ne bi morao pamtiti memorijske adrese, svi programski jezici omogućavaju da se vrijednost objekata dohvaćaju preko simboličkih naziva razumljivih ljudima. Gledano iz perspektive računala, objekt je samo dio memorije u koji je pohranjena njegova vrijednost u binarnom obliku. *Tip* objekta, između ostalog, određuje raspored bitova prema kojem je taj objekt pohranjen u memoriju.

Da bi prevoditelj pravilno preveo naš izvorni C kod u strojni jezik, svaku varijablu treba prije njena korištenja u kodu *deklarirati*, odnosno jednoznačno odrediti njen tip. Znakovne konstante tipa `char` pišu se uglavnom kao samo jedan znak unutar jednostrukih znakova navodnika:

```
char SlovoA = 'a';
```

Za znakove koji se ne mogu prikazati na zaslonu koriste se posebne sekvence (*engl. escape sequence*) koje počinju lijevom kosom crtom (*engl. backslash*). U tablici 1. su navedeni posebni znakovi.

Tablica 1. Posebni znakovi

\n	novi redak
\t	horizontalni tabulator
\v	vertikalni tabulator
\b	pomak za mjesto unazad (*backspace*)
\r	povratak na početak reda (*carriage return*)
\f	nova stranica (*form feed*)
\a	zvučni signal (*alert*)
\\	kosa crta u lijevo (*backslash*)
\?	upitnik
\'	jednostruki navodnik
\“	dvostruki navodnik
\0	završetak znakovnog niza
\ddd	znak čiji je kod zadan oktalno s 1, 2 ili 3 znamenke
\xddd	znak čiji je kod zadan heksadekadski

Želimo li pospremiti riječ, rečenicu ili čak veći tekst, u C-u ne postoji odgovarajući tip. U tu svrhu koriste se nizovi znakova `string`. Stringovne konstante navodimo kao nizove znakova između dvostrukih navodnika. Na primjer:

```
"Pero Sapun"
```

je string od 10 znakova (9 slova i 1 razmak). Za razliku od stringova, znakovne konstante se navode između jednostrukih navodnika. Tako je "A" `string`, a 'A' `char`. Ova razlika je izuzetno bitna, jer ne možemo koristiti znakove umjesto stringova (niti obrnuto). To bi bilo kao da pokušamo koristiti int umjesto niza brojeva (ili obrnuto).

Koliko string zauzima memorije?

String `"Pero Sapun"` *ima 10 slova, ali u memoriji zauzima 11 mjesta! Prisjetimo "običnih" nizova: postoji memorija dodijeljena nekom nizu (statički ili dinamički), ali nigdje nije pohranjeno koliko elemenata se stvarno nalazi u nizu (tj. koji dio niza zaista koristimo). U tu svrhu uvodili smo pomoću varijablu (najčešće smo ju zvali n) u kojoj smo čuvali broj elemenata niza.*

Kod stringova, kao kraj niza se koristi (nevidljivi) znak, tzv. ***null-character*** *('\0'), čija je ASCII vrijednost nula. Sve funkcije koje barataju sa stringovima upotrebljavaju taj znak kao oznaku za kraj stringa.*

Ako stringu pristupamo direktno, kao nizu znakova (umjesto pomoću specijalnih stringovnih funkcija), moramo paziti na označavanje završetka!

Kod ispisa stringova naredbom `printf()` koristi se format `%s`. Učitavanje je nešto složenije: format `%s` označava učitavanje jedne riječi. Želimo li pročitati cijeli redak (tj. string do prvog skoka u novi red), treba upotrijebiti format `%[^\n]` ili funkciju `gets()`.
Općenito, ako želimo učitavati dijelove teksta odvojene znakovima, na primjer, 'a', 'b' i 'c' onda navodimo format `%[^abc]`. Prilikom takvog učitavanja, na primjer, teksta `"Ovaj kolegij je bas zanimljiv!"`, C će razlikovati "riječi": `"Ov"`, `"j kolegij je "`, `"s z"`, `"nimljiv!"`.
Dakle, format `%s` je samo pokrata za `%[^ \t\n]` (jer kao separatore riječi uzima razmake, TAB-ove i skokove u novi red.)
Nakon čitanja riječi, razmak kojim riječ završava ostaje nepročitan. No, iduće učitavanje (s istim formatom) će taj razmak zanemariti, te se on neće naći niti u jednoj od učitanih riječi.

Zadatak 1.

Napišite program koji učitava jednu riječ duljine najviše 17 znakova, te ispisuje:

a) *tu riječ*
b) *tu riječ bez prvog znaka*
c) *treće slovo te riječi (pretpostavite da je riječ dovoljno dugačka)*

Rješenje. Prisjetimo se da je niz isto što i pokazivač na prvi element (dakle na početak) niza! Kako naredba `scanf()` mora primati adrese na koje se spremaju podaci, ispred stringovnih varijabli NE STAVLJAMO "`&`"! Također, ako imamo stringovnu varijablu `rijec` (a ona je, kako smo već rekli, ekvivalentna `&rijec[0]`), onda `&rijec[1]` predstavlja adresu drugog znaka. Nastavimo li čitati znakove od drugog znaka do `'\0'`, dobit ćemo upravo riječ bez prvog znaka (rješenje podzadatka b)). Treće (ili bilo koje drugo) slovo je znak, dakle tipa `char`, i ispisuje se pomoću formata `%c`.
Prilikom deklaracije stringa fiksne duljine, kao i kod dinamičke alokacije stringa, moramo uvijek ostaviti jedno mjesto "viška" u kojem će biti pospremljen završni znak `'\0'`. U našem zadatku, riječ može imati najviše 17 znakova, što znači da nam za pohranu te riječi u memoriji treba niz od 17 + 1 = 18 znakova.

Kod:

```
#include <stdio.h>

int main () {
      char rijec[18];

      printf("Upisite rijec: " );
      scanf("%s ", rijec);

      printf("a) rijec: %s \n", rijec);
      printf("b) rijec bez prvog znaka : %s \n", &rijec[1]);
      printf("c) treci znak : %c \n", rijec[2]);

      return 0 ;
 }
```

Izvedite program na računalu tako da upišete dvije riječi (npr. "Pero Sapun"). Pod a), program će ispisati samo prvu riječ, dok druga (bez razmaka) ostaje nepročitana! Ako programski kod kopirate tako da se izvrši dva puta, onda će drugi `scanf()` učitati drugu riječ, bez čekanja na unos.

Usporedba znakova.

Zanimljivo je da se `char` konstante i varijable mogu uspoređivati, poput brojeva, pri čemu se u biti uspoređuju njihovi brojčani ekvivalenti u nekom od standarda:

```
char a='a', b='b';
printf("%d",a<b);
```

Najraširenija je ASCII niz (*American Standard Code for Information Interchange*) u kojem su svim ispisivim znakovima, brojevima i slovima engleskog alfabeta pridruženi brojevi od 32 do 127, dok specijalni znakovi imaju kodove od 0 do 31 uključivo. U prethodnom primjeru se ispisuje 1, jer je ASCII kod malog slova a (97) manji od koda za malo slovo b (98).

Znakovi se mogu uspoređivati sa cijelim brojevima, pri čemu se oni pretvaraju u cjelobrojni kodni ekvivalent. Naredbom

```
printf("%d",32==' ');
```

ispisat će se broj 1 jer je ASCII kod praznine upravo 32.

Zadatak za vježbu:

1. Napisati C program koji će u zadanom stringu koji se učita sa tipkovnice pronaći koliko ima samoglasnika.
2. Napisati C program koji će u zadanom stringu pronaći onaj znak koji ima najveću ASCII vrijednost. Ispisati koja je to najveća vrijednost na ekran.
3. Napisati C program koji će učitati 2 stringa duljine najviše 20 znakova. Program treba izračunati i na ekran ispisati koliko su ta 2 stringa slična u postocima.