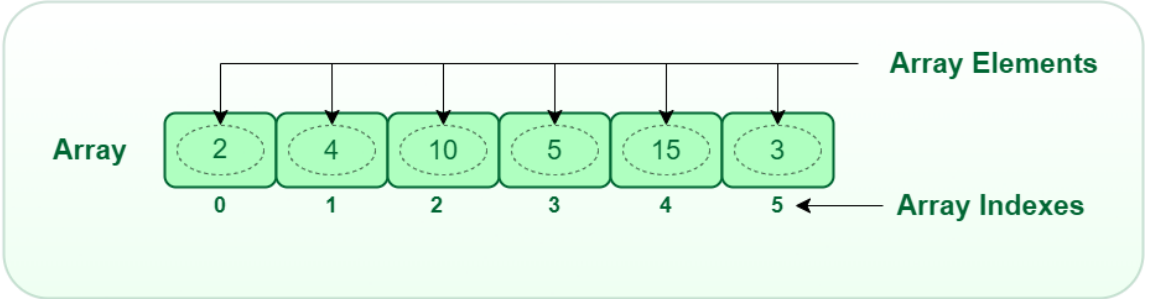
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Операции со низи

РАБОТА СО ЕДНОДИМЕНИЗОНАЛНИ НИЗИ

Потсетување од минатиот час:

- Основна цел на секоја програма е да обработи податоци.
- НИЗА - основна податочна структура која овозможува зачувување на голем број на **податоци од ист тип** (int, float, char, итн.)
- Низите во C++ гарантираат секвенцијалност на елементите, односно елементите се наредени **едноподруго**.



Изминување на низа

- Подразбира пристапување на сите елементи во специфичен редослед
  - Веќе го правиме кога ја пополнуваме низата со елементи преку стандарден влез
  - Кога ја печатиме низата
- Редоследот може да биде произволен, но најчесто се однесува на една од следниве 2 насоки:
  - Нанапред (од почетокот на низата кон крајот)
  - Наназад (од последниот елемент на низата кон почетокот)

Изминување на низа

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

int main() {
  // Declaration of the array
  const int MAX = 100;
  int a[MAX] = {0};

  int n;
  // Filling up the first n elements
  // of the array with specific values
  cin >> n;
  for (int i = 0; i < n; i++) {
    cout<< "a[" <<i <<"]:";
    cin >> a[i];
  }

  // Printing the array to STDOUT
  for (int i = 0; i < n; i++) {
    cout<< a[i] << "\n";
  }
  return 0;
}
```

Излез:

```
5
a[0]:1
a[1]:2
a[2]:3
a[3]:4
a[4]:5
1
2
3
4
5
```

Изминување на низа нанапред

Изминување на низа

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

int main() {
  // Declaration of the array
  const int MAX = 100;
  int a[MAX] = {0};

  int n;
  // Filling up the first n elements
  // of the array with specific values
  cin >> n;
  for (int i = n-1; i >= 0; i--) {
    cout<< "a[" <<i <<"]:";
    cin >> a[i];
  }

  // Printing the array to STDOUT
  for (int i = 0; i < n; i++) {
    cout<< a[i] << "\n";
  }
  return 0;
}
```

Излез:

```
5
a[4]:10
a[3]:20
a[2]:30
a[1]:40
a[0]:50
50
40
30
20
10
```

Изминување на низа наназад

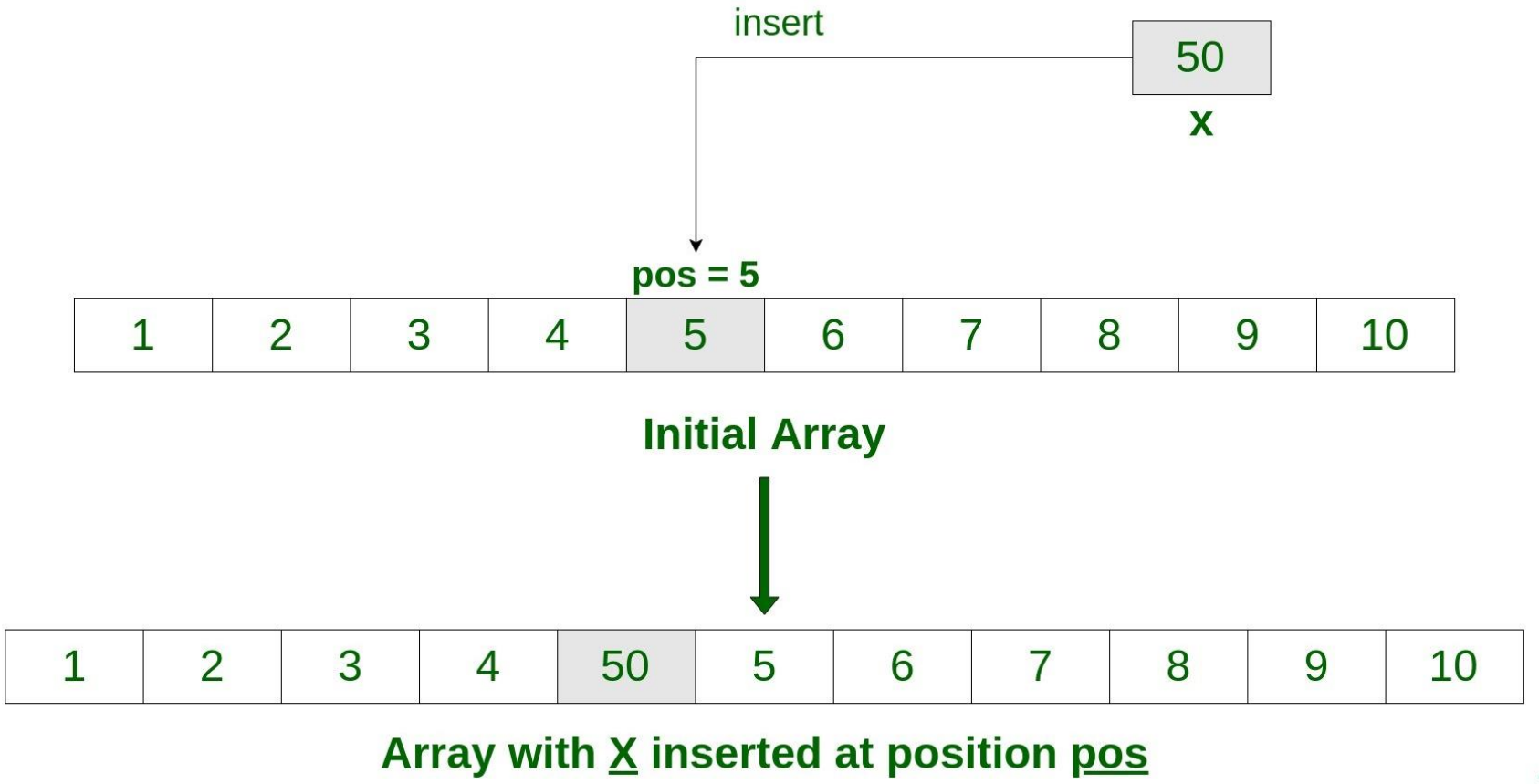
Изминување на низа нанапред

Вметнување елемент

- Кога се додава елемент во низа, потребно е да се знае дали низата има капацитет (алоцирана меморија) за да го сочува елементот

- *Низата има капацитет за да го смести новиот елемент (има непополнети елементи на крај)*

Вметнување елемент



Вметнување елемент

Случај А: **Низата има капацитет за да го смести новиот елемент (има непополнети елементи на крај)**

1. Најди ја локацијата (индексот) каде треба да се вметне елементот
2. Ако веќе постои елемент на таа локација, премести го елементот и сите елементи по него надесно
3. Додади го елементот во низата на специфичниот индекс
4. Зголеми го бројот на елементи на низата

Вметнување елемент

```
// Function to insert element
// at a specific position
void insertElement(int arr[], int &n, int el, int pos)
{
  // shift elements to the right
  // which are on the right side of pos
  for (int i = n - 1; i >= pos; i--)
    arr[i + 1] = arr[i];

  arr[pos] = el; // insert the element in the array
  n++; // increment the number of elements in the array
}
```

Пребарување на низа

- Многу честа операција која се сведува на:

„Најди го елементот со соодветна вредност“

Линеарно пребарување на низа

```
// Find if the array has an element with value key
   int linear_search(int arr[], int n, int key) {
     // Traverse the array
     for (int i = 0; i < n; i++){
       // Check if the key is found
       if (arr[i] == key) {
         // Key found, stop searching
         // and return the position
         return i;
       }
     }
     // If the key is not found,
     return -1;
   }
```

Подредедување (сортирање)

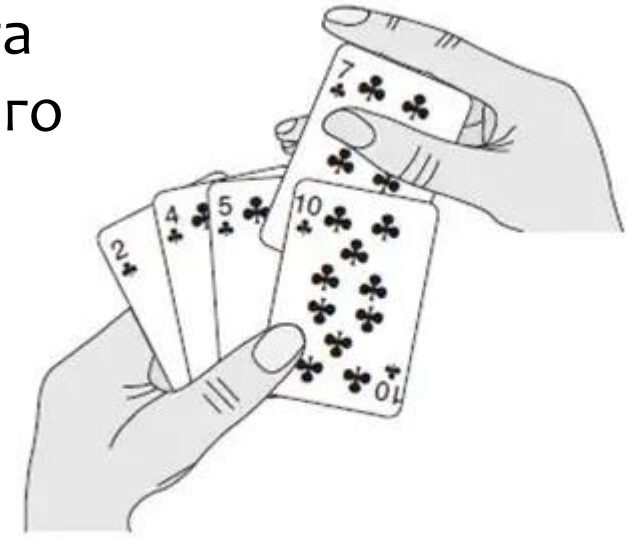
- Многу од операциите врз колекција од елементи се поедноставуваат и убрзуваат ако истите се подредени.

- Оттука, пожелно е да проучиме и научиме неколку пристапи за сортирање на низи.

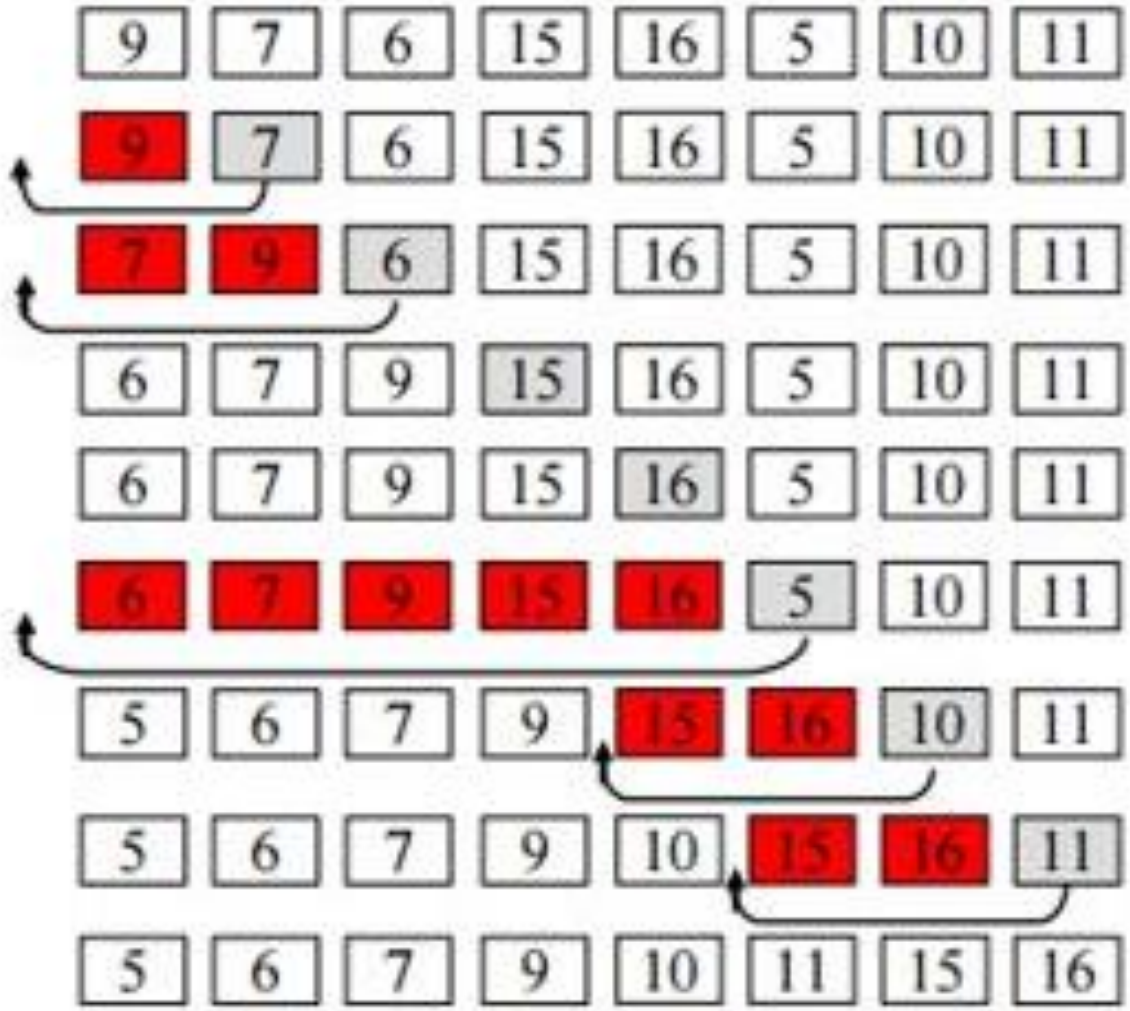
Пример: Подредување со вметнување

- **Пристап:** Подели ја низата на сортиран и несортиран дел, и секој елемент од несортираниот дел вметни го на соодветното место во сортираниот дел

- Изминувај го секој елемент од низата (почнувајќи од почеток), споредувај го со сите елементи пред него и вметни го на соодветната локација.



Подредување со вметнување



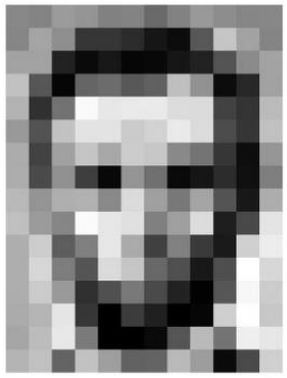
Подредување со вметнување

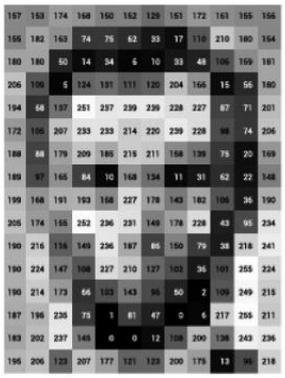
```
void insertionSort(int arr[], int n) {
     int i, key, j;
     for (i = 1; i < n; i++) {
       key = arr[i];
       j = i - 1;

       // Move elements of arr[0..i-1],
       // that are greater than key,
       // to one position ahead of their
       // current position
       while (j >= 0 && arr[j] > key) {
         arr[j + 1] = arr[j];
         j = j - 1;
       }
       arr[j + 1] = key;
     }
   }
```

ПОВЕ́ЌЕДИМЕНЗИОНАЛНИ НИЗИ (МАТРИЦИ)

Потсетување

- Кај повеќедимензионалните низи потребни се повеќе од 1 број (индекс) за да се пристапи до соодветен елемент.
- Кај дводимензионалните (2Д) низи се воведуваат редици и колони
- Имаат широка употреба (наједноставен пример - слики)





Ред на матрица

- Ред на матрица претставува бројот на редици и колони во истата

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}_{3x4}$$

- Во горниот пример, редот на матрицата е 3x4 (3 реда и 4 колони).

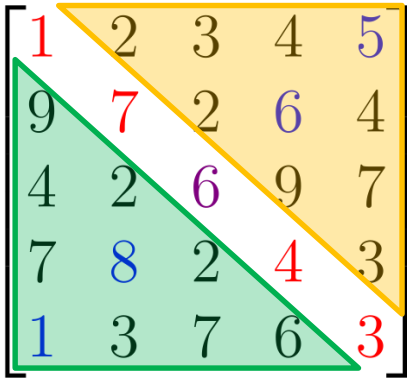
Особини на квадратни матрици

- Кај квадратните матрици може да дефинираме две дијагонали
  - Главна дијагонала (црвено и виолетово)
  - Споредна дијагонала (сино и виолетово)

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 9 & 7 & 2 & 6 & 4 \\ 4 & 2 & 6 & 9 & 7 \\ 7 & 8 & 2 & 4 & 3 \\ 1 & 3 & 7 & 6 & 3 \end{bmatrix}$$

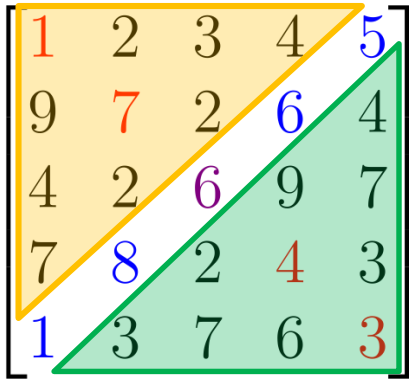
Особини на главна дијагонала

- За елементите **од** главната дијагонала важи дека индексот на редицата е еднаков со индексот на колоната (i == j)
- За елементите **над** главната дијагонала (во портокаловиот триаголник) важи дека индексот на колоните е поголем од индексот на редиците (i < j)
- За елементите **под** главната дијагонала (во зелениот триаголник) важи дека индексот на колоните е помал о д индексот на редиците (i > j)



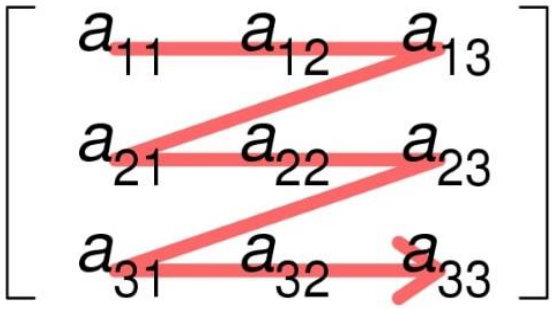
Особини на споредна дијагонала

- За елементите **од** споредната дијагонала важи дека сумата на индексите е константна (во C++, i + j = n-1, каде n е бројот на редици/колони)
- За елементите **над** споредната дијагонала (во портокаловиот триаголник) важи дека сумата на индексите е помала од n-1
- За елементите **под** споредната дијагонала (во зелениот триаголник) важи дека сумата на индексите е поголема од n-1



Изминување и пополнување на матрица

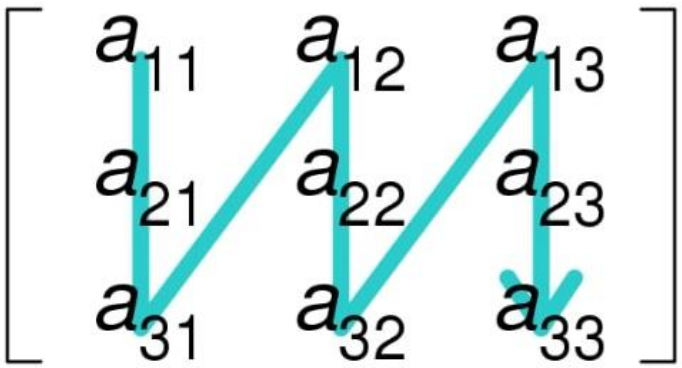
- Ред по ред




```
// Declaration of the matrix
const int MAX = 10;
int mat[MAX][MAX];

int n, m;
// Reading the matrix dimensions
cin >> n >> m;

// Reading the matrix row-by-row
for (int i = 0; i < n; i++) {
  for (int j = 0; j < m; j++) {
    cin >> mat[i][j];
  }
}
```

Изминување и пополнување на матрица

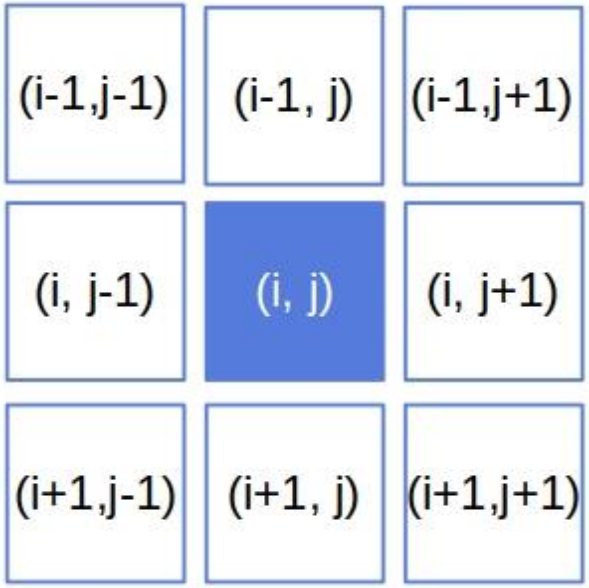
- Колона по колона



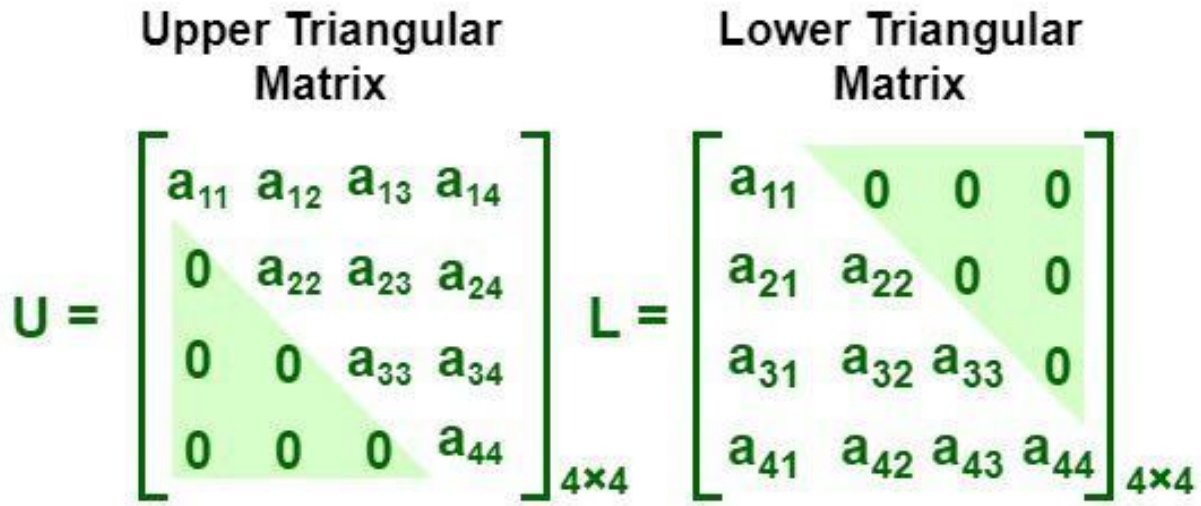

```
// Declaration of the matrix
const int MAX = 10;
int mat[MAX][MAX];

int n, m;
// Reading the matrix dimensions
cin >> n >> m;

// Reading the matrix col-by-col
// Notise the reversed indices
for (int i = 0; i < m; i++) {
  for (int j = 0; j < n; j++) {
    cin >> mat[j][i];
  }
}
```

Соседи на даден елемент

- Сите елементи во 2д матрица имаат соседи
- Елементите на ќошињата и рабовите од матрицата имаат по 3 и 5 соседи соодветно.
- Сите останати елементи имаат 8 соседи
- Индексите на соседите за даден елемент (i, j), се:



Триаголни матрици




```
// Konverzija vo Gornotriagolna matrica
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
            matrix[i][j] = 0;
 // POD glavna dijagonala vo 0        }
    }
```

```
// Konverzija vo Dolnotriagolna matrica

for (int i = 0; i < n; ++i)
 { for (int j = i + 1; j < n; ++j)
     { matrix[i][j] = 0;
// NAD glavna dijagonala vo 0 } }
```

Задача

- C++ програма што го печати прикажаниот дел на дадената 3 x 3 квадратна матрица.

Влез
1 2 3
4 5 6
7 8 9

Излез

1 2 3
4 5
7

```
int main()
{
    int arr[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

for (int i = 0; i < 3; i++)
    {
       for (int j = 0; j < (3 - i); j++)
       {
           cout << arr[i][j] << "  ";
       }
       cout << "\n";
    }
}
```

Прашања?