Programiranje I

Nizovi (vektori)

Stringovi

Funkcije

Nizovi (vektori)

- Deklaracija niza se obavlja kao:

  tip a[5];

  gdje je tip neki od, recimo, osnovnih tipova (float, int, double, itd.).

- Na ovaj način je zauzeta memorija za 5 promjenljivih i to redom jedna za drugom.

- Te promjenljive se nazivaju a[0], a[1], a[2], a[3] i a[4] **(obratite pažnju na indekse)** i sa tim promjenljivim su dozvoljene sve operacije kao sa datim tipom, npr. **a[1]+=a[3]-a[4];**

Nizovi

- Adrese članova niza su: &a[0], &a[1], &a[2], &a[3] i &a[4], ali postoji i alternativni oblik a, a+1, a+2, a+3 i a+4.
- Ime niza je početna adresa niza (adresa prvog člana niza ili, ponekad se kaže, pristupna adresa nizu).
- Uočite da **a+1** nije adresa narednog bajta u memoriji, već adresa narednog člana niza.
- Ovo otvara mogućnost da se članovima niza pristupa i kao: ***(a+1)=3;** čime bi se drugi član niza (onaj sa indeksom 1) postavio na 3.
- Uočite da ***a+1** ne znači **a[1]** već **a[0]+1** zbog većeg prioriteta * u odnosu na +.

Nizovi i pokazivači

- Veliki broj obrada nizova se sastoji u obilasku svih članova uz njihovo korišćenje ili mijenjanje po nekom kriterijumu.
- Pokazivači su izuzetno pogodni za obilazak jer se mogu koristiti operatori inkrementiranja i dekrementiranja.
- Kod nizova puni smisao ima korišćenje pokazivača i primjena na njih raznih operacija koje su pomenute na prethodnom času. Na primjer:
  tip *b;
  b=a+3;
- Sada b pokazuje na član niza sa indeksom 3, razlika b-a iznosi 3 (opet nije broj bajtova, već broj elemenata toga tipa između članova određenih pokazivačima a i b).

Višedimenzioni nizovi

- Deklaracija 2D niza se obavlja kao:
  float m[4][5];
  gdje m predstavlja niz od 4 komponente od kojih svaka sadrži 5 elemenata tipa float.
- Dakle, u C-u: **matrica=niz nizova**.
- Elementi niza **m** su: **m[0]**, **m[1], m[2]** i **m[3]**, gdje je **m[0]** zapravo adresa (**uslovno govoreći**) nulte vrste matrice, **m[1]** adresa prve vrste itd.
- Podržan broj dimenzija višedimenzionog niza zavisi od kompajlera.
- Ako se deklaracije nizova i matrica obogate pokazivačima može da nastane prava zbrka vezana za to koji je zapravo tip podataka deklarisan.

Indeksiranje članova niza

- Ako imate deklarisan niz:

  int a[7];

  a negdje u programu koristimo a[10] neće doći do prekida rada programa, a vjerovatno ni do bilo kakvog upozorenja, što znači da se nesmetano može pristupiti memorijskim lokacijama van niza, tj. gdje su neke druge promjenljive ili dio sistemskog koda. Čak se u nekim situacijama mogu koristiti i negativni indeksi, npr. a[-5].

- Nije potrebno naglašavati da se ove situacije moraju izbjeći.

Inicijalizacija nizova

- Nizovi se mogu inicijalizovati zajedno sa deklarisanjem (definisati) kao:
  int a[5]={0,1,2,3,4};
  čime smo redom članove niza postavili na vrijednosti navedene u vitičastim zagradama. Dozvoljena je varijanta:
  int a[5]={0,1,2};
  ali se sada eksplicitno neinicijalizovane vrijednosti (a[3] i a[4]) niza postavljaju na 0.
- Varijanta:
  int a[5]={0};
  postavlja sve članove niza na 0.

Inicijalizacija nizova

- Dozvoljeno je:
  int a[]={0,1,2,3,4};
  čime se implicitno zauzima 5 pozicija za cijele brojeve.
- Kod inicijalizacije
  int a[2]={0,1,2,3,4};
  samo se prva dva člana niza inicijalizuju, ostali brojevi u zagradama se zanemaruju.
- Kod višedimenzionih nizova inicijalizacija se može obaviti sa:
  int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
  int a[ ][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
- Prvi metod je, nadam se, jasan, dok smo drugim metodom implicitno zauzeli potreban broj vrsta.

Inicijalizacija nizova

- Dozvoljena je i inicijalizacija tipa:
  int a[ ][4]={1,2,3,4,5,6,7,8,9,10,11,12};
  jer će se podaci sada smiještati vrstu po vrstu, a računar ima podatak koliko je elemenata u svakoj vrsti.
- Od standarda C99, koristi se označeni inicijalizator:
  int x[7]={[2]=9,[4]=12}; // ostali 0
  int x[]={[2]=9,[4]=12}; // maksimalni index je 4
- Nije dozvoljena inicijalizacija oblika:
  int a[ ][ ]={1,2,3,4,5,6,7,8,9,10,11,12};
  jer ne postoji podatak kako smiještati elemente iz vitičastih zagrada u matricu (tj. koliko elemenata ide u vrste).

Stringovi

- Niz karaktera (podataka tipa `char`) se naziva **string**.
- Sa stringovima smo se već upoznali kroz funkcije printf:

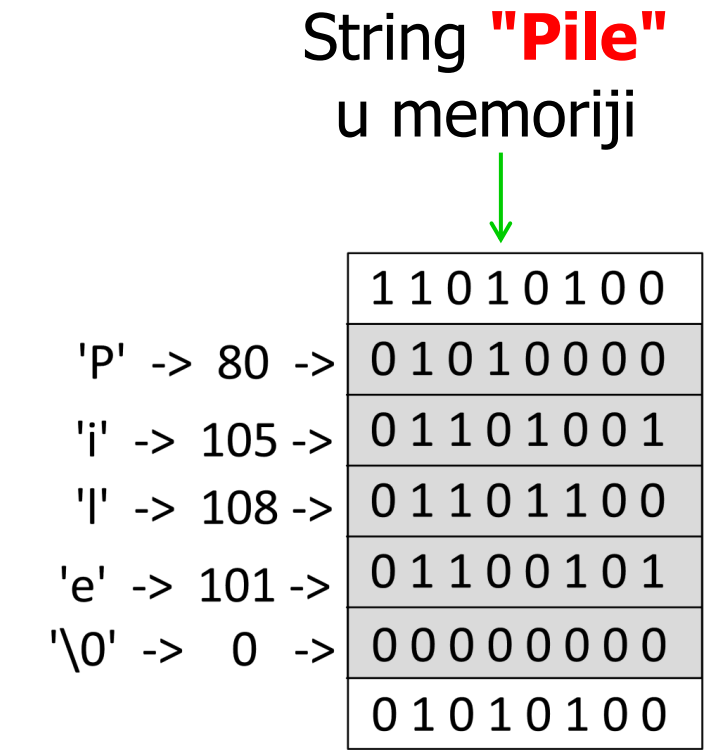
```
printf("Suma je %d", suma);
```

- String je niz karaktera pod znacima navoda - "Program".
- Za razliku od stringa, pojedinačni karakteri se navode koristeći apostrofe - 'P', 'r', ... , 'a', 'm'.
- String se deklariše kao svaki drugi niz, koristeći uglaste zagrade [ ] i broj elemenata unutar zagrada:

```
char s[30];
```

Terminacioni karakter

- Pored pojedinačnih karaktera, stringovi u C-u sadrže dodatni karakter **'\0'**. Ovaj karakter se nalazi na kraju stringa i naziva se **terminacioni karakter**.
- Na osnovu terminacionog karaktera, kompajler zna gde se završava string u memoriji.
- Terminacioni karakter se automatski dodaje na kraj stringa prilikom njegovog učitavanja, bez uticaja programera.
- Terminacioni karakter ima ASCII kod 0, ali to je manje važno.



String kao promjenljiva

- Prije nego pređemo na stringove kao promjenljive u obliku niza karaktera, objasnimo jedan detalj koji umije da zbuni.
- Funkcija koja se često koristi u radu sa stringovima **strlen("prvi cas")** i koja je definisana u zaglavlju string.h, daje rezultat **8**, odnosno ne broji terminacioni karakter, za razliku od **sizeof** koji mjeri memoriju potrebnu za smještaj stringa. U ovom slučaju, **sizeof("prvi cas")** vraća broj 9.
- Napominjemo da je **strlen** implementirana, kao i većina funkcija za rad sa stringovima, u okviru programske biblioteke string.h, koja mora biti uključena pomoću pretprocesora da biste imali pristup ovoj funkciji.

String kao promjenljiva

- Naravno, niz karaktera – string se može deklarisati kao i svaki niz na sljedeći način:
  char str[20];
- Inicijalizacija se može obaviti kao:
  char str[20]= "cert";
  char str[]= "cert";
  char str[]={'c', 'e', 'r', 't', '\0'};
- Koje vrijednosti će vratiti **sizeof(p)** i **strlen(p)**?
- Dozvoljena je i sljedeća inicijalizacija (spajanje stringova, pri čemu se stringovi razdvajaju bjelinom):

  char s[50] = "Dobar" "dan, " "kolega";

Inicijalizacija stringa preko pokazivača

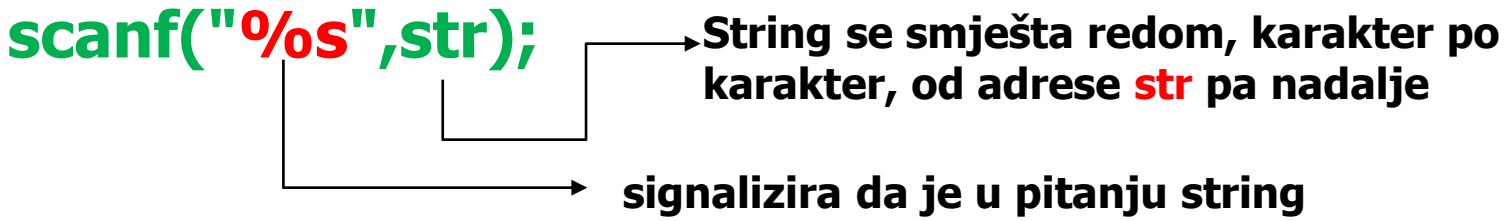
- Pokazivač na tip char se može inicijalizovati na sljedeći način:
  char *p = "Test";
- Na ovaj način je deklarisan pokazivač na read-only memoriju koja sadrži string literal **"Test"**.
- Pokušaj izmjene ovog stringa, na primjer

  p[0] = 'B';

  dovodi do greške prilikom kompajliranja.

Učitavanje i štampanje stringova

- Funkcijom scanf se može "odjednom" učitati string (ne kao kada su u pitanju nizovi - član po član). Sintaksa je:



- Funkcijom scanf string se učitava samo do prve bjeline tako da ova funkcija ne može služiti recimo za učitavanje imena i prezimena odjednom.
- Funkcijom printf štampa se string, pri čemu se u okviru stringa mogu štampati i bjeline, uključujući čak i znak za novi red.

Učitavanje i štampanje stringova

- Primjer funkcije printf kod štampanja stringova:

  printf("%s",str);

- Alternativno su u zaglavlju stdio.h implementirane funkcije:
  - **gets(str);** koja učitava string do znaka za novi red ne uključujući znak za novi red, a uključujući eventualne bjeline.
  - **puts(str);** koja štampa string **str** i automatski nakon štampanja prelazi u novi red, što ne radi printf.

Učitavanje i štampanje karaktera

- Funkcija **getch()** vraća rezultat koji je jednak karakteru koji se učitao sa tastature. Ova funkcija ne baferuje podatke koje učitava, odnosno ne čeka pritisak tastera Enter, već ono što učita odmah proslijedi u odgovarajuću promjenljivu:

```
char a;
a=getch();
```

- Funkcija putch(a) prikazuje na ekranu karakter a koji joj je argument.

Korisne funkcije iz string.h

- strcpy(tar,or) kopira string or **(od origin)** u string tar **(od target).**
- strcat(tar,or) nadovezuje string or na kraj stringa tar.
- strcmp(a,b) leksikografski poredi stringove a i b.
  - ako je string a veći od stringa b vraća pozitivan broj;
  - ako su stringovi jednaki vraća nulu;
  - ako je string a manji od stringa b vraća negativan broj.
  - String a je leksikografski veći od stringa b ako je:
    - ASCII kod prvog karaktera stringa **a** veći od ASCII koda prvog karaktera stringa b;
    - ukoliko su prvi karakteri isti porede se naredni.
    - Stringovi su jednaki ako su im svi karakteri jednaki.
  - ASCII kod terminacionog karaktera je manji od bilo kog drugog karaktera i u C-u je jednak 0.

Korisne funkcije iz string.h

- strstr(tar,or) traži podstring or u stringu tar i vraća pokazivač na prvo pojavljivanje takvog podstringa.
- strchr(tar,c) i strrchr (tar,c), slično prethodnom, traže prvo i posljednje pojavljivanje karaktera c u stringu tar, respektivno.
- strspn(tar,or) i strcspn(tar,or) prebrojavaju početne karaktere stringa or koji se pojavljuju u stringu tar, odnosno koji se ne pojavljuju u njemu, respektivno.
- Funkcije atoi(s), atol(s) i atof(s) pretvaraju broj sadržan u stringu s u int, long int i float broj, respektivno.

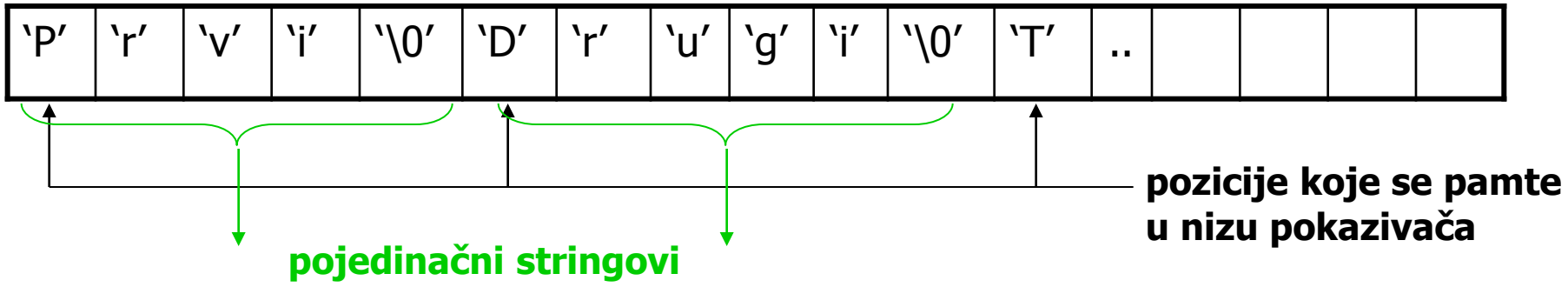
Rad sa mnoštvom stringova

- Aplikacije koje rade sa stringovima obično rade sa mnoštvom stringova.
- Svaki string se mora dimenzionisati na najveću očekivanu dužinu.
- Na primjer, baza prezimena se mora dimenzionisati na najveće očekivano prezime od, recimo, 15 slova, dok većina prezimena kod nas ima manje od 10.
- Još gora je varijanta za riječi opšte namjene, gdje se dimenzionisanje mora obaviti sa 20 i više slova, a realno mnogo riječi kao što su veznici imaju samo po nekoliko slova.

Rad sa mnoštvom stringova

- Da bi se izbjegla ova nepotrebna memorijska zahtjevnost uobičajeno se koristi sljedeći trik.
- U jednom nizu karaktera **(namjerno sad koristim ovaj pojam)** se drže svi stringovi odijeljeni terminacionim karakterima. Pozicije početaka pojedinih stringova se zatim pamte preko pokazivača.

Niz karaktera



Rad sa mnoštvom stringova

- Realizacija načina rada sa mnoštvom stringova nije stvar kompajlera, već programerske vještine.

```
#include<stdio.h>
int main(){
char s[1000],*p[200],temp[20];
int size=1,tp=0,i,j=0;
p[0]=s;
while(size) {
    gets(temp);
    size=strlen(temp);
    for(i=0;i<size;i++) s[tp+i]=temp[i];
    s[tp+size]='\0';
    tp+=size+1;
    if(size) p[++j]=s+tp;
}
for(i=0;i<j;i++)
printf("%s\n",p[i]);
}
```

nastavak gore

Rad sa mnoštvom stringova

- Objasnimo u par rečenica prethodni primjer.
- Deklarisali smo string i niz pokazivača na stringove.
- U while petlji ostajemo sve dok je učitani string dužine veće od nula, odnosno dok korisnik ne unese prazan string.
- Svaki uneseni string se pozicionira na odgovarajuće mjesto, uz ažuriranje promjenljive tp, koja pamti poziciju dokle se stiglo u "velikom stringu".
- Nakon unosa stringa postavlja se terminacioni karakter na odgovarajuće mjesto i podesi da odgovarajući pokazivač iz niza pokazivača pokaže na naredni string čiji se unos očekuje.
- Na kraju ovog demonstracionog programa smo iz "velikog stringa" štampali pojedinačne stringove.

Rad sa mnoštvom stringova

- Primjer proučite sami, a kao pomoć neka vam posluže boje koje povezuju objašnjenje sa djelovima koda.
- Program nije optimalan ni direktno praktično upotrebljiv, već je samo ilustrativni primjer.
- Često se u sličnim situacijama koristi činjenica da naredba scanf može da vrati rezultat koji je jednak broju bajtova koji su učitavaju:
  a=scanf("%s",str);
- Napominjemo da gotovo sve objašnjene naredbe imaju veći broj mogućnosti (kao što je slučaj kod scanf-a), ali ih mi rijetko koristimo.

Funkcije

- Podsjetite se osnovnih pojmova o potprogramima koje smo uveli na uvodnom času.
- U programskom jeziku C, suprotno nekim drugim često korišćenim programskim jezicima, postoji samo jedan tip potprograma, a to je funkcija.
- Funkcija ima ime, tip rezultata koji vraća i listu argumenata, kao i tijelo funkcije oivičeno vitičastim zagrada:

```
tip_rezultata ime_funkcije(argumenti i tipovi argumenata)
{
    /*naredbe koje čine tijelo funkcije*/
}
```

Funkcija može biti bez argumenata, a ponekad se ni tip funkcije ne mora naglašavati

Funkcije

- Funkcija se može pozvati iz glavnog programa, kao i iz drugih funkcija, pa čak i iz same sebe.
- Ako funkcija, nakon obavljene operacije, vraća rezultat, to se postiže naredbom return.
- U slučaju da funkcija ne vraća rezultat deklariše se kao void (funkcija neodređenog tipa).
- Ako se ne navede tip rezultata funkcije podrazumjeva se da je funkcija tipa int, a ne void.

Primjer 3 funkcije

```
int zbir1(int x, int y)
{
        return x+y;
}
```

```
void zbir2(int x, int y)
{
        printf("%d\n",x+y);
}
```

```
int zbir3(int x, int y)
{
        printf("%d\n",x+y);
        return 1;
}
```

Bijelom bojom su označena zaglavlja funkcija gdje se, pored imena funkcije, navodi tip njenog rezultata, a u zagradi parametri funkcije. Vrijednosti koje se prosljeđuju funkciji u pozivu a=zbir1(3,b) se nazivaju argumentima funkcije.

Ove tri funkcije su slične, ali rade tri različite stvari:

- Prva funkcija vraća rezultat koji je suma argumenata;
- Druga funkcija ne vraća rezultat već samo štampa zbir;
- Treća funkcija štampa zbir i vraća rezultat 1 koji bi se mogao koristiti kao indikacija da je operacija obavljena uspješno.

Primjeri - Napomene

- Iz glavnog programa se funkcije tipa void pozivaju bez navođenja promjenljivih na lijevoj strani znaka jednakosti:
  zbir2(5,c);
- Rezultati funkcija koje nijesu void mogu da se pridruže dozvoljenoj lijevoj strani izraza, ali i ne moraju:
  c=zbir1(3,4);
  zbir3(c,d);
- U drugom slučaju funkcija će vratiti vrijednost, ali pošto nema lijeve strane izraza privremena promjenljiva u kojoj je sačuvana vrijednost rezultata će biti dealocirana.