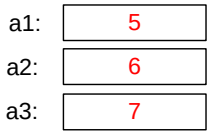
Zadaci – 11. dio, *Jednodimenzionalni statički niz*

Prisjetimo se: Sa naredbama …

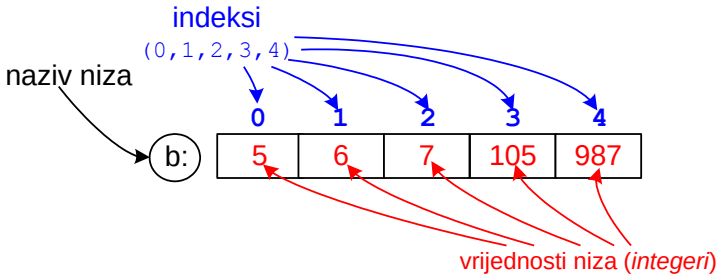
```
1:    int a1;
2:    int a2;
3:    int a3;
4:    a1 = 5;
5:    a2 = 6;
6:    a3 = 7;
```

…deklarišemo i incijalizujemo tri integera **a1**, **a2** i **a3** tako da svaki od njih ,,pamti" neku vrijednost. Ta vrijednost mora biti cijeli broj – integer. Svaka varijabla može pamtiti samo jednu vrijednost. Pogledajte sliku br. 1.



Slika 1

Sa naredbom **`int b[5];`** deklarišemo niz integera dužine **5**. To znači, da sada **b** može pamtiti više od jedne vrijednosti, jer se radi o *nizu*. Niz **b** može pamtiti **5** vrijednosti (integera). Prva vrijednost se smješta na indeks poziciju **0**, druga vrijednost na indeks poziciju **1**,…, a peta vrijednost na indeks poziciju **4**. Pogledajte sliku br. 2.



Slika 2

Kako smjestiti neku vrijednost u niz na željenu indeks poziciju?

To možete vršiti na sličan način kao i sa običnim varijablama, tako što nakon naziva niza (**b**) slijede srednje zagrade '**[]**' a u njima vrijednost indeks pozicije. Pa ako želimo popuniti vrijednosti u niz kao u prethodnoj slici to bi činili na slijedeći način (redoslijed nije bitan):

```
1:    b[0] = 5;
2:    b[1] = 6;
3:    b[3] = 105;
4:    b[2] = 7;
5:    b[4] = 987;
```

Primjer ne dozvoljenog pristupa:

```
1:    b[5] = 988;  // Greška, indeks broj 5 u nizu b (čija je dužina 5) ne postoji!)
```

Na slici br. 2 možete vidjeti da ne postoji elemenat sa indeksom 5.

Kako vidimo **b[0], b[1], …, b[4]** se mogu koristiti kao obične varijable (npr. **a1, a2, a3**). Sa elementima niza možemo vršiti...

- aritmetičke operacije, in/dekrement (post i pre), …

```
b[0] = 2 * b[1] + b[2];  // isto kao b[0] = 2 * 6 + 7;
b[0]++;
--b[1];
```

- logičke operacije

```
if (b[0] > b[2])
```

- cout i cin

```
cin >> b[0];
cout << b[0];
```

Da zaključimo:

Do sada smo koristili pojedinačne varijable tipa *int*, *char*, ili druge objekte. Često imamo potrebu da deklarišemo kolekciju objekata kao npr. 20 cijelih brojeva (*int*) ili drugu vrstu objekata.

Šta je niz?

Niz je skup memorijski lokacija za smještanje podataka, a svaka od njih sadrži isti tip podataka. Svaka lokacija naziva se ***element niza***.

Niz deklarišemo tipom podatka, zatim imenom niza i brojem elemenata niza u uglatim zagradama. Na primjer, naredba

```
int imeNiza[25];
```

deklariše niz od 25 cjelobrojnih vrijednosti pod nazivom `imeNiza`. Kada kompajler naiđe na ovu deklaraciju, on rezerviše dovoljno memorijskih lokacija za svih 25 elemenata. Znamo da svaki *integer* zahtijeva 4 bajta (u Visual C++), ova deklaracija rezerviše 100 uzastopnih bajtova memorije.

Elementi niza broje se od `0`. Zato, prvi element niza je `imeNiza[0]`. U `imeNiza` primjeru, `imeNiz[0]` je prvo element, `imeNiza[1]` drugi, itd.
Treba paziti! Niz `nekiNiz[3]` ima tri elementa: `nekiNiz[0]`, `nekiNiz[1]`, i `nekiNiz[2]`. Općenito, `nekiNiz[n]` ima `n` elemenata koji se broje od `NekiNiz[0]` do `OvajNiz[n-1]`. Zato, `imeNiza[25]` se broji od `imeNiza[0]` do `imeNiza[24]`.

Zadatak 122:

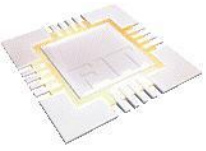
Napravite program u kome ćete deklarisati niz od pet cijelih brojeva i u taj niz smjestite vrijednosti koje se trebaju unijeti sa tastature. Nakon unosa svih brojeva program treba ispisati sve vrijednosti niza na ekran. Na primjer:

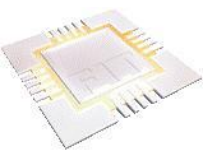
```
Vrijednost za OvajNiz[0]: 6
Vrijednost za OvajNiz[1]: -5
Vrijednost za OvajNiz[2]: 55
Vrijednost za OvajNiz[3]: 33
Vrijednost za OvajNiz[4]: -114
========
0: 6
1: -5
2: 55
3: 33
4: -114
```

Rješenje se nalazi na stranici 173.

Pomoć:

- Koristite jednu *for*-petlju za učitavanje brojeva. Početna vrijednost brojača neka bude vrijednost prvog elementa, a krajnja vrijednosti brojača neka bude indeks zadnjeg elementa.
- Koristite jednu *for*-petlju za ispis vrijednost
- Iz iskustva preporučujemo da za krajini uslov prekida petlji ne koristite logički operator '**<**' nego da koristite '**<=**', npr.:
    - umjesto `i<10` koristite `i<=9`





Kako se računa memorijska lokacija nekog elementa niza?

Kada dodjeljujemo vrijednost nekom elementu niza kompajler računa, na bazi veličine (u bajtama) podatka jednog elementa i indeks pozicije elemenata u nizu, gdje pohraniti tu vrijednost. Pretpostavimo da upisujemo vrijednost u `intNiz[5]`, šesti element. Kompajler množi indeks poziciju (5) sa veličinom jednog elementa. U ovom slučaju je 4 bajta (jer veličina jednog *integera* u kompajleru *Visual C++* iznosi 4 bajta). Onda se pomjera toliko bajta (20) od početka niza i upisuje tu novu vrijednost.

Ako hoćemo upisati `intNiz[50]`, kompajler ignoriše činjenicu da taj element ne postoji i računa koliko je daleko od prvog elementa (*200 bajta*) i upisuje tu vrijednost preko bilo kojeg podatka koji se u toj nepoznatoj lokaciji nalazi na kojoj može biti bilo kakav podatak, i to može dovesti do nepredvidivih rezultata.

U ovom slučaju kompajler izgleda kao slijepac koji broji odstojanje od kuće u koracima. Počinje od prve kuće, `GlavnaUlica[0]`. Ako treba da ode do šeste kuće on računa da mora proći još pet kuća po deset koraka 50 koraka. Ako mu kažete da ide do `GlavnaUlica[100]`, a `GlavnaUlica` ima samo 25 kuća, on će napraviti 1000 koraka. Tako ga možete poslati u pogrešnu ulicu, odnosno u tuđi memorijski prostor. U najboljem slučaju operativni sistem nam neće dozvoliti pristup, a malo gori slučaj bi bio da padne aplikacija (naša ili neka druga) ili čak operativni sistem, a moguća je izmjena nekih podataka koju nećemo možda nikada primijetiti.

Inicijalizaciju članova niza možete vršiti poslije deklaracije kao što je učinjeno u primjeru sa prve stranice ovog poglavlja, a možete i pri deklaraciji niza npr.:

```
int b[5] = {5, 6, 7, 105, 987};  // deklaracija i inicijalizacija niza
```

Naravno,moguće je inicijalizovati manje članova nego što se deklariše, a vrijednost neinicijalizovanim članovima niza možete kasnije dodijeliti:

```
1:   int b[5] = {5,6,7}; // deklaracija niza sa 5 članova i inicijalizacija članova sa indeks-pozicijom 0, 1 i 2
2:   b[3] = 105;     // izmjena vrijednosti članu niza sa indeksom 3
3:   cin >> b[4];    // izmjena vrijednosti članu niza sa indeksom 4
4:   cin >> b[0];   // izmjena vrijednosti članu niza sa indeksom 0
```

Da zaključimo, srednje zagrade '**[]**' koristimo kod deklaracije niza linija br. 1 i kod pristupa nekom članu niza (linija br. 2 do 4).

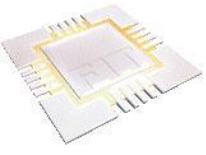
Pri deklaraciji niza vršite i inicijalizaciju onda ne morate navoditi dužinu niza, jer će kompajler kreirati niz onolike dužine koliko ste inicijalizovali članova niza:

```
int b[] = {5, 6, 7, 105, 987}; // deklaracija niza dužine 5 i inicijalizacija članova
```

Pri pristupu nekom članu niza možete umjesto brojeva koristiti i varijable ili neki aritmetički izrazi, ali pri deklaraciji niza ne smije biti varijabla:

```
int x = 3;
b[x] = 105;   // ovo je isto kao b[3] = 105;
b[x-1] = 7;   // ovo je isto kao b[2] = 7;
```

Korištenje varijabli umjesto brojeva pri pristupu članovima niza (indeks niza), čini nizove



veoma moćnim. Tako možemo sa nekoliko linija koda dodijeliti neku vrijednost svim članovima niza:

```
1:   int b[1000];         // deklaracija niza dužine 1000
2:
3:   for (int i = 0; i <= 999; i++)
4:   {                     // prvi  član dobija vrijednost 0, jer je i = 0
5:       b[i] = 2*i;      // drugi član dobija vrijednost 2, jer je i = 1
6:   }
```

Kao indeks niza možemo naravno koristi varijable tipa *integer*, neke složenije izraze ili čak druge elemente niza (varijable):

```
1:   int niz[10];                // deklaracija niza sa 10 elemenata
2:
3:   int i = 0;
4:   niz[i] = 4;                 // prvi element: niz[0]=4
5:
6:   i++;                        // i=i+1, tj. sad je i=1
7:
8:   niz[i] = 59                 // drugi element: niz[1]=59
9:
10:  niz[i+1] = 5                // treci element: niz[2]=5
11:
12:  niz[niz[2]] = 63            // isto kao niz[5]=63, tj.  šesti elemenat=63
```

Pazite: Pri deklaraciji niza, morate koristiti brojeve (kao što smo već vidjeli) ili ***const***-varijable:

```
1:   const int i = 10; // konstanta varijabla tipa integer
2:
3:   int niz[i];       // deklaracija niza koristeći konstantnu varijablu kao dužinu nizu
4:
```

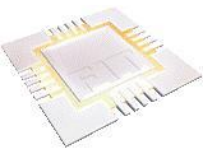
To znači da nije moguće navoditi kao dužinu nizu neku običnu varijablu. Ako pokušate:

```
1:   int i;
2:   cout << "Unesi dužinu niza: ";
3:   cin >> i;
4:
5:   int niz[i];  // u ovoj liniji koda kompajler će prijaviti grešku:
6:                       // expected constant expression
7:                       // cannot allocate an array of constant size 0
                          // 'niz' : unknown size
```

Razlog je u tome što se memorija za niz rezerviše u *compile-time*, a ne u *run-time* aplikacije.

Zadatak 123:
Napravite program koji će od korisnika zahtijevati da unese broj elemenata (`x`) koji nije veći od 20, a zatim da unese te elemente, jedan za drugim. Zatim, program treba sve elemente ispisati obrnutim redoslijedom.

Rješenje se nalazi na kraju dokumenta.



Pomoć:

- Prilikom određivanja veličine niza ne smijete koristiti obične varijable. Smijete koristiti brojeve ili konstante varijable, ali njihova vrijednost mora dodijeljena u trenutku pisanja kôda, a ne da ih upisuje korisnik u toku izvršavanja programa u toku.
- Odredit ćemo dužinu niza dovoljno veliku za smještanje svih brojeva: Neka niz bude veličine 20.
- Pazite na indekse (pozicije) elemenata:
  - za učitavanje koristite uzlaznu *for*-petlju: `0` do `x-1`
  - za ispis koristite silaznu *for*-petlju: `x-1` do `0`

Zadatak 124:

Evo dobre prilike za isprobate konstante varijable:

Prepravite program, tako da koristite broj 20 samo jednom u kôdu umjesto tri puta.

To možete postići korištenjem konstanti (podatak tipa `const`). Na taj način, ako bude bilo neophodno da vaš program može raditi i sa većim nizom, onda morate samo jedan broj prepraviti (npr. broj 20 zamijenite sa brojem 30) koji se nalazi samo jednom na početku programa, inače, bez konstante, bilo bi neophodno na tri mjesta prepravljati broj 20 sa 30.

Dodajte i računanje kvadrata unesenih brojeva. Program formatirajte na sljedeći način:

```
Unesite duzinu niza: 5
1: 6
2: -5
3: 55
4: 33
5: -114
========

5: -114 * -114  = 12996
4: 33 * 33  = 1089
3: 55 * 55  = 3025
2: -5 * -5  = 25
1: 6 * 6  = 36
```

Rješenje se nalazi na kraju dokumenta.

Zadatak 125:

Napravite program u kome ćete deklarisati niz veličine 50 tipa *double*. Pitajte korisnika koliko brojeva želi unijeti. U slučaju da odgovor bude veći od 50, pitajte ponovo sve dok ne unese broj koji nije veći od 50.
Zatim od korisnika zahtijevate unos toliko brojeva. Program treba nakon što je korisnik unio sve brojeve ispisati uzastopno vrijednosti korijena tih brojeva. Vrijednost korijena negativnog broja ispišite kao u obliku imaginarnih brojeva na matematički ispravan način (ovo možete riješiti pomoću jednog *if-else*-iskaza). Program formatirajte na sljedeći način:

```
Unesite duzinu niza: 5
1: 6
2: 2.78
3: -9
4: -100.000
5: 81
========
2.44949
1.66733
3i
10i
9
```

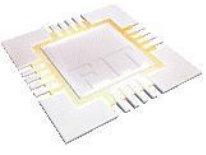
Rješenje se nalazi na kraju dokumenta.

Pomoć:

- Koristite konstantu varijablu za veličinu niza.

- Ispis niza vršite u jednoj uzlaznoj *for*-petlji, koja mijenja broj `n` od `0` do `x-1`.

- u *for*-petlji:
  - ako je (`n+1`)-ti element (element sa indeks pozicijom `n`) pozitivan broj ispišite vrijednost korijena od tog broja, tj. korijen broja `(niz[n])`
  - ako je element sa indeksom `n` negativan broj ispišite vrijednost korijena od



negativne vrijednost tog elementa, tj. korijen iz broja `(-niz[n])`, zatim dodajte (ispišite) matematičku oznaku za imaginarni broj (tj. slovo '`i`')

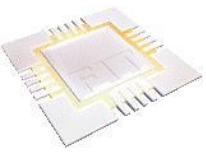
Rješenja

Rješenje zadatka br. **122**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void main()
5:   {
6:          int OvajNiz[5];
7:
8:          //ucitavanje niza
9:          for (int i=0; i<=4; i++)
10:         {
11:                cout << "Vrijednost za OvajNiz[" << i << "]: ";
12:                cin >> OvajNiz[i];
13:         }
14:
15:         cout << "========\n";
16:
17:         //ispis niza
18:         for (int n=0; n<=4; n++)
19:                cout << n << ": " << OvajNiz[n] << endl;
20:  }
```

Rješenje zadatka br. **123**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void main()
5:   {
6:          int x;
7:          int niz[20];
8:
9:          cout << "Unesite duzinu niza: ";
10:         cin >> x;
11:
12:         if (x>20)  // ako je slučajno x>20
13:                x=20;     // neka onda x bude 20
14:
15:         for (int i=0; i<=x-1; i++)
16:         {
17:                cout << i+1 <<": ";  // (i+1) na pokazuje redni broj unosa
18:                cin >> niz[i];
19:         }
20:
21:         cout << "========\n";
22:
23:         for (i=x-1; i>=0; i--)
24:         {
25:                cout << niz[i] << endl;
26:         }
27:  }
```



Rješenje zadatka br. **124**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void main()
5:   {
6:          int x;
7:          int niz[20];
8:
9:          cout << "Unesite duzinu niza: ";
10:         cin >> x;
11:
12:         if (x>20)  // ako je sluèajno x>20
13:                x=20;     // neka onda x bude 20
14:
15:         for (int i=0; i<=x-1; i++)
16:         {
17:                cout << i+1 <<": ";  // (i+1) na pokazuje redni broj unosa
18:                cin >> niz[i];
19:         }
20:
21:         cout << "========\n\n";
22:
23:         for (int i=x-1; i>=0; i--) // u nekim kompajlerima ne smjijete ponovo deklarisati varijablu i
24:         {
25:                cout << i+1 << ": " << niz[i] << " * " << niz[i] << "  = "
                                                     << niz[i] * niz[i] << endl;
26:         }
27:  }
```

Rješenje zadatka br. **125**:

```
1:   #include <iostream>
2:   #include <math.h>
3:   using namespace std;
4:
5:   void main()
6:   {
7:          const int d = 50;   // d maksimalna dozvoljena dužina niza
8:
9:          int x;
10:         double niz[d];            // niz tipa double dužine n
11:
12:         do
13:         {
14:                cout << "Unesite duzinu niza: ";
15:                cin >> x;
16:         }while(x >= 51);  // ponovi unos ako
17:
18:
19:         for (int i=0; i<=x-1; i++)
20:         {
21:                cout << i+1<<": ";
22:                cin >> niz[i];
23:         }
24:         cout << "========\n";
25:
26:         for (int n=0; n<=x-1; n++)
27:         {
28:                if (niz[n] >= 0)
29:                       cout << sqrt(niz[n]) << endl;
30:                else
31:                       cout << sqrt(-niz[n]) << "i" << endl;
32:         }
33:  }
```