

c++ tutorial sa primjerima

[es] :: C++ programiranje :: c++ tutorial sa primjerima

Autor

peromalosutra

Puno ime: Ivan Rajkovic
 Član broj: 54774
 Poruke: 216
 Lokacija: Banjaluka
 *.dialup.blic.net.
 OS: Windows XP

[Profil](#) [Email](#) [Privatna Poruka](#)

Pretraga teme:

[Markiranje](#)

[Štampanje](#)

[Email praćenje](#)

[Traži](#)

[RSS](#)

Re: c++ tutorial sa primjerima

10.02.2006. u 17:58

Varijable: imena, tipovi, deklaracija i dodjela vrijednosti

Varijabla ili promjenjiva je zapravo lokacija u memoriji koja sadrži određeni podatak. Svaka varijabla mora imati ime. Tokom imenovanja varijable moraju se poštovati neka pravila, npr. ime varijable ne smije početi brojem i ne smije sadržavati razmake ili specijalne znake osim donje crte (_). Treba voditi i računa da c++ razlikuje velika i mala slova pa su varijable broj, Broj i BROJ tri različite varijable.. Budući da postoje različite vrste podataka, postoje i različite vrste varijabli. Za početak ću pomenuti samo 3 tipa: int, char i float.

int sadrži cijele brojeve u intervalu od -maxint do +maxint, gdje je maxint najčešće 32768.

char može da sačuva bilo koje slovo, broj ili karakter dužine 1 znaka (8 bajta).

float služi za rad sa decimalnim brojevima.

Izbor varijable zavisi uglavnom od vrijednosti koju će ona sadržavati i mora se definisati prije pokretanja samog programa. Varijable se zato moraju "deklarirati", odnosno svakom imenu varijable mora se odrediti tip. Na primjer sa int broj deklariramo promjenjivu po imenu broj, odnosno u memoriji se otvara nova lokacija pod imenom "broj" koja može da sadrži cijele brojeve tipa integer. Sada možemo da u ovu varijablu spremimo na primjer broj 5 tako što ćemo napisati broj=5. Ako ovoj varijabli međutim pokušamo da dodjelimo vrijednost drugog tipa, npr broj='a' kompajler će prijaviti grešku. Tokom dodjeljivanja vrijednosti varijabli tipa char moramo da znak koji se varijabli predaje stavimo pod apostrofe, npr. char znak='d'. Jednoj varijabli možemo da dodjelimo vrijednost druge varijable, ali i tada naravno moramo da pazimo da te dve varijable budu istog tipa (tokom rada na zadacima koji slijede pokazaćemo da ovo nije uvijek tačno). Evo primjera nekoliko ispravnih deklaracija i dodjeljivanja vr. varijablama:

Code:

```
int i,j=3,n;
float f;
char c;
i=4;
f=3.14;
c='a';
```

Aritmetički operatori

Spomenućemo sabiranje (+), oduzimanje (-), množenje (*) i djeljenje (/). Njihova upotreba je jednaka kao u matematici, a isti je i redoslijed operacija (množenje i djeljenje imaju prednost nad sabiranjem i oduzimanjem). Pri djeljenju integera drugim integerom

ostatak se zanemaruje, a operator koji nam pokazuje ostatak pri djeljenju dva broja je %. Tako je $21/5=4$, a $21\&5=1$ ukoliko radimo sa integerima. Ukoliko moramo da znamo i decimalne vrijednosti pri djeljenju tada koristimo tip float.

Unos i ispis na ekran

Objasniću upotrebu samo po 1 po 1 naredbe za unos i ispis koju smatram najpogodnijom za početnike: cout i cin.

cout (Console OUTput) služi za ispis teksta, brojeva, sadržaja varijabli i sličnog na ekran. Sintaksa je cout << var; Gdje je var ime neke varijable, a isto tako to može biti i tekst (tada ga ograničavamo navodnicima). Naravno možemo da "šampamo" na ekran istovremeno i varijable i tekst... Da ne bih prekršio neki zakon ili napravio jeres, evo programa "Zdravo svijete" (primjetite kako smiješno zvuči) za početak.

Code:

```
#include <iostream.h>
main()
{
cout << "Zdravo svijete!";
return 0;
}
```

cin (Console INput) služi za prihvatanje podataka od korisnika. Po nailasku na ovu naredbu program privremeno zaustavlja rad i očekuje od korisnika da unese neku povratnu informaciju. Nakon unosa i pritiska na ENTER, program prvo provjerava da li unos sa tastature odgovara tipu varijable, pa ako je sve regularno unešenu vrijednost dodjeljuje navedeoj varijabli. Sintaksa je cin >> var; gdje je var neka varijabla koja je već definisana. Obratite pažnju na "overloading operator", tj dve strelice okrenute na desno. O ovome ću govoriti kasnije, ali za sada samo uočite da su okrenute na suprotnu stranu nego kod cout naredbe. Primjer:

Code:

```
#include <iostream.h>
main()
{
int n;
cin << n;
return 0;
}
```

Budući da programi koji koriste samo input ili output komande nisu od nikakve koristi (kao što vidite na prethodnom primjeru), najčešće se koriste kombinacije:

Code:

```
#include <iostream.h>
main()
{
cout << "OVAJ PROGRAM SABIRA 2 BROJA" << endl;
int a,b;
cout << endl << "a=";
cin >> a;
cout << endl << "b=";
```

```
cin >> b;
cout << endl << "ZBIR JE " << a+b;
return 0;
}
```

If uslov

Provjerava istinitost nekog izraza. Sintaksa je sledeća:

if (uslov) {naredba ako je uslov tačan}; ili

if (uslov) {naredba ako je uslov tačan} else {naredba ako je uslov netačan};

uslov može biti bilo koji izraz čija se tačnost može provjeriti, najčešće međutim se provjerava odnos između dva broja. U tu svrhu koriste se relacioni operatori:

== (jednako), != (različito), < (manje), <= (manje ili jednako), > (veće), >= (veće ili jednako)... primjer:

Code:

```
#include <iostream.h>
main()
{
    int a,b;
    cout << "a,b=";
    cin >> a,b;
    if (a==b) {cout <<"brojevi su jednaki";}
    else {cout <<"brojevi su različiti";}
    return 0;
}
```

For petlja

Služi za ponavljanje naredbe ili grupe naredbi ukoliko je unaprijed poznat broj ponavljanja. Sintaksa glasi: for (početna vrijednost; uslov; stepen rasta) {naredba}.

For petljom "upravlja" kontrolna varijabla tipa integer (int) koja mijenja vrijednost za određeni stepen kroz svaki prolaz petlje. Prvi korak je inicijacija gdje kontrolnoj varijabli pridružimo neku početnu vrijednost. Zatim navodimo uslov; petlja će se vrtiti dok kod je on tačan. Stepenom rasta označavamo za koji interval će se kontrolna varijabla uvećavati kroz svaki ciklus petlje. for (int i=1; i<10; i++) znači da će se varijabla i, počevši od 1 uvećavati za jedan dok god je i manje od 10, dakle sve naredbe u sklopu petlje će se ponoviti 10 puta. Evo primjera koji će sabrati i pomnožiti brojeve do n.

Code:

```
#include <iostream.h>
main()
{
    int n,s=0,p=1;
    cout << "n=";
    cin >> n;
    for (int i=1; i<=n; i++)
    {
        s+=i;
        p*=i;
    }
    cout << "s=" << s;
    cout << "p=" << p;
}
```

```
return 0;
}
```

While petlja

Ukoliko nije unaprijed poznat broj ponavljanja onda koristimo while petlju. while (a>b) { naredba ili grupa naredbi } znači "ponavlja naredbu u sklopu petlje dok kod je uslov tačan". Poenta je da kod ove naredbe ne mora biti unaprijed poznat broj ponavljanja, te se naredbe ponavljaju sve dok se istinitost uslova ne promjeni. Tu međutim moramo da budemo pažljivi, ako uslov nije dobro formulisan i ne postoji mogućnost da se on u toku rada petlje izmjeni program će ili preskočiti petlju ili beskonačno dugo ostati u njoj (do nasilnog prekidanja rada samog programa). Ovo zna da bude iritanto početnicima jer tako često gube cijeli program, te moraju da ga pišu iz početka (zna biti nezgodno na ispitu)... Evo primjera jednog programa koji sabira brojeve sve dok je zbir manji od 5000;

Code:

```
#include <iostream.h>
main()
{
    int i=0,s=0;
    while (s<5000)
    {
        i++;
        s+=i;
    }
    cout << "s=" << s;
    return 0;
}
```

Ovdje se vidi bitna razlika između while i for petlje. Ovaj program bi bilo (gotovo) nemoguće napisati sa for petljom jer ne znamo unaprijed koliko brojeva ćemo morati sabirati, nego ih sabiramo dok zbir ne dostigne određenu vrijednost.

Jednodimenzionalni nizovi

Nizove korisimo kada imamo više varijabli jednog tipa koje su na neki način povezane. Niz je skup varijabli istog tipa kojima pristupamo prema njihovom indeksu. int niz[10] će u memoriji rezervirati 10 memorijskih lokacija za varijable tipa int. Ukoliko želimo da se obratimo nekom članu niza jednostavno navedemo ime niza i zatim indeks (rednim broj) člana u tom nizu. Na primjer niz[0]=26 bi prvom članu niza dalo vrijednost 26 (prvi član niza ima indeks 0). Pri upisivanju članova niza moramo da pazimo da ne pređemo maksimalni broj elemenata niza koji smo pri definisanju naveli. Tako kompajler ne bi prijavio grešku kad bi u naš prethodni niz pokušali da upišemo niz[135]=15, već bi on u lokaciju koja se nalazi 135 "varijabli" daleko od početka niza upisao broj 15, zamijenjujući na taj način sve što se tu prije nalazilo. Ovo je opasno i nepredvidivo i često je uzrok čudnih grešaka ili pada programa. A sada evo zadatka koji će u niz upisati prvih 10 parnih brojeva:

Code:

```
#include <iostream.h>

main()
```

```

{
    int nizParnihBrojeva[10];
    for (int i=0; i<10; i++)
    {
        nizParnihBrojeva[i]=i*2;
        cout << "NIZ
[" << i << "]=" << nizParnihBrojevai[i];
    }
    return 0;
}

```

Dvodimenzionalni nizovi (matrice)

Matricu možemo da zamislamo kao tabelu; sastoji se od elemenata koji pripadaju redovima i kolonama koje su značeni indeksima. int matrica[10][15] bi rezervisalo prostor za $10 \cdot 15 = 150$ varijabli tipa int. Uzmimo sada na primjer matricu mat[x][y]. Ako je $x=y$, matrica je kvadratna. U takvoj matrici svi elementi za čije indekse važi $x=y$ nalaze se na glavnoj dijagonali. Za elemente za čije indekse važi da su $x+y=x+1$, odnosno $x+y=y+1$ (iz uslova $x=y$) slijedi da se nalaze se na sporednoj dijagonali. Evo jednog zadatka čije je rješenje nedavno traženo na forumu: Napisati program koji će sa tastature prvo učitati prirodni broj n koji mora biti između 2 i 10. Nakon toga učitati nxn realnih brojeva u matricu. Pretpostaviti da u matricu može stati najviše 10x10 brojeva. Pronaći i na ekran ispisati sumu svakog reda matrice i proizvod svake kolone matrice. Rješenje:

Code:

```

#include <iostream.h>
#define max 10

main ()
{
    int matrica[max][max], suma[max], proizvod[max];
    int n,s=0,p=1;
    cout << "n=";
    cin >> n;
    for (int i=1; i<=max; i++)
    {
        suma[i]=0;
        proizvod[i]=1;
    }
    for (int i=1; i<=n; i++)
    for (int j=1; j<=n; j++)
    {
        cout <<"Mat ["<<i<<","<<j<<"]=";
        cin >> matrica[i][j];
        suma[j]+=matrica[i][j];
        proizvod[i]*=matrica[i][j];
    }
    for (int i=1; i<=n; i++)
    cout<<"Suma ["<<i<<"]="<<suma[i]<<endl;
    for (int i=1; i<=n; i++)
    cout<<"Proizvod ["<<i<<"]="<<proizvod[i]<<endl;
    return 0;
}

```

peromalosutra

Puno ime: Ivan Rajkovic
 Član broj: 54774
 Poruke: 216
 Lokacija: Banjaluka
 *.dialup.blic.net.
 OS: Windows XP

[Profil](#) [Email](#) [Privatna Poruka](#)

Re: c++ tutorial sa primjerima

10.02.2006. u 19:30

Dva zadatka kao klasičan primjer za izdvajanje pojedinih cifara iz integera:

1) Napisati C program koji će pronaći i ispisati na ekran sve trocifrene brojeve kod kojih je suma cifri djeljiva sa 7.

Code:

```
#include <iostream>
using namespace std;

main()
{
    int s,d,j;    //stotine, desetice, jedinice...
    for (int i=100; i<1000; i++)    //
    petlja vrti sve 3-cifrene brojeve
    {
        int temp=i;
        j=temp%10;    //
        izdvajamo redom cifre broja tako što
        temp/=10;    //
        prvo tražimo ostatak pri djeljenju
        d=temp%10;    //
        sa 10, a zatim broj i djelimo sa 10
        temp/=10;    //
        da se oslobodimo zadnje cifre i
        s=temp%10;    //
        omogućimo traženje nove
        int sum=s+d+j;
        if (sum%7==0)    //
        provjeravamo da li je suma djeljiva sa 7
        cout << i << endl;    //
        ako jeste, ispisujemo broj
    }
    system ("pause");
    return 0;
}
```

2)

Napisati program koji će pronaći i ispisati na ekran koliko ima peterocifrenih brojeva kod kojih je suma zadnje 4 cifre jednaka prvoj cifri.

Code:

```
#include <iostream>
using namespace std;

main()
{
    long int i;
    int cf[6],br=0;
    for (i=10000; i<100000; i++)    //
    petlja koja vrti sve 5-cifrene br.
    {
        long int temp=i;
        for (int j=5; j>0; j--)    //
        petlja u kojoj izdvajamo cifre broja
```

```

        {
            cf[j]=temp%10;           //
        uzimamo zadnju cifru broja i spremamo je u niz
            temp/=10;               //
        "odsjećamo" zadnju cifru da bi mogli naći sledeću
        }
        int sum=cf[2]+cf[3]+cf[4]+cf[5];           //
        suma zadnje 4 cifre
        if (cf[1]==sum)                             //
        ako je suma jednaka prvoj cifri
        {                                           //
            ispisuje se taj broj                    //
            cout << i << endl;                       //
        a brojač se uvećava za 1
            br++;
        }
    }
    cout << "br=" << br << endl;                   //
    ispis brojača
    system ("pause");
    return 0;
}

```

10.02.2006. u 19:30

[Odgovor na temu](#)**peromalosutra**

Puno ime: Ivan Rajkovic
 Član broj: 54774
 Poruke: 216
 Lokacija: Banjaluka
 *.dialup.blic.net.
 OS: Windows XP

[Profil](#) [Email](#) [Privatna Poruka](#)
 **Re: c++ tutorial sa primjerima**

10.02.2006. u 23:02

A evo i 2-3 primjera za nizove i matrice...

1)
 Napisati parne članove niza veće od aritmetičke sredine. (nije potrebno objašnjenje)

Code:

```

#include <iostream>
#define max 10
using namespace std;

main()
{
    int n, niz[max], s=0;
    float as;
    cout << "n=";
    cin >> n;
    for (int i=0; i<n; i++)
    {
        cout << "NIZ[" << i << "]=";
        cin >> niz[i];
        s+=i;
    }
    as=s/n;
    for (int i=0; i<n; i++)
        if (niz[i]>as && niz[i]%2==0)
            cout << "NIZ[" << i << "]=" <<niz[i]
    << " zadovoljava uslov. \n";

    system ("pause");
}

```

```

        return 0;
    }

```

2)

Napisati C program koji će s tipkovnice prvo učitati prirodni broj m koji mora biti između 2 i 10. Nakon toga učitati $m \times m$ cijelih brojeva u matricu A . Odrediti najveći broj u matrici te načiniti matricu B istih dimenzija koja će za elemente matrice sadržavati produkt najvećeg elementa i svih ostalih elemenata matrice A .

Code:

```

#include <iostream>
#define dim 10
using namespace std;

main()
{
    int n,a[dim][dim],b[dim][dim],max=0;
    cout << "n=";
    cin >> n;
    for (int i=1; i<=n; i++)
        for (j=1;j<=n; j++)
        {
            cout <<"a["<<i<<" , "<<j<<"]=";
            cin >>a[i][j];
            if (a[i][j]>max)
                max=a[i][j];
        }
    for (int i=1; i<=n; i++)
        for (j=1;j<=n; j++)
            b[i][j]=max*a[i][j];
    system ("pause");
    return 0;
}

```

3)

Napisati C program koji će s tipkovnice prvo učitati cijeli broj m . Taj broj m mora biti pozitivan i ne veći od veličine matrice A (za matricu predvidjeti najviše 10×10 elemenata). Program zatim treba učitati s tipkovnice realne brojeve u matricu A veličine $m \times m$. Nakon toga s tipkovnice unijeti m cijelih brojeva u vektor V koji je veličine 10 elemenata. Program zatim treba brojeve iz vektora V dodati elementima matrice A koji su na glavnoj dijagonali. Tako dobivenu matricu ispisati na ekran.

Code:

```

#include <iostream>
#include <vector>
#define max 10
using namespace std;

main()
{
    int n;
    float a[max][max];

```

```

vector <float> v;
cout << "n=";
cin >> n;
if (!(n>0 && n<max))
    exit(0);
for(int i=1; i<=n; i++)
    for (int j=1; j<=n; j++)
    {
        cout << "a["<<i<<","<<j<<"]=";
        cin >> a[i][j];
    }
for (int i=1; i<=n; i++)
    {
        float temp;
        cout << "vektor["<<i<<"]=";
        cin >> temp;
        v.push_back(temp);
    }
for(int i=1; i<=n; i++)
    for (int j=1; j<=n; j++)
    {
        if (i==j)
            a[i][j]+=v[i];
    }
for(int i=1; i<=n; i++)
    {
        cout << endl;
        for (int j=1; j<=n; j++)
            cout <<a[i][j]<<" ";
    }
system ("pause");
return 0;
}

```

10.02.2006. u 23:02

Odgovor na temu**peromalosutra**

Puno ime: Ivan Rajkovic
 Član broj: 54774
 Poruke: 216
 Lokacija: Banjaluka
 *.dialup.blic.net.
 OS: Windows XP

[Profil](#) [Email](#) [Privatna Poruka](#)
Re: c++ tutorial sa primjerima

09.04.2006. u 12:52

Poslije duže pauze, idemo dalje...

Funkcije

Funkcije su potprogrami, tj. izdvojeni djelovi programa koji se po potrebi pozivaju iz glavnog programa. Zapravo ono int main() koje smo do sada stavljali na početak programa nije ništa drugo do definicija funkcije koja će se prva pozvati po pokretanju i to je "glavna" funkcija: ona poziva sve ostale funkcije i završetak funkc. main ujedno znaci i završetak glavnog programa. Deklaracija funkcije je sledeća:

Code:

```

tipVarijable/void imeFunkcije
(argumentiKojeFunkcijaPrima/void)
{
    //tijelo funkcije
    ...
    ...
    return povratnaVrijednost; //
ako je prethodno nismo definisali kao void
}

```

Većina funkcija vraća neku (samo jednu) vrijednost i zato ispred imena funkcije stavimo tip varijable koji će funkcija da vrati; ako međutim funkcija ne treba da vrati nikakvu vrijednost tada umjesto tipa varijable stavimo riječ void. Argumente koje funkcija prima definisemo u zagradi iza imena funkcije. Za svaki argument treba posebno navesti njegov tip, a broj argumenata nije ograničen. Kao primjer, evo programa koji sadrži funkciju prost:

Code:

```
#include <iostream>
#include <math.h>
using namespace std;

int prost (long int n);

int main(void)
{
    long int n;
    cout << "n=";
    cin >> n;
    for (int i=2; i<n; i++)
        if (prost(i))
            cout << "Broj " << i << " je prost.\n";
    return 0;
}

int prost (long int n)
{
    for (int i=2; i<=sqrt(n); i++)
        if (n%i==0)
            return 0;
    return 1;
}
```

Kao što vidimo u ovom primjeru, funkcija prost na prvi pogled vraća više vrijednosti (return 1 i return 0), a rekli smo da funkcija može vratiti samo jednu vrijednost. Trik je u tome da kada funkcija naidje na naredbu return odmah završava sa radom i predaje kontrolu funkciji koja ju je pozvala (u našem slučaju main), tako da se sve naredbe iza return naredbe neće izvršiti. Treba opaziti da i funkcija prost poziva još jednu funkciju; to je funkcija sqrt (square root=drugi korijen) definisana u biblioteci math.h. Kada funkcija treba samo da procijeni istinitost nekog izraza (opet kao u našem slučaju), tada za istinu vraćamo bilo koju vrijednost različitu od nule (obično 1), a za laž 0.

Već smo rekli da funkcije mogu da vrate samo jednu vrijednost; međutim šta da radimo ako neka funkcija treba da vrati više vrijednosti ili treba kao rezultat da vrati neki niz? Očigledno je da sa naredbom return to ne možemo, jer smo rekli da funkcija prekida rad poslije prve return naredbe na koju naiđe. Prije nego objasnim kako da se izborimo sa ovim problemom, trebaće nam još malo teorije. Naime, funkcijama možemo prosleđivati argumente na 2 načina: prema vrijednosti i prema adresi. Funkcija prost je argument n (broj koji provjeravamo da li je prost)

dobijala prema vrijednosti, tj. funkcija bi jednostavno kopirala vrijednost datog argumenta i svaka operacija nad tim argumentom bi uticala samo njegovu kopiju u funkciji, a ne i na samu originalnu vrijednost argumenta... Sve će biti jasnije uz primjer:

Code:

```
#include <iostream>
using namespace std;
int promjeni (int vrijednost);

int main(void)
{
    int original=1000;
    cout << original << endl;
    cout << promjeni (original) << endl;
    cout << original << endl;
    system ("pause");
    return 0;
}

int promjeni (int vrijednost)
{
    vrijednost=0;
    return vrijednost;
}
```

Vidimo da vrijednost originalne varijable ostaje nepromjenjena iako ju funkcija mjenja u 0. TO je zato sto funkcija zapravo mjenja vrijednost njene kopije, a ne i same varijable. Očigledno, to nije ono što nam treba! Zato da vidimo drugi slučaj: prosleđivanje vrijednosti prema adresi.

Prosleđivanje vrijednosti prema adresi je nešto komplikovaniji slučaj, i potrebno je poznavanje pokazivača koje cu obraditi u sledecem postu. Sama ideja je jednostavna: umjesto da kopiramo vrijednost varijable i onemogućimo tako funkciji pristup samoj originalnoj varijabli, mi funkciji prosleđujemo adresu varijable tj. njenu lokaciju u memoriji, tako da funkcija sada "zna" gdje je originalna varijabla i može direktno da mijenja vrijednost originala. Na ovaj način funkcija može rezultate svoga rada da snimi u varijable koje smo joj prosljedili kao argumente, a već smo rekli da broj ovih argumenata nije ograničen! Funkciji isto tako možemo predati pokazivač na početak nekog niza i funkcija ce odmah imati direktan pristup svim njegovim vrijednostima. Praktični primjeri slijede u sledecem postu.

09.04.2006. u 12:52

Odgovor na temu**peromalosutra**

Puno ime: Ivan Rajkovic
 Član broj: 54774
 Poruke: 216
 Lokacija: Banjaluka
 *.dialup.blic.net.
 OS: Windows XP

[Profil](#) [Email](#) [Privatna Poruka](#)
**Re: c++ tutorial sa primjerima**

01.05.2006. u 19:40

A evo i jednog zadatka sa takmicenja: Izračunati ukupan broj zrna pšenice koje je Seta (navodni pronalazač šaha) tražio od cara kao nagradu, te ispisati koliko je trebalo "staviti" pšenice na svako od 64 šahovska polja.

Code:

```
#include <iostream>
#define max 50
using namespace std;
```

```
void init(int niz[]);
void ispisi(int niz[]);

int main(void)
{
    int br[max];
    init (br);
    br[0]=1;
    cout << "1. polje:\t" << br[0] << endl;
    for (int n=1; n<=64; n++)
    {
        int temp=0;
        int i=0;
        while (br[i]!=-1)
        {
            br[i]=br[i]*2+temp;
            temp=0;
            if (br[i]>9)
            {
                if (br[i+1]==-1)
                    br[i+1]=0;
                temp=br[i]/10;
                br[i]%=10;
            }
            i++;
        }
        if (n<64)
        {
            cout << n+1 << ". polje:\t"; ispisi(br);
        }
        else
        {
            br[0]-=1;
            cout << "\nUKUPNO:\t\t"; ispisi(br);
        }
    }

    cout << endl << endl;
    system ("pause");
    return EXIT_SUCCESS;
}

void init(int niz[])
{
    for (int i=0; i<max; i++)
        niz[i]=-1;
}

void ispisi(int niz[])
{
    int i=0;
    while (niz[i]!=-1)
        i++;
    for (int j=i-1; j>=0; j--)
        cout << niz[j];
    cout << endl;
}
```

