Основи програмирања 1
Лекција 6

др Зоран Бањац

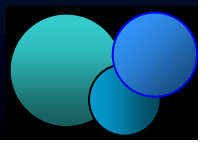
zoran.banjac@viser.edu.rs

Висока школа електротехнике и рачунарства струковних студија
Београд

Садржај

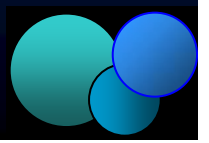
Низови

- Дефинисање низова
- Иницијализација низова
- Приступање елементима низова



Увод

- Подела типова података
  - Основни
  - Сложени
- Основни типови података:
  - `int`
  - `long int`
  - `double`
  - `...`
- Зову се основни, јер променљиве могу истовремено да садрже само једну вредност
- Променљиве СЛОЖЕНОГ типа могу да садрже више од једне вредности

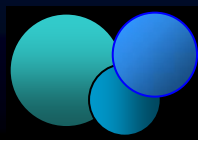


Увод

- Обрада информација за 20, 200 или 2000 студената
- Може да се дефинише 20, 200 или 2000 променљивих

```
int st1, st2, st3, st4, st5, … st2000;
st1 = 0; st2 = 0; …; st2000 = 0;
```

- Много куцања
- Грешке
- ...



Решење...

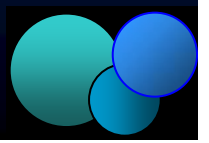
- Да ли може да се користити индексирање:

  st_i - примена индекса *i* за приступ сваком податку

- Да ли је могућ запис следећег облика?

```
for (i=0; i < 2000; i++)
   st_i = 0;
```

za 20 000 студената се мења само један број



Низови (*Array*)

- Синоними: Табеле, Поља, Матрице...
- Сложени тип податка
- Елементи, чланови низа могу бити описани једном вредношћу (скаларни) или су сложени тип податка-описани са више вредности
- Служе за представљање мартрица (вектора), односно више-димензионалних података

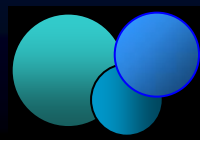
$$\begin{bmatrix} 2 & 5 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 55 & 32 \\ 6 & 83 & 4 \\ 78 & 1 & 7 \end{bmatrix}$$

СВИ ПОДАЦИ У НИЗУ ИМАЈУ ИСТИ ТИП

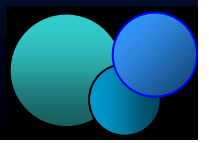
Низови

- Податак унутар низа се назива Елемент низа
- Сваки елемент низа има свој редни број, преко кога се идентификује
- Ти редни бројеви се називају индекси.
- Индекс почиње од нуле (не од јединице!)



Једнодимензионални низови

- Низ или вектор је једнодимензионално поље
- То је скуп података истог типа
- У меморији се смешта у скуп узастопних локација
- Сви подаци у низу имају једно име = ИМЕ НИЗА
- Име низа је почетна адреса низа у меморији (показивачи...)



Једнодимензионални низови

- Сваки елемент низа је одређен :
  - именом низа, и
  - индексом
    (померај у односу на почетак низа)

Дефинисање низова

```
tip ime_niza [duzina1] [duzina2]..[duzinaN];
```

- `tip` било који тип који је до сада коришћен
- `ime_niza` у складу са правилима за идентификаторе
- `duzina` број елемената низа (цео број)

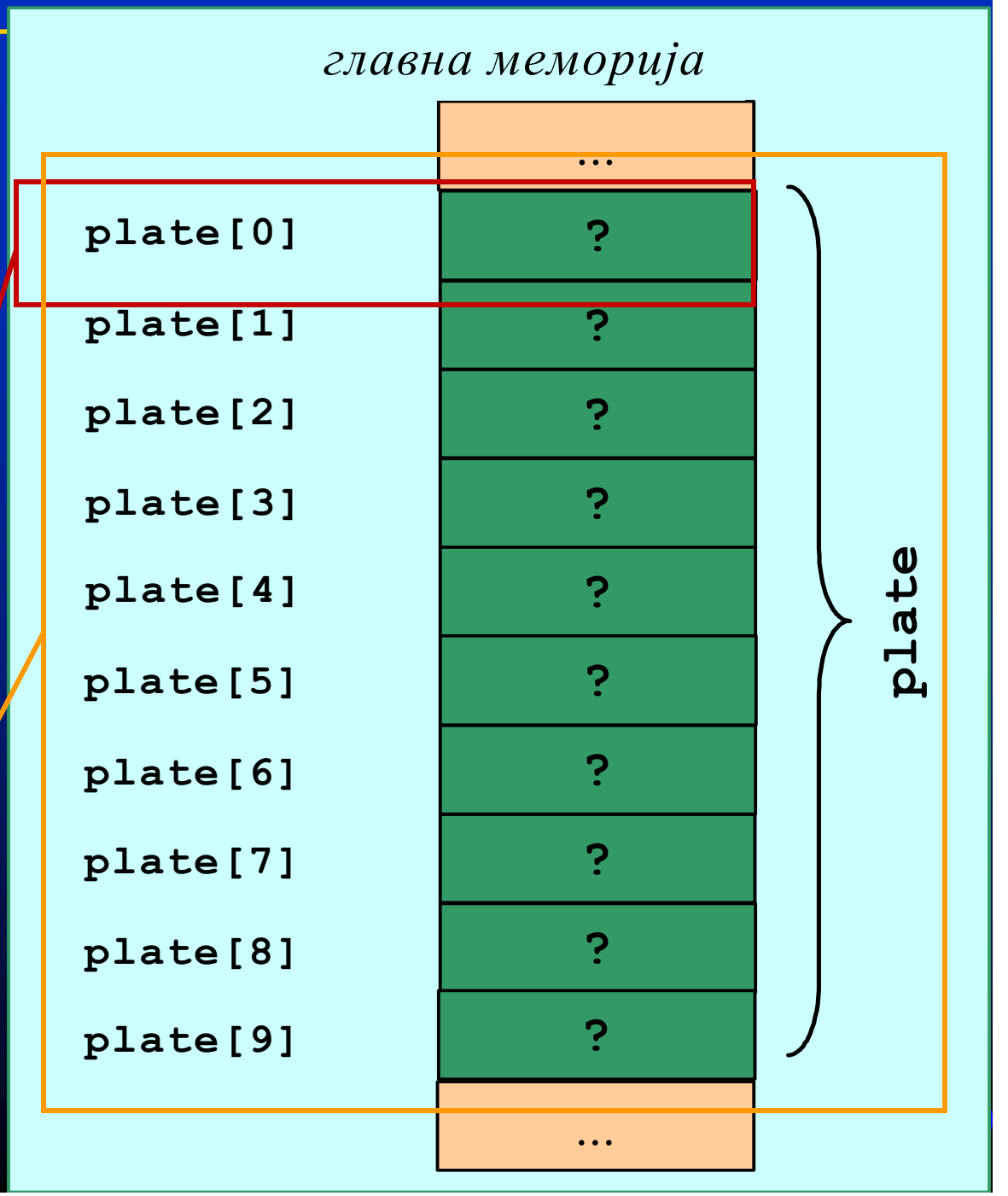
```
float plate[10]; /* 10 elemenata */
int a[10][5];    /* 10x5 elemenata */
```

Дефинисање низова

```
float plate[10];
```

Колико бајтова заузима један елемент низа?

Колико бајтова заузима цео низ?



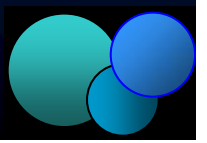
Иницијализација низова

```
tip ime[Duzina] =
{vrednost0, vrednost1,…,vrednostDuzina-1};
```

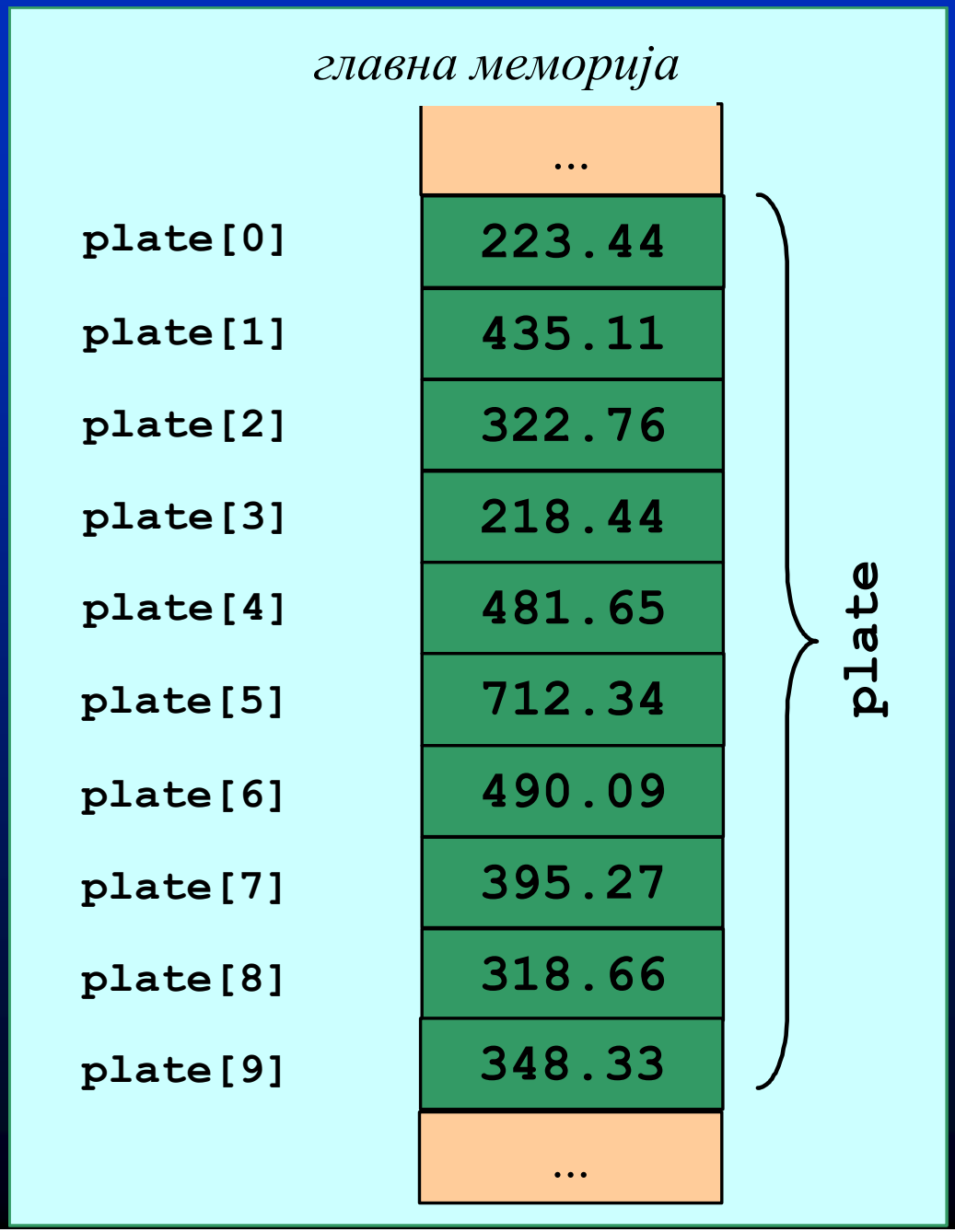
- ***vrednost0*** иницијализује ***ime*[0]**, ***vrednost1*** иницијализује ***ime*[1]**, итд.
- ***vrednost*** мора да буде одговарајућег типа, иначе ће доћи до аутоматске конверзије типова

Примери иницијализације

```
int tablica [4]={3, 5, 7, 9};

long x[6]={3, 5, 7, 9, 0, 0};

float plate[10]={223.44, 435.11, 322.76,
                 218.44, 481.65, 712.34,
                 490.09, 395.27, 318.66,
                 348.33};
```

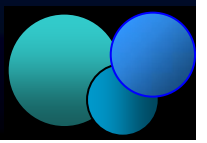


Примери иницијализације



Примери иницијализације

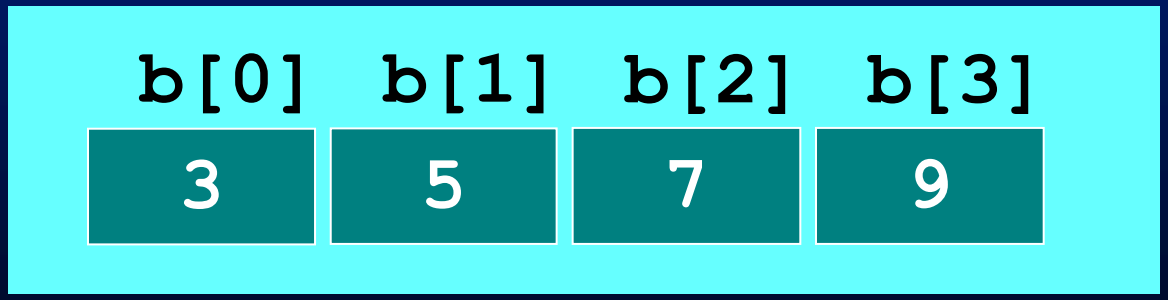
```
int NumDays1[12] = { 31, 28, 31, 30,
   31, 30, 31, 31, 30, 31, 30, 31 };
       /* Jan je 0, Feb je 1, ... */

  int NumDays2[13] = { 0, 31, 28, 31,
    30, 31, 30, 31, 31, 30, 31, 30,31 };
      /* Jan je 1, Feb je 2,... */
```



Примери иницијализације

- Ако се наведу почетне вредности, **`duzina`** може да се изостави приликом дефинисања низа.
- Број вредности одређује дужину низа

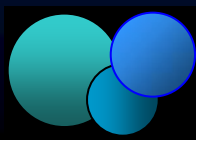
```
int b[ ]={3, 5, 7, 9};
```



Примери иницијализације

- Ако у дефиницији постоји дужина, број наведених вредности за иницијализацију не сме да буде већи од броја елемената низа!

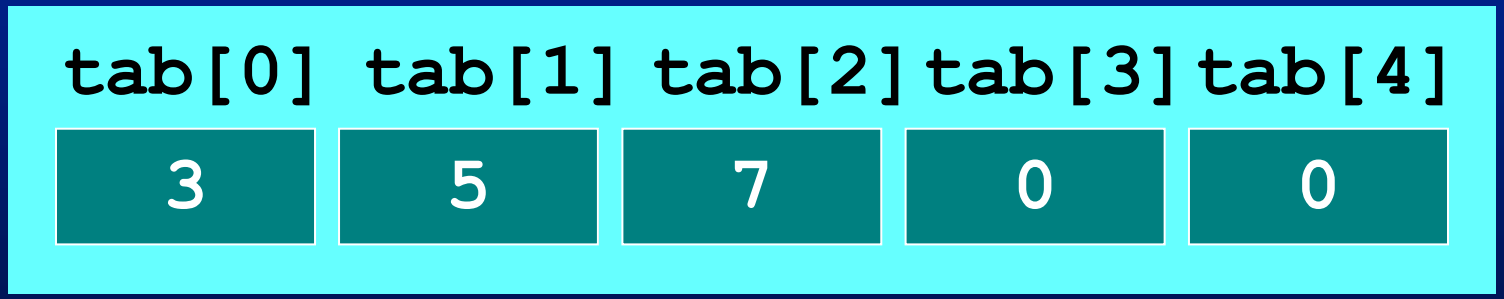
```
int tablica [3]={3, 5, 7, 9};
    /*Greska! Posedice....*/
```



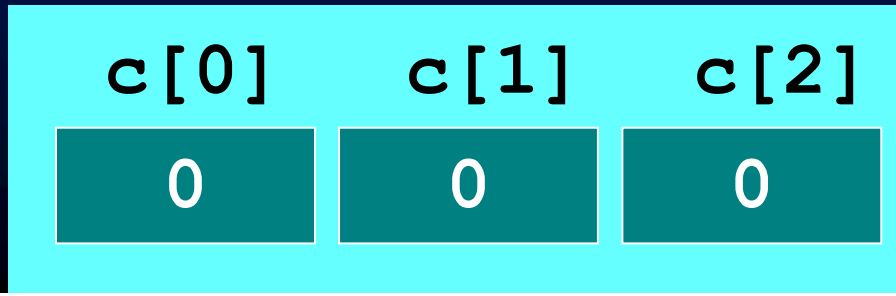
Примери иницијализације

- Ако је број вредности које се додељују мањи од димензија низа оне се додељују елементима на почетку низа а остале ће имати вредност нула.

```
int tab[5]={3, 5, 7};
```




```
float c[3]={0};
```



Низови - рекапитулација

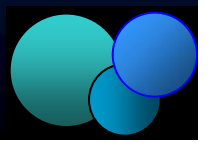
```
tip imeNiza[brojVrednosti]={listaVrednosti};
```

Вредност елемената у низу (иницијализација), није обавезна, вредности се раздвајају зарезом

Број елемената низа. Целобројна константа, резервише потребан број бајтова за елем. низа

Идентификатор, важе правила као и за основне променљиве

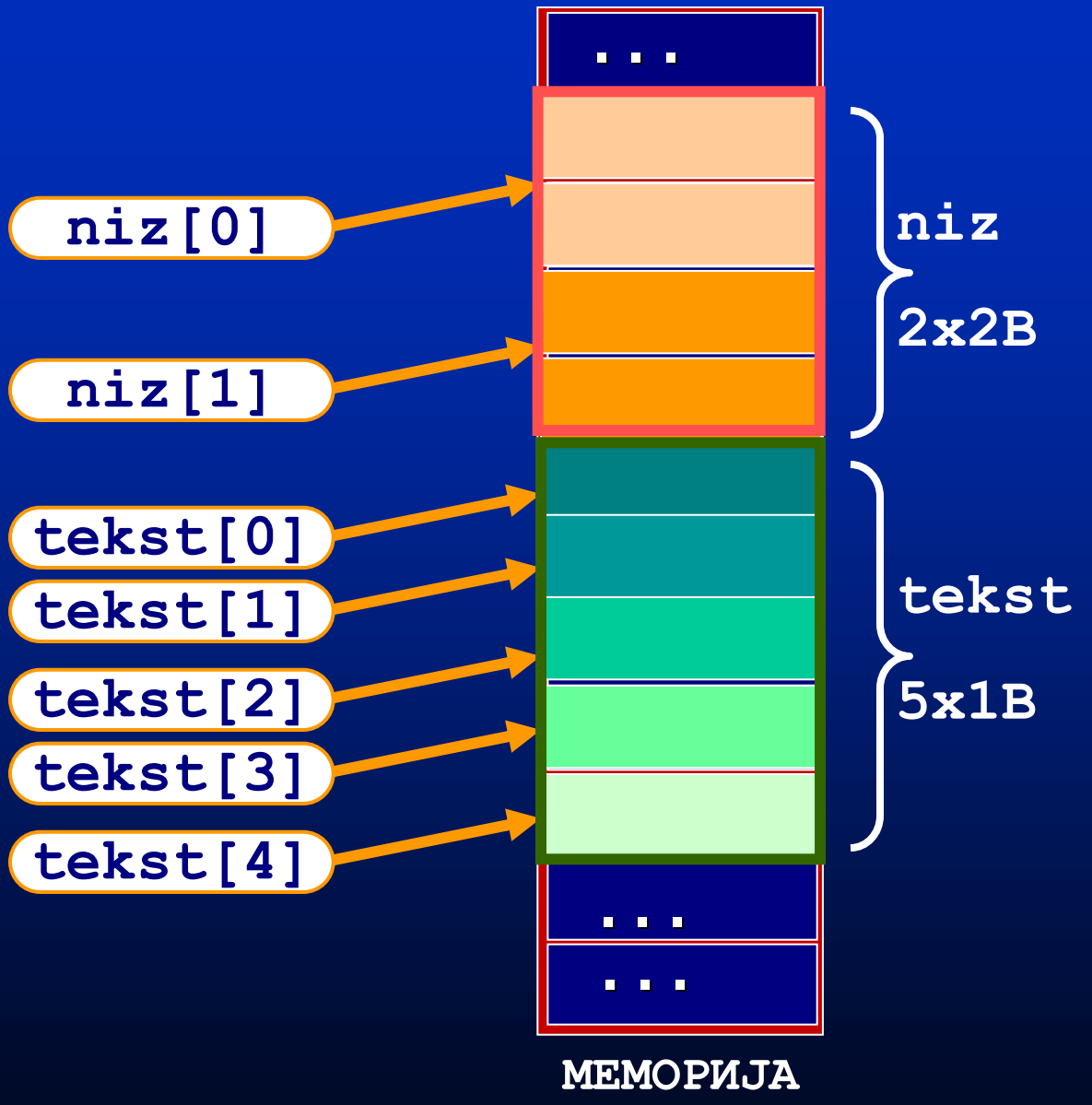
Тип података у низу, сви подаци имају исти тип



Пример декларације низа

```
short int niz[2];
char tekst[5];
```

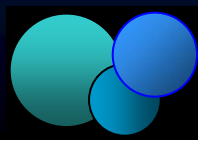
```
#define MAX 2
#define LEN 5
main()
{
 short int niz[MAX];
 char tekst[LEN];
   ...
}
```



Приступање елементима низа

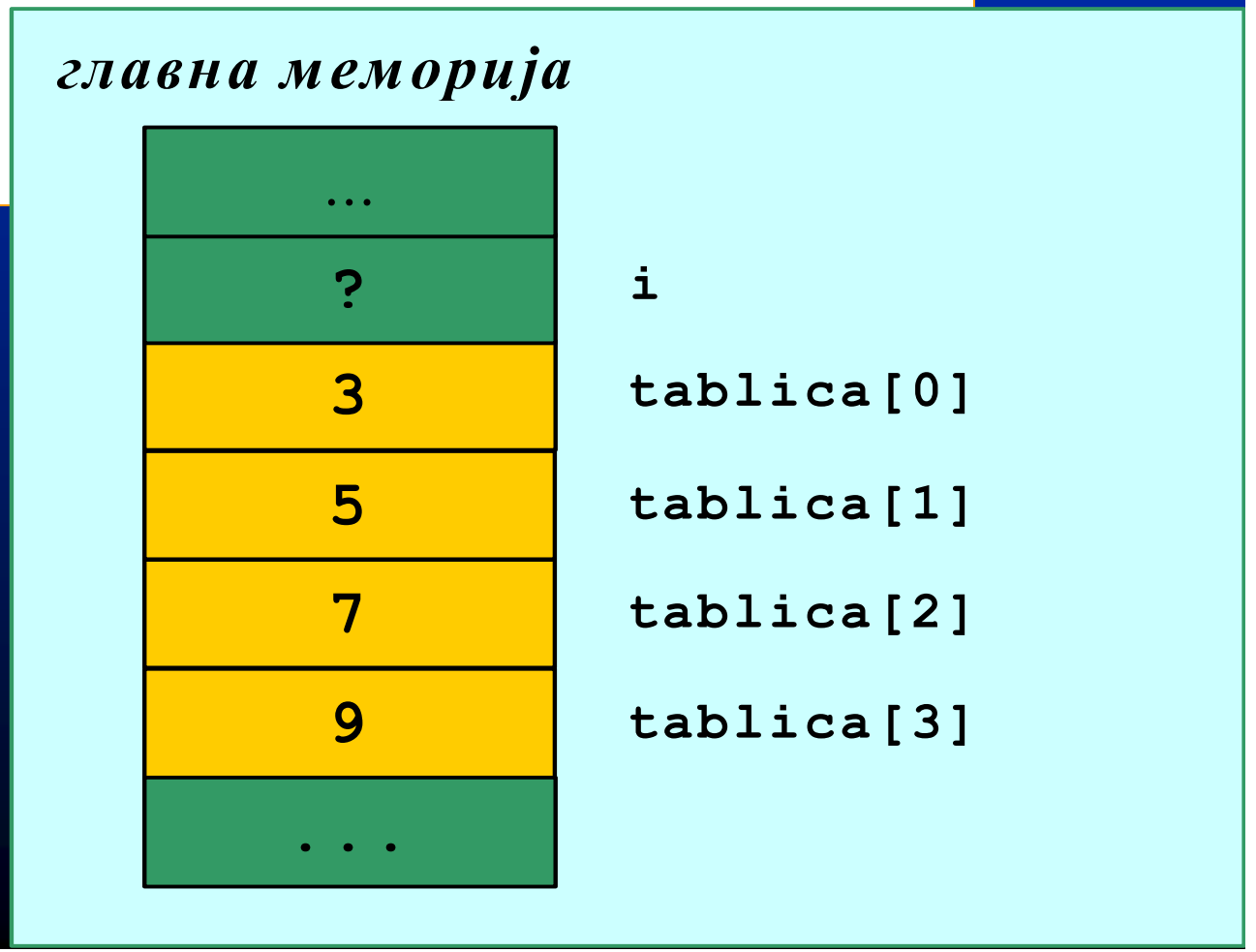
- Елементи низа се идентификују помоћу редног броја који се назива индекс.

- Ако је дужина низа дефинисана преко вредности **`duzina`** онда индекс низа може да има вредност од `0` до (**`duzina-1`**)



Приступање елементима низа

```
int main()
{
   int i, tablica[4]={3, 5, 7, 9};
   ...
}
```



Приступање елементима низа

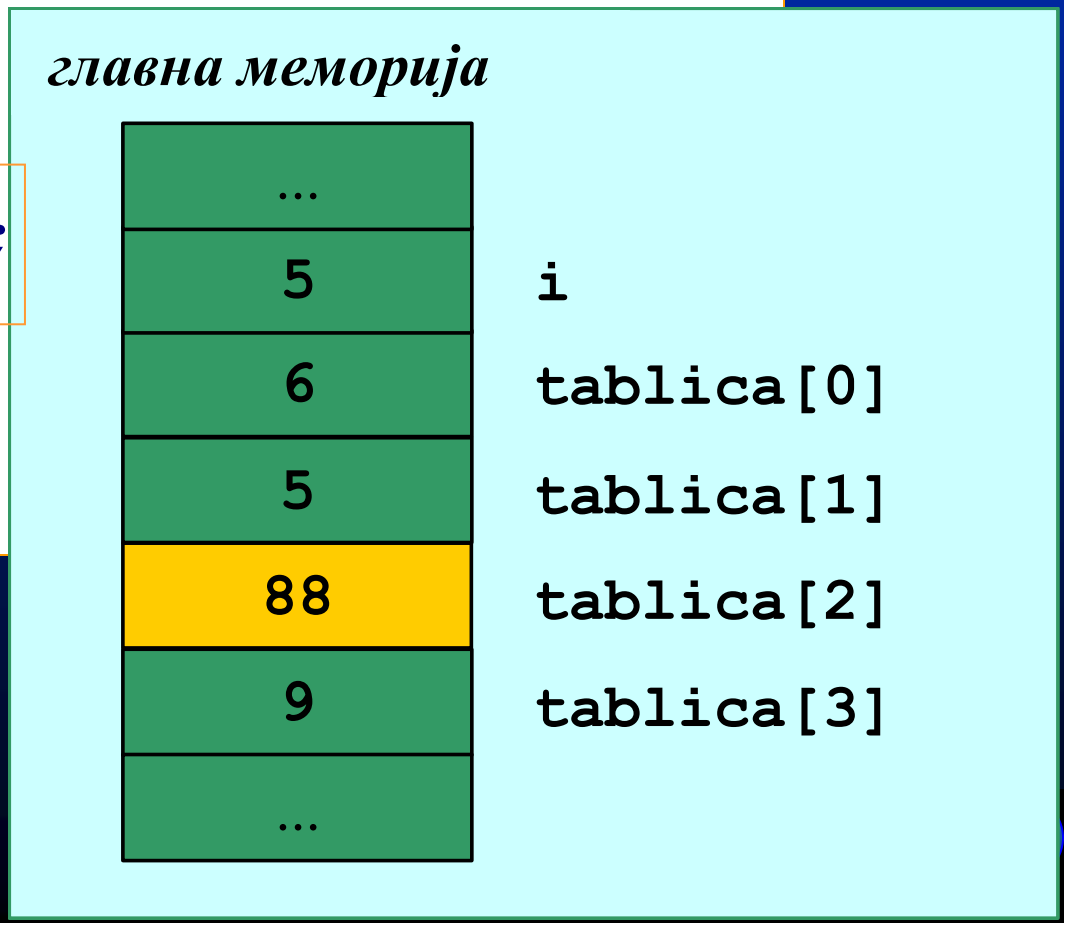
```
int main()
{
   int i, tablica[4]={3, 5, 7, 9};
   i = tablica[1];
   ...
}
```

Приступање елементима низа

```
int main()
{
   int i, tablica[4]={3, 5, 7, 9};
   i = tablica[1];
   tablica[0]=i+1;
      ...
}
```

Приступање елементима низа

```
int main()
{
   int i, tablica[4]={3, 5, 7, 9};
   i = tablica[1];
   tablica[0] = i+1;
   tablica[i-3] = 88;
      ...
}
```



Структура низа

- `int s[10];` према `int s0,s1,...s9`

Вежба

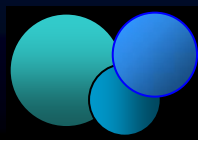
- Са декларацијом:

```
float Sales[10];
x = 1;
```

- Шта је резултат следећих наредби?

```
Sales[5] = 123.45;
Sales[x] = 100.50;
Sales[x + 5] = 20.31;
Sales[x - 1] = 3.46;
```

- Прикажите вредност низа



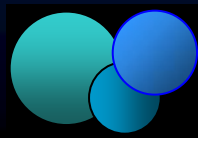
Грешке

Неправилно додељивање вредности:

```
float Sales[10];

Sales = 17.50;          /* nedostaje indeks */
Sales[-1] = 17.50;      /* negativan indeks */
Sales[10] = 17.50;      /* indeks van opsega */
Sales[7.0] = 17.50;     /* tip indeksa */
Sales['a'] = 17.50;     /* tip indeksa */
Sales[7] = 'A';         /* tip podatka */
```

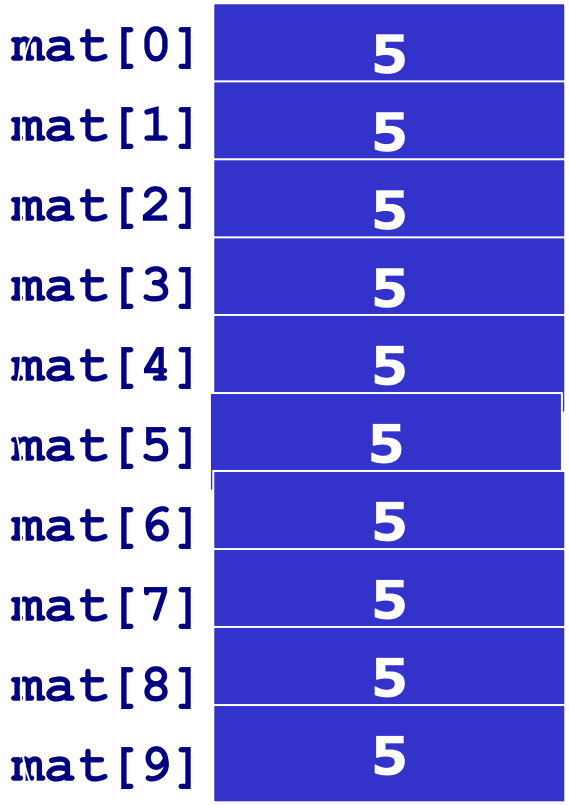
Аутоматска конверзија:

```
Sales[7] = 17;   /* 17.00,  float tip */
```

Додела вредности ...

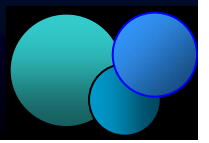
```
int mat[10];
int i;
...

 for(i = 0;i < 10;i++)
 {
    mat[i]=5;
 }
...
```



Индексирање

- При индексирању низа може да се користити било који израз, који као резултат даје цео број
- `[ ]` је оператор, прво израчуна израз а потом тражи ту локацију у низу
- Код примене израза мора да се проверава вредност пре употребе:

```
if ((brojE >= 0) && (brojE <= 9))
  Zarada[brojE] = iznosS;
else
  /* Problem ... */
```



Стил

- Дефинише се константа за дужину низа:

```
#define MAXDUZ 10
```

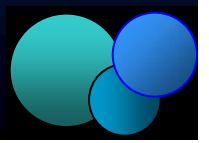
- Користи се константа у декларацији низа:

```
float Plata[MAXDUZ];
```

- Користи се константа у петљама:

```
for (i = 0;i < MAXDUZ;i++)
  Plata[i]=5345;
```

- Ако треба да се **MAXDUZ** промени, онда се то ради само на једном месту у програму

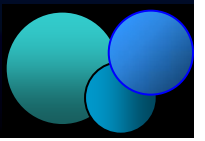


Збир елемената низа

```
#include<stdio.h>
#define RADNICI  4
int main()
{
 float plate[RADNICI] =  {117.0,
       129.95, 276.22, 201.10};
 float zbir = 0.0; int i;

 for(i = 0; i < RADNICI; i++)
   zbir += plate[i];

 printf("Za isplatu: %.2f\n", zbir);
}
```

```
plate
117.00 plate[0]
129.95 plate[1]
276.22 plate[2]
201.10 plate[3]
```

```
Za isplatu:724.47
```

Максимална вредност низа

```
#define RADNICI 4
int main()
{
  float Plate[RADNICI] =
        {117.0, 129.95, 201.10, 276.22};
  float maxP;
  int i;

  maxP = Plate[0];
  for (i = 1; i < RADNICI; i++)
  {
    if (Plate[i] > maxP)
      maxP = Plate[i];
  }
}
```

Минимална вредност низа

```
#define RADNICI 4
int main()
{
  float Plate[RADNICI] =
       {117.0, 129.95, 201.10, 276.22};
  float minP;
  int i;

  minP = Plate[0];
  for (i = 1; i < RADNICI; i++)
  {
    if (Plate[i] < minP)
      minP = Plate[i];
  }
}
```

Испис вредности низа

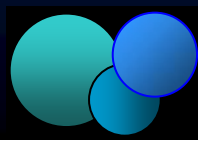
```
#include<stdio.h>
#define RADNICI 4
int main()
{
  float Plate[RADNICI] =
        {117.0, 129.95, 201.10, 276.22};
  int i;

  printf("Radnik broj   Zarada\n");
  printf("--------------------\n");


 for (i= 0; i < RADNICI; i++)
    printf("%4d%17.2f\n", i, Plate[i]);
}
```

```
Radnik broj   Zarada
--------------------
   0          117.00
   1          129.95
   2          201.10
   3          276.22
```

Вишедимензионални низови

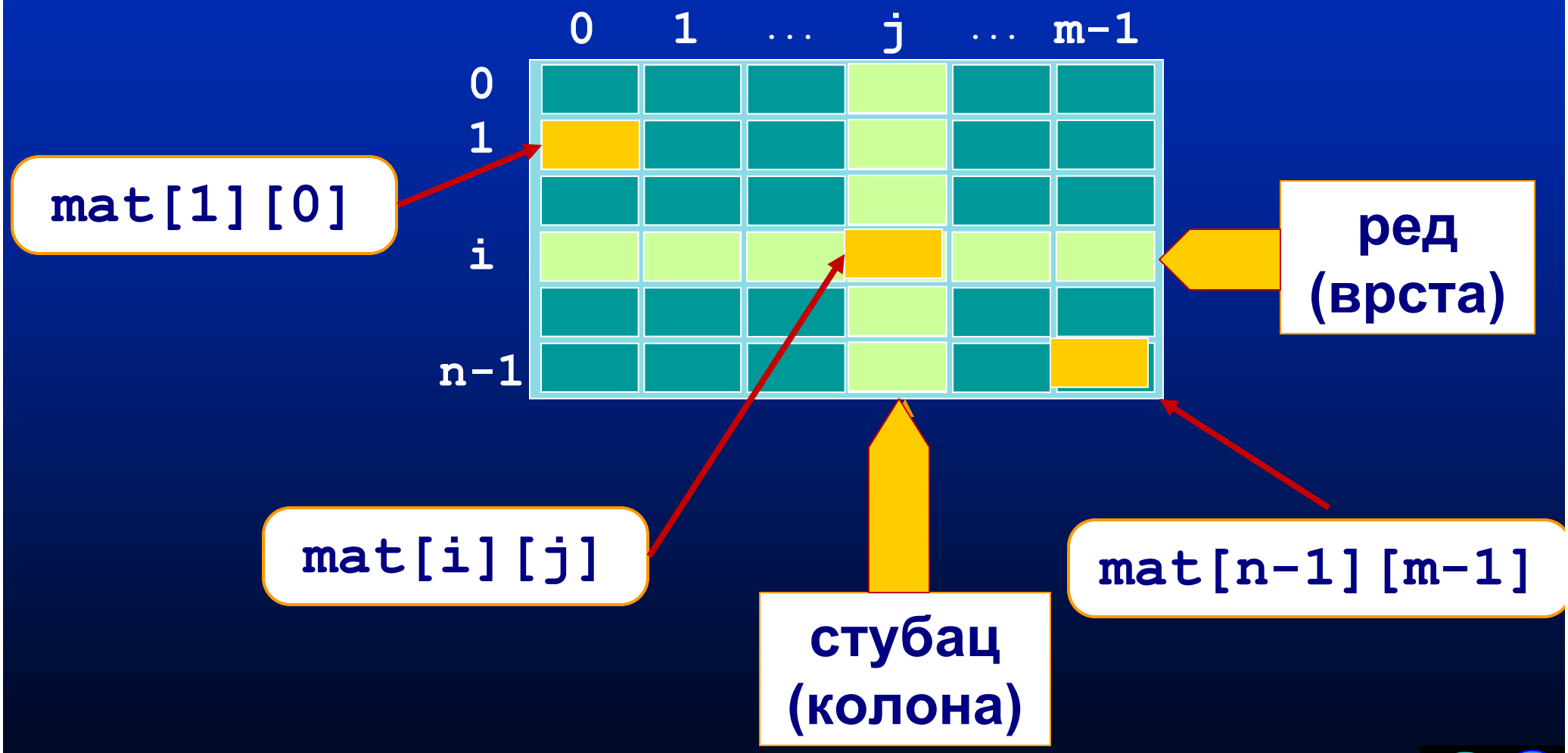
- Дводимензионални-матрице
- три димензије
- четири димензије
- ...

- Елемент низа може да буде други низ



Вишедимензионални низови

- Матрице



Дводимензионална поља матрице

```
tip imeMat[D1][D2]={{V0}, {V1},..{Vn-1}};
```

Вредност елемената у матрици (иницијализација), није обавезна, вредности се раздвајају зарезом

Број елемената низа D1xD2, Целобројне конст. D1 број врста, D2 број колона

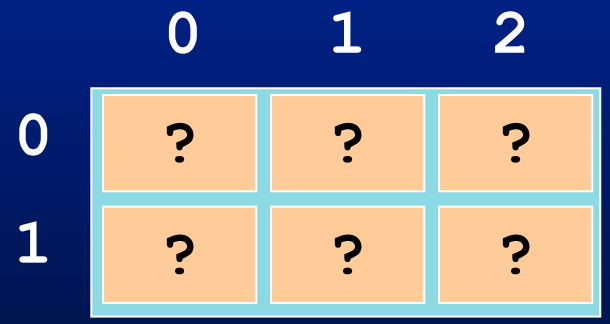
Идентификатор, важе правила као и за основне променљиве

Тип података у низу, сви подаци имају исти тип

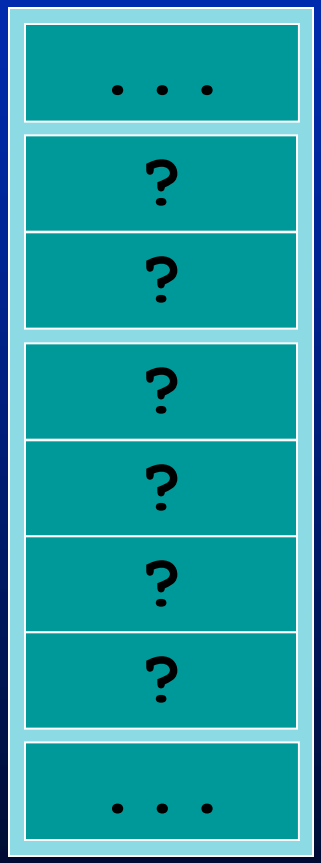
Декларација матрице

- Матрице се попуњавају по принципу ред по ред:

```
char tablica[2][3];
```



Логички изглед



tablica[0][0]
tablica[0][1]
tablica[0][2]
tablica[1][0]
tablica[1][1]
tablica[1][2]

МЕМОРИЈА

Физички изглед

Декларација матрице

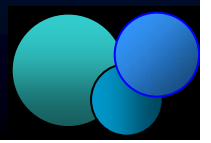
- Пример декларације и иницијализације

```
int mat[2][3] = { {3,1,8}, {2,5,6} };
```

	0	1	2
0	3	1	8
1	2	5	6

```
int mat[2][3] = { {3,1,8} };
```

	0	1	2
0	3	1	8
1	0	0	0



Дефинисање низова

```
int A[10][5];
```

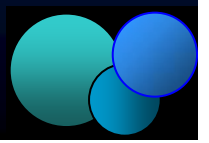
A_{00} A_{01} A_{02} A_{03} A_{04}

A_{10} A_{11} A_{12} A_{13} A_{14}

A_{20} A_{21} A_{22} A_{23} A_{24}

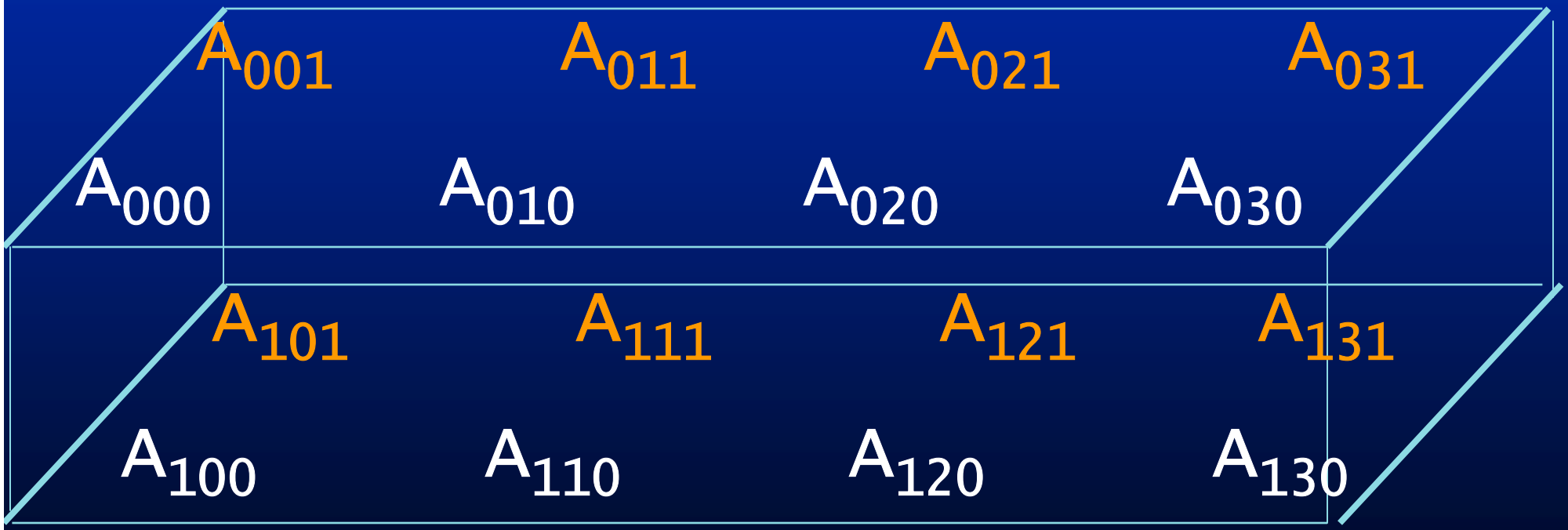
. . .

A_{90} A_{91} A_{92} A_{93} A94



Дефинисање низова

```
int A[2][4][2];
```



Иницијализација

- Више димензионални низови
- Појединачне вредности треба да су и саме низови одговарајућих димензија

```
int mat2[3][2]={
{1,2},
{3,4},
{5,6}
};
```

Логички изглед

```
int mat2[3][2]={{1,2},{3,4},{5,6}};
```

Иницијализација

```
int NumDays2[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}
};
```

```
short kvadratA[4][4]={
{0,1,2,0},{3,4,0,0},{5,6,7,8},{9,0,0,0}
};
```

да ли су исте?

```
short kvadratB[4][4]={
{0,1,2},{3,4},{5,6,7,8},{9}
};
```

Иницијализација

```
int kvA[4][4]={0,1,2,3,4,5,6,7,8,9};
```

да ли су исте?

```
int kvB[4][4]={
{0,1,2,3},{4,5,6,7},{8,9,0,0},{0,0,0,0}
};
```

kvA

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	0	0
3	0	0	0	0

kvB

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	0	0
3	0	0	0	0

Додела вредности ...

- свим елементима матрице додели вредност 5

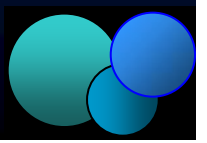
```
main()
{
 int mat[4][3];
 int i,j;

  for( i = 0; i < 4; i++ )
  {
     for( j = 0; j < 3; j++ )
     {
          mat[i][j] = 5 ;
     }
  }
}
```

5	5	5
5	5	5
5	5	5
5	5	5

Учитавање података са таст.

```
for ( red=1; red<=n; red++ )
{
   for ( kol=1; kol<=m; kol++ )
   {
      printf("mat[%d][%d]=",red-1,kol-1);
      scanf("%d", &mat[red-1][kol-1]);
   }
}
```

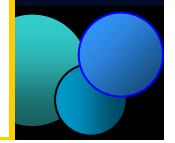
```
for ( red = 0; red < n; red++ )
{
   for ( kol = 0; kol < m; kol++ )
   {
      printf("mat[%d][%d]=",red, kol);
      scanf("%d", &mat[red][kol]);
   }
}
```



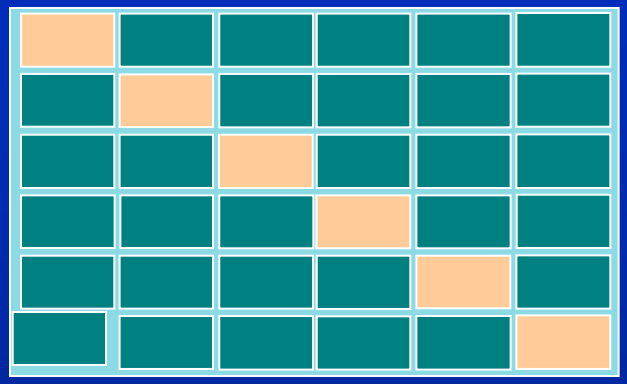
Испис елемената матрице

```
for ( red = 1; red <= n; red++ )
{
   for ( kol = 1; kol <= m; kol++ )
     printf("%4d",mat[red-1][kol-1]);

   printf("\n");
}
```

```
for ( red = 0; red < n; red++ )
{
   for ( kol = 0; kol< m; kol++ )
     printf("%4d",mat[red][kol]);

   printf("\n");
}
```



Елементи главне дијагонале



mat[0][0]
mat[1][1]
mat[i][i]
mat[n-1][n-1]

} mat[i][j], i=j

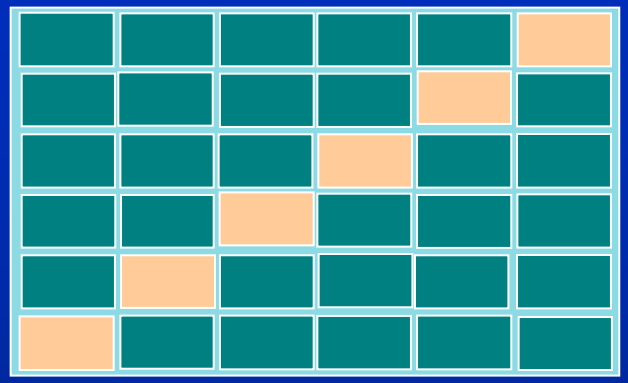
- Приказ елемената главне дијагонале

```
printf("Glavna dijagonala: ");
for ( i = 0; i < n; i++ )
     printf(" %d", mat[i][i]);
```

- Сума елемената главне дијагонале

```
printf("Suma elem. glavne dijagonale:");
for ( s=0,i=0; i < n; i++ )
   s += mat[i][i];
printf(" Suma elemenata GD je %d", s);
```

Елементи споредне дијагонале



mat[0][n-1]
mat[1][n-2]
...
mat[n-1][0]

mat[i][j], i+j=n-1

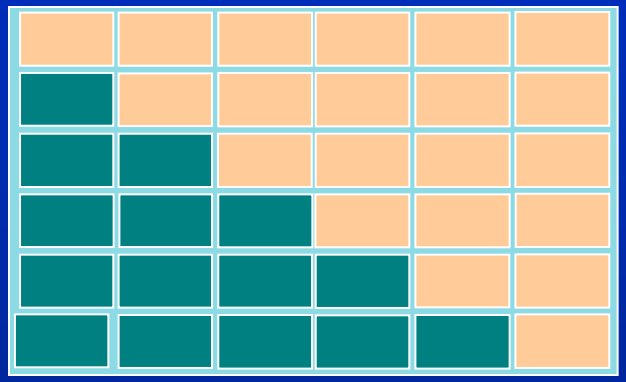
- Приказ елемената споредне дијагонале

```
printf("Sporedna dijagonala: ");
for ( i = n-1; i >= 0; i-- )
     printf(" %d", mat[i][n-1-i]);
```

- Сума елемената споредне дијагонале

```
printf("Suma elem. sporedne dijagonale:");
for ( s=0,i=0; i < n; i++ )
   s += mat[i][n-1-i];
printf(" Suma elemenata SD je %d", s);
```

Елементи горње троугаоне матрице




```
printf("Gornja trougaona matrica:\n ");
for( i = 0; i < n; i++ )
{
  for( j = 1; j <= i; j++ )
     printf("%5c",' ');
  for( j = i; j < n; j++ )
    printf(" %4d", mat[i][j]);
  printf("\n");
}
```

Транспоновање матрице

1	2	3	4
5	6	7	8
9	10	11	12

1	5	9
2	6	10
3	7	11
4	8	12

`mat[n][m]` `matTr[m][n]`

- Испис транспоноване на основу оригиналне

```
printf("Transponovana:\n ");
for ( i = 0; i < m; i++ )
{
  for ( j = 0; j < n; j++ )
      printf(" %4d", mat[j][i]);
  printf("\n");
}
```

Транспоновање матрице

1	2	3	4
5	6	7	8
9	10	11	12

1	5	9
2	6	10
3	7	11
4	8	12

`mat[n][m]` `matTr[m][n]`

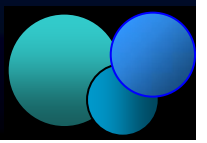
- Формирање транспоноване матрице

```
for ( i = 0; i < m; i++ )
{
  for ( j = 0; j < n; j++ )
  {
    matTr[i][j] = mat[j][i];
  }
}
```

rand()

```
#include<stdlib.h>
int rand( void )
```

- Функција која враћа псеудослучајну целобројну вредност
- Распон вредности: 0 до **RAND_MAX**.
- Не захтева улазне параметре **(void)**
- Дефинисана у **stdlib.h**

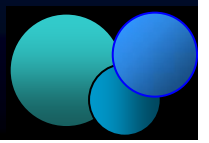


srand()

```
void srand(unsigned int seed)
```

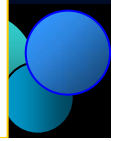
- Поставаља почетну позицију ПСГ (*RNG*)
- Да би почетна вредност сваки пут била другачија

```
srand( (unsigned)time( NULL ) );
```



rand, srand, time

```
#include<stdlib.h>
#include<time.h>
#include<stdio.h>

void main(  )
{
  int i;

  srand( (unsigned)time( NULL ) );
  /* Prikazi 10 brojeva */
  for( i = 0;  i < 10; i++ )
    printf( " %6d\n", rand() );
}
```

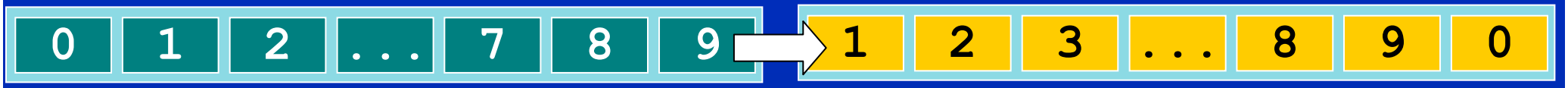


Пример:

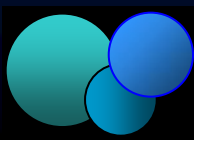
Збир првог елем. низа a и последњег елем. низа b другог елем. низа a и претпоследњег елем. низа b, ...

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 10

main()
{
  int a[MAX], b[MAX], c[MAX], i;

  for(i = 0; i < MAX;i++)
  {
    a[i] = rand();
    b[i] = rand();
  }
  for(i = 0; i < MAX; i++)
  {
    c[i] = a[i] + b[MAX-1-i];
    printf("c[%d]=%d\n",i, c[i]);
  }
}
```

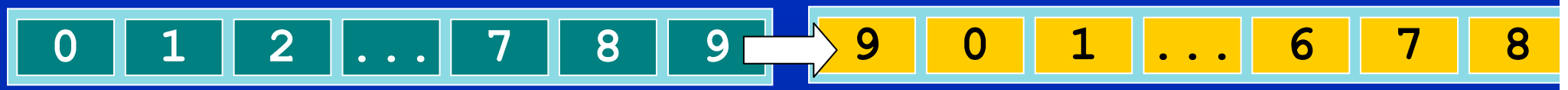
Циклично померање у лево




```
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
main()
{
  int a[MAX], priv, i;

  for(i = 0; i < MAX; i++)
    a[i]=rand();

  priv = a[0];

  for(i = 0; i < MAX-1; i++)
    a[i] = a[i+1];

  a[MAX-1] = priv;
}
```

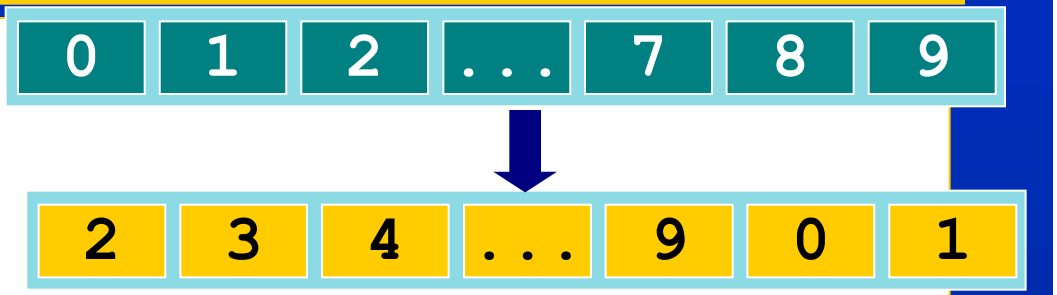



Циклично померање у десно




```
#include<stdio.h>

#define MAX 10
main()
{
  int a[MAX], priv, i;

  for(i = 0; i < MAX; i++)
    a[i] = rand();

  priv = a[MAX-1];

  for(i = MAX-1; i > 0; i--)
    a[i] = a[i-1];

  a[0] = priv;
}
```

Циклични померај за *m* места




```
#include<stdlib.h>
#define MAX 10
main()
{
  int a[MAX], priv, i, m = 2;

  for(i = 0; i < MAX; i++)
       a[i] = rand();
  for(j = 0, j < m, j++)
  {
    priv = a[0];

    for(i = 0; i < MAX-1;i++)
      a[i] = a[i+1];

    a[MAX-1] = priv;
  }
}
```

Сума бита појединих елем. низа

```
#include<stdio.h>

#define DIM 10
main()
{
 short a[DIM]={1,2,3,4,5,6,7,8,9,10};
 short priv, i, j, suma;

 for(i = 0; i < DIM;i++)
 {
  suma = 0;
  for(j = 0; j < 16; j++)
  {
     priv = (a[i] >> j) & 1;
     suma += priv;
  }
  printf("suma bita broja %d je %d\n", a[i], suma);
 }
}
```

Хвала на пажњи

Питања?