



Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Низи

Структурно програмирање

Задача

- Да се напише програма која ќе го пресмета просекот на 10 броеви внесени од тастатура.

Задача

- Да се напише програма која ќе го пресмета просекот на 10 броеви внесени од тастатура.

```
#include <iostream>
using namespace std;
int main() {
    int number, n, avg=0;
    cout << "Vnesuvaj broevi:" << endl;
    for(n=0; n<10; n++)
    {
        cin >> number;
        avg+=number;
    }
    cout << "Srednata vrednost na vnesenite broevi e: " <<
(float)avg/n << endl;
    return 0;
}
```

Задача

- Да се напише програма која ќе го пресмета просекот на 10 броеви внесени од тастатура.

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

```
int main() {
```

```
    int number, n, avg=0;
```

```
    cout << "Vnesuvaj broevi:" << endl;
```

```
    for(n=0; n<10; n++)
```

```
    {
```

```
        cin >> number;
```

```
        avg+=number;
```

```
    }
```

```
    cout << "Srednata vrednost na vnesenite broevi e: " <<
(float)avg/n << endl;
```

```
    return 0;
```

```
}
```

**За секој број да се
отпечати дали е под или
над просекот.**

Организација на податоци од ист тип

мал број:

```
int bp1, bp2, bp3;
```

```
int total;
```

```
cin >> bp1 >> bp2 >> bp3;
```

```
total = bp1 + bp2 + bp3;
```

Организација на податоци од ист тип

мал број:

```
int bp1, bp2, bp3;  
int total;
```

4000



bp1

4004



bp2

4008



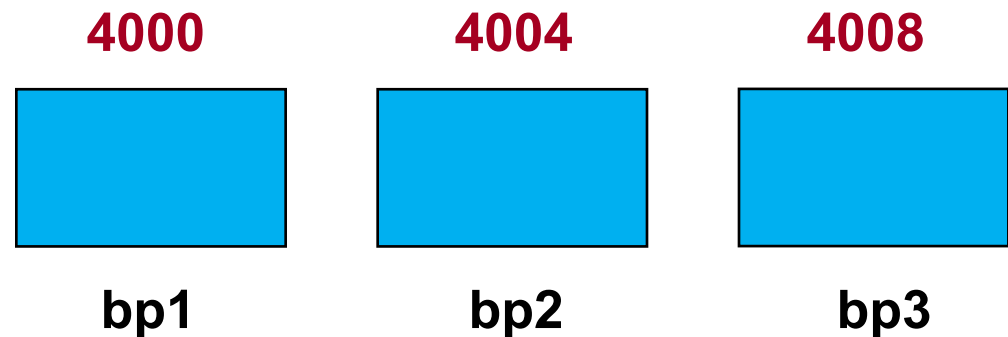
bp3

```
cin >> bp1 >> bp2 >> bp3;  
total = bp1 + bp2 + bp3;
```

Организација на податоци од ист тип

мал број:

```
int bp1, bp2, bp3;
int total;
```



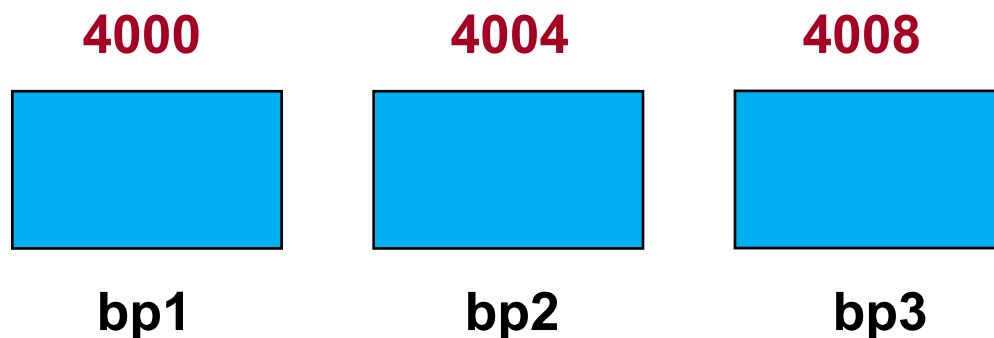
```
cin >> bp1 >> bp2 >> bp3;
total = bp1 + bp2 + bp3;
```

Но, што ако се потребни 100 променливи од ист тип?

Организација на податоци од ист тип

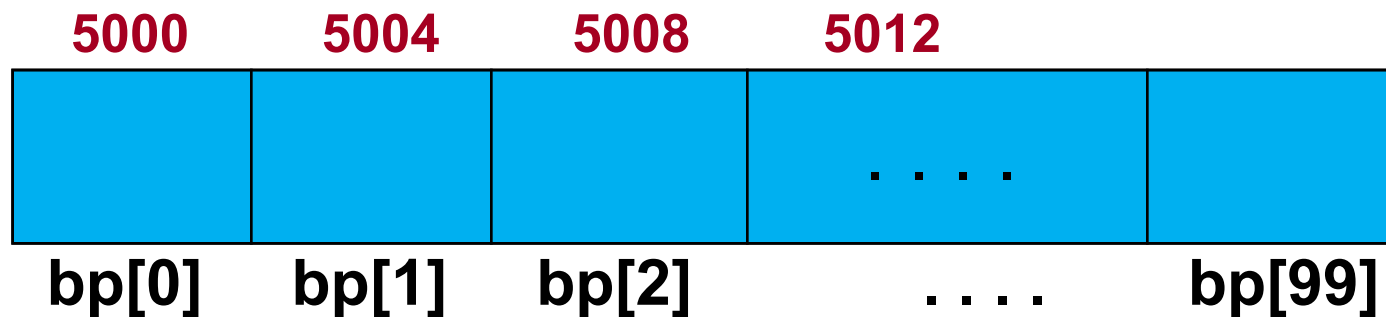
мал број:

```
int bp1, bp2, bp3;
int total;
```



```
cin >> bp1 >> bp2 >> bp3;
total = bp1 + bp2 + bp3;
```

Но, што ако се потребни 100 променливи од ист тип?



Вовед во низи

■ Низа

- структура со релациски поврзани податоци
- бројот на елементи е однапред познат
- сите елементи се променливи од ист тип
- колекција од променливи од ист тип
сместени во низа последователни мемориски
локации, на кои им е доделено единствено
име

■ За пристап до кој и да е елемент од низата се користи името и позицијата на елементот во низата

- позицијата се одредува со **индекс**

Име на низа

Сите елементи имаат
заедничко име **c**

c[0]	-45
c[1]	6
c[2]	0
c[3]	72
c[4]	1543
c[5]	-89
c[6]	0
c[7]	62
c[8]	-3
c[9]	1
c[10]	6453
c[11]	78

Индекс (број)

Ја одредува позицијата
на елементот во низата **c**

Низа

Формална (прецизна) дефиниција:

- Низа е хомогена, линеарна, податочна структура со директен пристап (времето на пристап до секој елемент е константно)

Неформална дефиниција

- Променлива која содржи повеќе елементи од ист податочен тип
- Колекција од повеќе индивидуални елементи со заедничко име (овозможен е пристап до индивидуалните елементи)

Вовед во низи

- Според бројот на димензии низите се делат на
- еднодимензионални,
- дводимензионални (матрици), ИТН...

Име на низи

Сите елементи имаат
заедничко име **c**

c[0]	-45
c[1]	6
c[2]	0
c[3]	72
c[4]	1543
c[5]	-89
c[6]	0
c[7]	62
c[8]	-3
c[9]	1
c[10]	6453
c[11]	78

Индекс (број)

Ја одредува позицијата
на елементот во низата **c**

Што треба да знаеме за низите

- Декларација
- Иницијализација
- Пристап до елементите

Декларирање на еднодимензионална низа

- За декларирање на една еднодимензионална низа потребно е нејзиното име, типот на елементите (променливите) и бројот на елементи во низата

Формат на наредбата

tip ImeNiza[BrojNaElementi];

Примери:

```
int c[10];
```

```
float mojaNiza[3284];
```

```
double Tez[100];
```

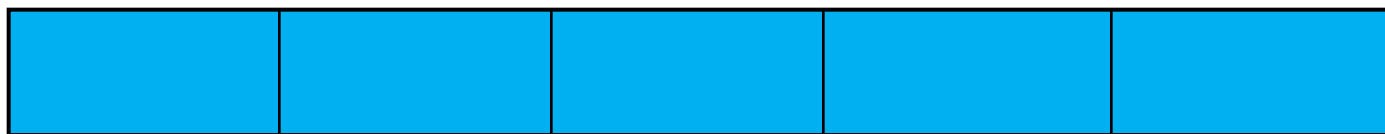
```
int b[100], x[27];
```

Декларирање на еднодимензионална низа

- **Пример:** да се дефинира еднодимензионална низа `c` што содржи 5 индивидуални реални вредности.

`float c[5];` – декларацијата значи
резервирање мемориски простор

број на елементи – мора да биде константа



`c[0]` `c[1]` `c[2]` `c[3]` `c[4]`

индекси на елементите

Декларирање на еднодимензионална низа

Погрешно:

```
int i;
```

```
static int j;
```

```
double a[i];
```

```
int b[j];
```

```
int b[];
```

Иницијализација на еднодимензионална низа

- При декларацијата може и да се иницијализираат елементите на низата

```
int iff[10] = { 1,2,3,4,5,6,7,8,9,10 },  
n[5] = {1, 2, 3, 4, 5 };  
float nums[4] = {1.0, 0.3, 2.25, 4.5};
```

- Ако не се наведени доволно вредности за иницијализирање, најдесните елементи се иницијализираат на 0

```
int jf[5] = {0,1,3}; /* preostanatite elementi se 0 */  
int area[10]={0}; /* najednostaven nacin za inicijalizacija na  
site vrednosti na 0 */  
int year[80] = {97}; /* prviot element ima vrednost 97, a site  
ostanati 0 */
```


Иницијализација на еднодимензионална низа

- Ако се врши иницијализација, големината на низата може да се изостави во декларацијата (се одредува од бројот на иницијализирани елементи)

```
int n[] = { 1, 2, 3, 4, 5 }; /* 5 вредности за  
иницијализација, согласно низата ќе биде декларирано  
како низа со 5 елементи */
```

```
int kf[] = {1,3,5,7,11,13}; /* низа со 6 елементи */
```

```
int size[] = {3,1,5,99,18,-1}; /* низа со 6 елементи  
*/
```

Погрешно:

```
int a[10], b[10];  
a=0; b=a;
```

Пристап до елемент на еднодимензионална низа

- Да се пристапи до елемент од низата, потребно е да се зададе името на низата и позицијата на елементот во низата

- формат:

`imeNiza[IndeksPozicija]`

- за низа со n елементи и име c важи:

`c[0], c[1] ... c[n-1]`

првиот елемент се наоѓа на позиција (има индекс) 0,
вториот на позиција (има индекс) 1, итн.

- индекс на низа може да биде само целобројна вредност
 - одговорност на програмерот е да обезбеди индексот да биде во интервалот $[0, n-1]$!!!

Пристап до елемент на еднодимензионална низа

- Низа може да се иницијализира и со следните наредби

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

пример:

- собирање на вредностите на елементите на две низи и нивно сместување во трета низа

```
for(i=0; i<n; i++)  
    c[i]=a[i]+b[i];
```

Пристап до елемент на еднодимензионална низа

Пример: Ако важи следната дефиниција

```
int array[35];
```

тогаш можни се следните доделувања

Пристап до елемент на еднодимензионална низа

Пример: Ако важи следната дефиниција

```
int array[35];
```

тогаш можни се следните доделувања

```
array[19] = 3 * array[32];
```

Пристап до елемент на еднодимензионална низа

Пример: Ако важи следната дефиниција

```
int array[35];
```

тогаш можни се следните доделувања

```
array[19] = 3 * array[32];
```

```
array[-3] = array[500];
```

Пристап до елемент на еднодимензионална низа

Пример: Ако важи следната дефиниција

```
int array[35];
```

тогаш можни се следните доделувања

```
array[19] = 3 * array[32];
```

```
array[-3] = array[500];
```



бидејќи во C++ не се проверуваат границите, оваа наредба може да се употреби во програмите, но, користење на индекс надвор од границите на низата предизвикува програмата да пристапи до локации надвор од низата

Пристап до елемент на еднодимензионална низа

- Операторот што го одредува индексот има најголем приоритет, поради што во следниот израз

a[2]++

компјутерот ќе ја зголеми вредноста на променливата што се наоѓа на третата позиција во низата a за 1.

Задачата...

Да се напише програма која ќе го пресмета просекот на 10 броеви внесени од тастатура. **За секој број да се отпечати дали е под или над просекот.**

```
#include <iostream>
using namespace std;
int main() {
    int broj[10], n;
    float prosek=0;
    cout << "Vnesuvaj broevi: " << endl;
    for(n=0; n<10; n++)
        cin >> broj[n];
    for(n=0; n<10; n++)
        prosek+=broj[n];
    prosek/=n;
    cout << "Srednata vrednost na vnesenite broevi e: " << prosek <<
endl;
    for(n=0; n<10; n++)
        cout << broj[n] << (broj[n]>prosek?"> ":"<=") << prosek <<
endl;
    return 0;
}
```


Пристап до елемент на низа

Пример: Да се прикаже бројот на деновите во сите месеци во годината.

```
#include <iostream>
using namespace std;
int main()
{
    int i,
    meseci[] = {0,31,28,31,30,31,30,31,31,30,31,30,31};

    for(i = 1; i <= 12; i++)
        cout << "Mesecot broj " << i << " ima " << meseci[i] << "
denovi" << endl;
    return 0;
}
```

Не се користи



Пристап до елемент на низа

Пример: Да се напише програмски код што ќе овозможи проверка дали две низи се идентични

1. Две низи се идентични ако имаат ист број на елементи и ако елементите што се наоѓаат на иста позиција на двете низи имаат идентична вредност;
2. Не е дефинирана наредба $A==B$ (под претпоставка дека A и B се две низи) што овозможува проверка дали две низи имаат идентична содржина, (оваа наредба може да се напише во C , но нејзиното значење е поинакво)

Проверка дали две низи се идентични

```
int IstiSe = 1;
```

Проверка дали две низи се идентични

```
int IstiSe = 1;  
if(n1 == n2) {
```

Проверка дали две низи се идентични

```
int IstiSe = 1;
```

```
if(n1 == n2) {
```

```
} else IstiSe = 0;
```

Проверка дали две низи се идентични

```
int IstiSe = 1;  
if(n1 == n2) {  
    for(i=0; i<n1; ++i)  
  
  
} else IstiSe = 0;
```

Проверка дали две низи се идентични

```
int IstiSe = 1;
if(n1 == n2) {
    for(i=0; i<n1; ++i)
        if(a[i]!=b[i]) {

} else IstiSe = 0;
```


Проверка дали две низи се идентични

```
int IstiSe = 1;
if(n1 == n2) {
    for(i=0; i<n1; ++i)
        if(a[i]!=b[i]) {
            IstiSe = 0;
            break;
        }
} else IstiSe = 0;
```

Проверка дали две низи се идентични

```
int IstiSe = 1;
if(n1 == n2) {
    for(i=0; i<n1; ++i)
        if(a[i]!=b[i]) {
            IstiSe = 0;
            break;
        }
} else IstiSe = 0;
```

Друг начин да се направи истото:

```
for(i=0; a[i]==b[i] && i<n1; i++);
if(i>=n1)    istise...
```

Пример: Хистограм од свезди

```
#include <iostream>
#include <iomanip>
using namespace std;
#define SIZE 10
int main() {
    int n[SIZE] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
    int i, j;
    cout << "Element" << setw(13) << "Value" << "\t\t" <<
    setw(19) << left << "Histogram" << endl;
    for(i = 0; i < SIZE; ++i){
        cout << right;
        cout << setw(7) << i << setw(13) << n[i] << "\t\t";
        for (j = 1; j <= n[i]; j++)
            cout << '*';
        cout << endl;
    }
    return 0;
}
```

Element	Value	Histogram
0	19	*****
1	3	***
2	15	*****
3	7	*****
4	11	*****
5	9	*****
6	13	*****
7	5	*****
8	17	*****
9	1	*

Користење на низи за организација на бројачи

Да се напише програма што ќе изброи колку пати секоја голема буква се појавува во текст внесен од тастатура.

letter	ASCII
'A'	65
'B'	66
'C'	67
'D'	68
.	.
.	.
.	.
'Z'	90

```
const int SIZE=91;
int fC[SIZE]={0};
```

fC[0]	0
fC[1]	0
.	.
.	.
fC[65]	2
fC[66]	0
.	.
.	.
.	.
fC[89]	1
fC[90]	0



бројач за 'A'
бројач за 'B'
.
.
.
бројач за 'Y'
бројач за 'Z'

Реализација на програмата

```
#include <iostream>
using namespace std;
#define SIZE 91
int main () {
    int m, fC[SIZE];
    char ch, index;
    for(m=0; m<SIZE; m++) fC[m]=0;
    while((ch=cin.get()) != EOF){
        if ( ch >= 'A' && ch <= 'Z' ) {
            fC[ch]++;
        }
    }
    cout << "Tekstot sodrzi" << endl;
    cout << "Bukva\t\tBroj na pojavi" << endl;
    for( index = 'A'; index <= 'Z'; index++ )
        cout << index << "\t\t" << fC[index] << endl;
    return 0;
}
```

Реализација на програмата

```
#include <iostream>
using namespace std;
#define SIZE 91
int main () {
    int m, fC[SIZE]={0};
    char ch, index;

    while((ch=cin.get()) != EOF){
        if ( ch >= 'A' && ch <= 'Z' ) {
            fC[ch]++;
        }
    }
    cout << "Tekstot sodrzi" << endl;
    cout << "Bukva\t\tBroj na pojavi" << endl;
    for( index = 'A'; index <= 'Z'; index++ )
        cout << index << "\t\t" << fC[index] << endl;
    return 0;
}
```

Повеќедимензионални низи

- Низите можат да имаат и повеќе димензии
- Така зборуваме за
 - Еднодимензионални низи
 - Дводимензионални низи (матрици)
 - Повеќедимензионални низи (3, 4, ...)

Матрици

■ Матрица или дводимензионални низи

- Табела со редови и колони (m по n елементи)
- При декларација на матрица прво се определуваат редиците, а потоа колоните
- Формат: `tip Ime[Redovi][Koloni];`

	Колона 0	Колона 1	Колона 2	Колона 3
Ред 0	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>
Ред 1	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>
Ред 2	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>

Име на матрица

Индекс на колона

Индекс на ред

Матрици

- Иницијализација во наредба за декларирање

```
int b[2][2]={ {1,2}, {3,4} };
```

Ако нема доволно вредности, елементите за кои недостасуваат вредности се поставуваат на нула

```
int b[2][2]={ {1}, {3,4} };
```

Пример:

```
int niza[3][4] = { {26, 34, 22, 17},
                   {24, 32, 19, 13},
                   {28, 38, 25, 20} };
```

1	2
3	4

1	0
3	4

е еквивалентно со

```
int niza[3][4] = {26,34,22,17,24,32,19,13,28,38,25,20};
```

Првата димензија може да се изостави и матрицата да се декларира на следниот начин:

```
int niza[][4] = { {26, 34, 22, 17},
                  {24, 32, 19, 13},
                  {28, 38, 25, 20} };
```

Матрици

- Пристап до елементи на матрицата - по името се задава редот, а потоа колоната во која припаѓа елементот

```
cout << b[0][1];
```

```
b[1][2] = b[2][1];
```

```
cin >> b[2][0];
```

Пример за употреба на матрици

```
#include <iostream>
using namespace std;
int main(){
    int day_tab[2][13] =
    { {0,31,28,31,30,31,30,31,31,30,31,30,31},
      {0,31,29,31,30,31,30,31,31,30,31,30,31} };
    int i, prest, den, mesec, godina;
    cout << "Vnesi datum: ";
    cin >> den >> mesec >> godina;
    prest = godina%4==0 && godina%100!=0 || godina%400==0;
    for(i=1; i < mesec; i++) den+=day_tab[prest][i];
    cout << "Datumot e " << den << " - iot den vo godinata " << endl;
    return 0;
}
```

Задача

```
#include <iostream>
using namespace std;
#define SIZE 8
int main()
{
    int board[SIZE][SIZE];
    int i,j;

    for(i=0; i<SIZE; i++)
        for(j=0; j<SIZE; j++)
            board[i][j]=(i+j)%2;
    for(i=0; i<SIZE; i++)
    {
        for(j=0; j<SIZE; j++)
            cout << board[i][j];
        cout << endl;
    }
    return 0;
}
```

Да се напише програма која матрица ќе пополни со 1 и 0 во форма на шаховска табла и ќе ја испечати.

```
01010101
10101010
01010101
10101010
01010101
10101010
01010101
10101010
```

Задача

Алтернативно решение

```
#include <iostream>
using namespace std;
#define SIZE 8
int main()
{
    bool board[SIZE][SIZE];
    int i, j;
    bool color = false;
    for(i=0; i<SIZE; color=!color, i++)
        for(j=0; j<SIZE; color=!color, j++)
            board[i][j]=color;

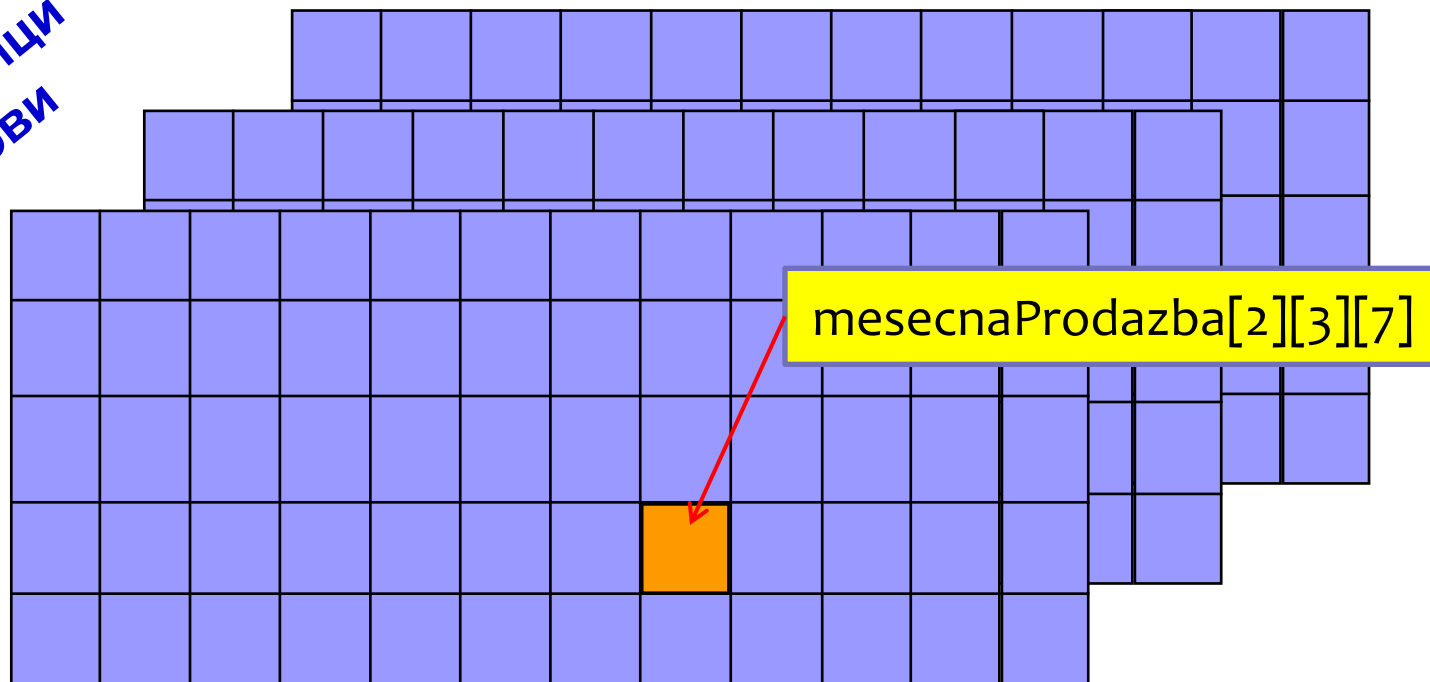
    for(i=0; i<SIZE; putchar('\n'), i++)
        for(j=0; j<SIZE; j++)
            cout << (board[i][j]?'1':'0');
    return 0;
}
```

Пример за декларирање на тридимензионална низа

```
#define ODDELI 5
#define MESECI 12
#define PRODAVNICI 3
int mesecnaProdazba [ PRODAVNICI ][ ODDELI ][ MESECI ];
```

3 продавници
листови

5 оддели
редови



12 месеци колони

Прашања?