## 14. NAREDBE BEZUSLOVNOG GRANANJA (BREAK, CONTINUE I GOTO)

Ponekad je potrebno imati mogućnost izlaska iz ciklusa/petlje, ali ne preko provere uslova, na vrhu ili na dnu petlje. Naredba **BREAK** izvršava "naprasni" izlazak iz bilo kog ciklusa/petlje ili iz naredbe višestrukog grananja – switch, za razdvajanje svakog pojedinačnog slučaja (case). Ova komanda obezbeđuje programu trenutno napuštanje ciklusa u kojoj se nalazi.

```
for( ... )
{
      ...
      if(uslov)
      break;
      ...
}
...
```

Primer:

```
for(i=1;i<n;i++)
{
      if(i>10)
      break; //prekid ciklusa kada je završeno 10 iteracija
}
```

Naredba **CONTINUE** je povezana s break, ali se koristi značajno manje. Njen zadatak je da počne sledeću iteraciju ciklusa/petlje u kojoj se program vrti bilo da se radi o for, while ili do - while ciklusa/petlje. Kod while i do - while ciklusa/petlje to znači da se trenutno prelazi na proveru uslova, a u for konstrukciji na sledeći stepen inkrementacije. Naredba continue koristi se samo u petljama, ali ne i kod switch naredbe (kao break). Naredba continue unutar switch naredbe prouzkovaće će prelazak u sledeću ciklus/petlju.

```
for( ... )
{
      ...
      if(uslov)
      continue;
      ...
}
...
```

Primer:

```
for(i=1;i<n;i++)
{
      if(i%10==0)
      continue; //preskakanje brojeva deljivih sa 10
}
```

Primer: Program koji računa proizvod n celih brojeva unetih sa tastature, ali preskače množenje ukoliko se unese 0 (nula).

```
//Proizvod brojeva od 1 do n, preskok za 0
#include <stdio.h>
int main ()
{
   int i,n,br,proizvod=1;
   printf("Unesite ceo prirodan broj do kog se racuna proizvod: ");
   scanf("%d", &n);
   for(i=1;i<=n;i++)
   {
        printf("Unesite broj%d: ", i);
        scanf("%d", &br);
        if(br==0)
```

```
            continue;
        proizvod*=br;
    }
    printf("Proizvod je: %d", proizvod);
    return 0;
}
```

Rezultat izvršavanja:

```
Unesite ceo prirodan broj do kog se racuna proizvod: 5
Unesite broj1: 2
Unesite broj2: 6
Unesite broj3: 0
Unesite broj4: 7
Unesite broj5: 3
Proizvod je: 252
```

C poseduje, ne baš preporučljivu i upotrebljivu, **GOTO** naredbu i **LABELE** na koje se grana. Ova naredba dovodi do loše strukturiranih programa, nečitljivih i teških za izmenu i održavanje. Formalno, GOTO naredba nikad nije nezaobilazna. Teorija kaže da sve što goto naredbom možemo postići može se i drugačije (i bolje) uraditi. Ovaj "izdanak" starih programskih jezika (asemblera, basic-a, fortran-a i sl.) ne preporučuje se u korišćenju, u današnjim verzijama programskih jezika, koje poseduju razne programske strukture kojima se može uraditi isti posao na daleko pregledniji I jednostavniji način. Ukoliko se, ipak, ova naredba koristi, preporuka je da skok bude isključivo na onom delu programa koji je okom vidljiv programeru na nekom ekranu. Labele treba da budu asocijativne i treba izbegavati način formiranja labela isključivo i samo sa brojevnim oznakama. Kako je to bilo jedino moguće, u već navedenim programskim jezicima, u prošlosti.

Ipak, ima nekoliko situacija koje čine GOTO ipak primenjivim. Najčešće se koristi u višeslojnim strukturama za prestanak odvijanja iteracija koje se "vrte", kakav je trenutni izlaz iz dve ili više petlji odjednom, s obzirom kada bi se koristila naredba break, ona bi delovala samo na unutrašnju petlju:

```
for( ... )
      for( ... )
      {
            ...
            if(uslov)
            goto greska;
            …
      }
...
greska:
prikaz_poruke;
```

Ovakav način rada je prihvatljiv ako program za rad s greškama nije baš jednostavan. Tada se sa napisanom GOTO naredbom, na mestima gde očekujemo greške, čeka da se ista, eventualno, desi. Sledi skok na liniju u kojoj se nalazi labela. Labele su istog oblika kao i imena promenljivih, ali se obeležavaju sa dve tačke ":". Labela može pripadati bilo kojoj naredbi.

# 15. JEDNODIMENZIONALNI NIZ ILI VEKTOR

**Niz** je izvedeni (struktuirani) tip podatka koji predstavlja strukturu (niz podataka) u kojoj su svi objekti (elementi) istog tipa. Tip niza proizilazi iz tipa iz tipa njegovih elemenata. Indeks niza je promenljiva ili izraz koji se nalazi između zagrada [ ]. Početni element niza ima indeks 0 (nula).

U programskom jeziku C svi nizovi, bez obzira na tip i veličinu, imaju sledeće zajedničke **karakteristike**:

- Pojedini članovi niza se nazivaju elementi niza i oni moraju biti istog tipa;
- Tip elementa niza je proizvoljne složenosti, tako da čak i izvedeni tipovi podataka predstavljaju legalne tipove za nizove,
- Elementima niza se pristupa preko indeksa niza. Vrednost indeksa prvog člana je 0, a ne 1.
- Naziv niza jednoznačno indentifikuje niz. On je konstantna vrednost (ne može se menjati operatorima dodele) koja predstavlja memorijsku adresu prvog člana niza.

Niz u celini ili njegovi elementi se mogu predavati funkcijama kao parametri poziva istih.

**Vektor** je naziv za jednodimenzionalni niz, koji se sastoji od više elemenata.

**Predstavljanje jednodimenzionalnog niza (vektora) u memoriji -** Niz se u memoriji implementira kao kontinualni skup susednih memorijskih lokacija**.** Niz se u memoriju smešta tehnikom **dopunjavanja**. Kod ove tehnike za smeštanje jednog elementa niza potrebno je nekoliko memorijskih reči. Tako se, pri smeštanju jednog elementa, preostali prostor u poslednjoj memorijskoj reči koju element zauzima do početka naredne reči ostavlja neiskorišćenim, pa sledeći element uvek počinje od sledeće memorijske reči. Ovim se postiže efikasan pristup elementima niza, što se plaća slabijoj iskorišćenju memorijskog prostora.

**Deklaracija vektora -** kao i sve druge promenljive u C programu, i jednodimenzionalni niz (tj. vektor) mora da se deklariše pre nego što se upotrebi. Tri podatka se formiraju prilikom deklaracije: ime/naziv, tip i veličina niza. Opšti oblik:

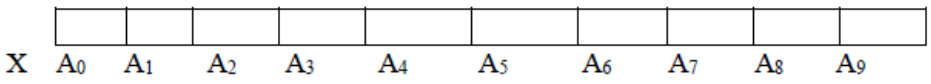
```
tip naziv[veličina];
```

Preko naziva niza se pristupa nizu, a veličina mora da bude konstanta ili konstantni izraz.

Primer:

```
int x[10];
```

U memoriji se za ovaj niz rezerviše prostor kao na slici:




U nazivu niza x upisana je adresa A0. Kako je veličina niza 10 to je u memoriji odvojeno toliko prostora. Adresa elementa niza sa indeksom **i** dobija se po formuli:

$$A_i = A_0 + i \cdot s$$

Ova tehnika adresiranja dobila je naziv **indeksno adresiranje**, jer se na osnovu indeksa dolazi do adrese elementa niza.

**Pristup elementu vektora -** Elementima niza se pristupa pomoću indeksa. Indeks elemenata niza predstavlja u stvari rastojanje između naziva i elementa niza, tako da prvi indeks niza ima vrednost 0 a ne 1 jer se nalazi na nultom rastojanju od naziva niza.

Primer:
Neka je niz y deklarisan kao niz znakova “program”:

```
char y[7];
y[0]="p";
y[1]="r";
y[2]="o";
y[3]="g";
y[4]="r";
y[5]="a";
y[6]="m";
```

Četvrti član niza koji dobija vrednost g. Peti član niza dobija vrednost r. U ovom slučaju brojevi 3 i 4 predstavljaju indeks niza. Redni broj člana niza ima vrednost za jedan veću od indeksa niza.

Indeks niza može biti i celobrojna promenljiva ili celobrojni izraz.

Postoji **vrsta niza koja se dobija nabrojavanjem elemenata** (enumerate). Nabrojavanje je formiranje liste konstantnih celobrojnih vrednosti, kao u primeru:

```
enum boolean{NO, YES};
```

Prvi naziv u enum listi ima vjednost 0, drugi 1. Da imamo još koji element, on bi imao vrednost 2, itd., dokle god se eksplicitno ne zada neka druga vrednost. Progresija ide od one vrednosti koja je poslednja eksplicitno zadana (inače od 0).

# 16. DEFINISANJE NIZA, INICIJALIZACIJA NIZA

**Niz** je izvedeni tip podatka koji predstavlja strukturu u kojoj su svi elementi istog tipa. Tip niza proizilazi iz tipa iz tipa njegovih elemenata. Indeks niza je promenljiva ili izraz koji se nalazi između zagrada [ ]. Početni element niza ima indeks 0 (nula). **Vektor** je jednodimenzionalni niz, koji se sastoji od više elemenata.

S obzirom da niz u sebi sadrži više elemenata, **inicijalizacija** podrazumeva da svi elementi nakon toga imaju neku početnu vrednost, a najčešće je to **0**. Za inicijalizaciju elemenata niza neophodno je koristiti ciklus, poput brojačkog ciklusa iI to se radi preko indeksa niza, tj. vektora. Brojač ciklusa se najčešće postavlja na 0, jer indeks niza polazi od nule.

**Definisanje i inicijalizacija vektora -** kao i sve druge promenljive u C programu, i jednodimenzionalni niz (tj. vektor) mora da se deklariše pre nego što se upotrebi. Opšti oblik naredbe za definisanje niza i inicijalizacije elemenata:

```
tip naziv[veličina];
for(indeks=0; indeks<=n, indeks++)
{
      naziv[indeks]=vrednost;
}
```

Primer definisanja i inicijalizacije vektora:

```
int x[10],i;
for(i=0; i<10; i++) x[i]=0;
```

**Definisanje i inicijalizacija dvodimenzionalnog niza -** dvodimenzionalni niz koji se još naziva i matrica je niz čiji su elementi jednodimenzionalni nizovi i deklariše se na sledeći način:

```
tip naziv[veličina1][veličina2];
```

Definicija matrice: Dvodimenzionalni niz ili matrica sastoji se od vrsta, pri čemu je svaka vrsta jedan vektor (jednodimenzionalni niz), tj matrica je niz vektora.

Primer definisanja i inicijalizacije dvodimenzionalnog niza:
Dvodimenzionalni niz se može predstaviti i kao tabela koja ima m vrsta i n kolona.
Matrica a[3][4], vrednosti se tabelarno mogu predstaviti na sledeći način:

| | Kolona 0 | Kolona 1 | Kolona 2 | Kolona 3 |
|---|---|---|---|---|
| Vrsta 0 | a[0][0] | a[0][1] | a[0][2] | a[0][3] |
| Vrsta 1 | a[1][0] | a[1][1] | a[1][2] | a[1][3] |
| Vrsta 2 | a[2][0] | a[2][1] | a[2][2] | a[2][3] |

Elementi dvodimenzionalnog niza identifikuju sa sa **a[i][j]**, gde su **i** i **j** pozitivni celi brojevi i predstavljaju indekse nizova. Dvodimenzionalni nizovi se mogu inicijalizovati na sličan način kao jednodimenzionalni, dodelom vrednosti svim elementima niza na sledeći način:

```
int a[3][4] = {
    {0, 1, 2, 3} ,    /*  inicijalizacija vrste sa indeksom 0 */
    {4, 5, 6, 7} ,    /*  inicijalizacija vrste sa indeksom 1 */
    {8, 9, 10, 11}   /*  inicijalizacija vrste sa indeksom 2 */
};
```

# 17. PRISTUPANJE ELEMENTIMA NIZA

**Nakon što je niz definisan, tj. deklarisan, preko naziva niza se pristupa nizu, a veličina niza (broj elemenata) mora da bude konstanta ili konstantni izraz.**

**Za pristup svim elementima niza** neophodno je koristiti ciklus. Koristi se brojački ciklus, a brojač ciklusa predstavljaće indeks niza (vektora) ili će figurisati u izrazu koji opisuje indeks vektora. Brojač ciklusa se najčešće postavlja na 0, jer indeks niza polazi od nule. Redni broj člana niza ima vrednost za 1 veću od indeksa niza.

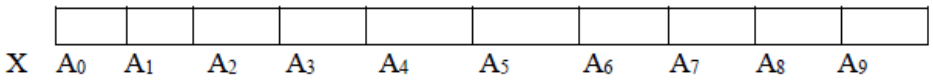
**Pristup jednom elementu niza, tj. vektora** se vrši pomoću indeksa. Indeks elemenata niza predstavlja rastojanje između naziva i elementa niza, tako da prvi indeks niza ima vrednost 0 a ne 1 jer se nalazi na nultom rastojanju od naziva niza. Indeks niza može biti celobrojna promenljiva ili celobrojni izraz koji se nalazi između velikih uglastih zagrada [ ]. Opšti oblik naredbe:

```
naziv_niza[indeks_elementa]
```

**Komponenta (element) niza** se označava imenom niza i **indeksom koji** jednoznačno određuje željeni element. Indeks je **izračunljivi** objekat - izraz, i to razlikuje niz od drugih strukturnih tipova.

**Primer:**
Neka je dat niz celih brojeva sa nazivom **x**, deklarisan kao **int x[10]**.

Prikaz niza **x** i njegovih elemenata:




Prvi element niza jeste npr. $A_0$, a poslednji $A_9$.

```
int x[10];
unsigned i;
i=3;
x[i]=5;
```

Sa x[3]=5 četvrti član niza dobija vrednost 4. U ovom slučaju broj 3 predstavlja indeks niza. U C-u se ne može se pisati x[11]=4 jer prilikom deklaracije niza nije rezervisano toliko memorijskog prostora (prekoračenje opsega). Ovo je fatalna greška.

**Dodela vrednosti elementima niza –** Primer niza celih brojeva sa imenom **a**:

```
int a[3];
a[0]=5;
a[1]=3;
a[2]=7;
int b[3];
b=a; /* Neispravno u C jeziku */
a++; /* Neispravno u C jeziku */
```

**U C-u se ne vrši provera da li je indeks u granicama**. Moguće je bez prijave greške ili upozorenja pristupati i lokaciji van opsega deklarisanog niza (na primer, element a[4] ili a[-1] u prethodnom primeru), ali će ovakav način rada sa nizovima najčešće dovesti do fatalnih grešaka prilikom izvršavanja programa.