Многомерни масиви

Изготвил: М. Денишева-Илиева

1. Възможни приложения

- В едномерните масиви се съхраняват данни, които се разполагат в линия (в математиката казваме редица). Така всеки елемент на едномерния масив се идентифицира с един индекс, който сочи мястото на елемента в редицата.
- Много често, обаче, се налага да се използват и данни, които са подредени в табличен вид. Например годишните оценки на всички ученици по всички предмети е логично да бъдат записани в таблица, в която редовете да съответстват на учениците, а колоните да съответстват на изучаваните предмети.

- Например:

	БЕЛ	Чужд език	Математика	Информатика	ИТ
Александър	6	4	5	6	6
Диана	5	6	4	6	6
Мартин	6	6	4	5	5

- За такива случаи езикът C# предлага структурата двумерен масив.

2. Двумерен масив

- В двумерните масиви данните са разположени в редове и стълбове. Затова всеки елемент на масива се идентифицира с два индекса – първият е номер на реда, а вторият – е номера на стълба.
- В езика C# номерецията на редовете и стълбовете, както и номерацията в едномерните масиви започва от нула(0).

	0	1	2	3
0	2	1	3	11
1	5	6	7	8
2	4	12	9	10

и 4 стълба.

Матрица

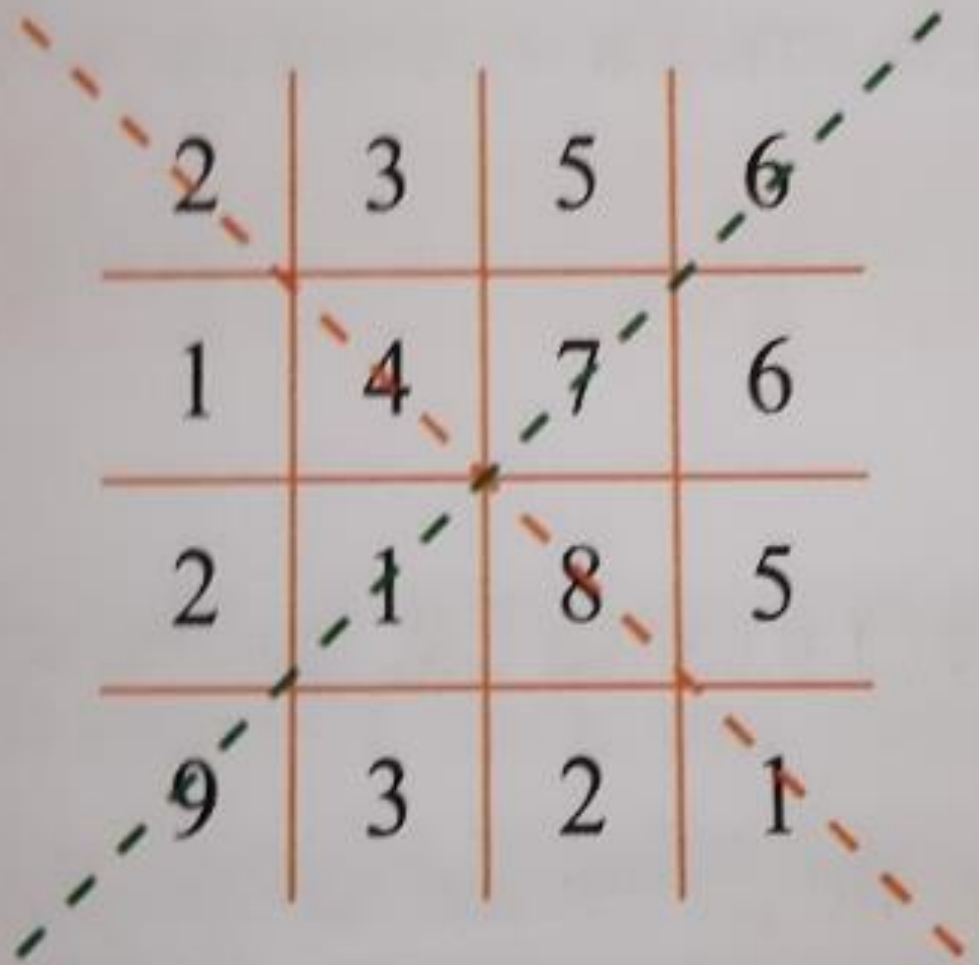
- Структура, при която индексирането на клетките е следното:

	0	1	2	3
0	a_{00}	a_{01}	a_{02}	a_{03}
1	a_{10}	a_{11}	a_{12}	a_{13}
2	a_{20}	a_{21}	a_{22}	a_{23}

- Когато в двумерен масив броят на редовете е равен на броя на стълбовете (като например в играта Судоку) е дефинирано и понятието **диагонал**.
- **Главен диагонал** наричаме редицата от елементи, която зависи от горния ляв ъгъл и минава през точно един елемент от всеки ред и стълб, докато достигне долния десен ъгъл.
- **Обратен диагонал (вторичен диагонал),** наричаме този диагонал, който започва от горния десен ъгъл и минава през елемент от всеки ред и стълб стига до долния ляв ъгъл.

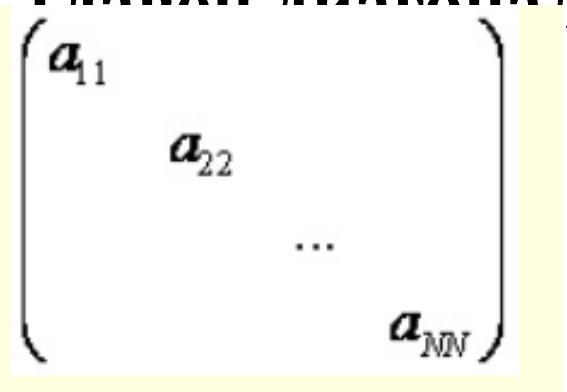
главен диагонал

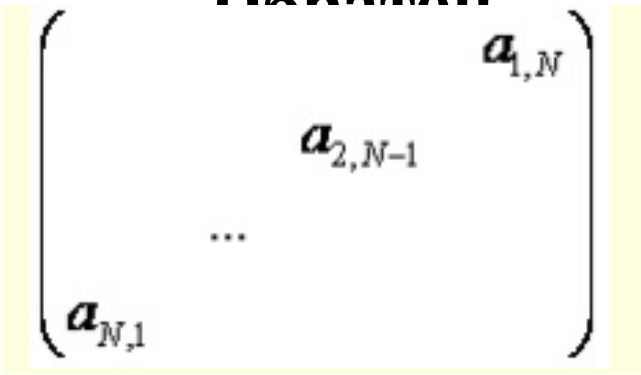
обратен диагонал



Фиг. 2

- От дефиницията на главен и обратен диагонал се вижда, че елементите от главния диагонал имат еднакви индекси.
- За елементите от обратния диагонал сумата на индексите е постоянно число (в примера от фиг.2 това е 3). Това означава, че ако масивът има n реда и **n** стълба, то в общия случай сумата на двата индекса за елементите от обратния диагонал е **n-1.**
- **Главен диагонал** **Обратен**





3. Създаване и инициализация.

- **Синтаксис на оператора за деклариране и заделяне на памет за двумерен масив:**

<тип>[,] <име> = new <тип>[<брой редове>,<брой стълбове>];

Например: int[,] intMatrix = new int[4,6];

И в този случай разположението на масива в паметта може да се направи отделно от декларирането, както при едномерните масиви, но не бива да се забравя, че **преди да бъде разположен в паметта масивът не може да се използва**. Това по което се различават двата оператора е, че в средните скоби след задаване на типа на масива, се поставя запетая, която показва, че ще има два индекса и значи масивът е

- И тук е възможна декларация, при която се разполага масива и се задават стойности на елементите му. Всеки от редовете на матрицата се инициалилизира като едномерен масив – списък от стойностите, разделени със запетаи и се поставят в къдрави скоби.
- **Например**:
- int[,] matrix = { { 0, 2, 4, 0, 9, 5},
- { 7, 1, 3, 3, 2, 1},
- { 1, 3, 9, 8, 5, 6},
- { 4, 6, 7, 9, 1, 0} };
- създава двумерен масив с 4 реда и 6 стълба.

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

4. Обхождане

- За достъп до елемент на двумерен масив, двата индекса се поставят в средни скоби и са разделени със запетая.
- Например, операторът **m[2,3] = 12;** ще присвои стойност 12 на елемента на масива m, който се намира на 2 ред и 3 стълб (номерата на редовете и стълбовете започват от 0).
- **Въвеждането на стойностите на масив a** с n реда и m стълба от конзолата става с **два вложени цикъла, тъй като имаме два размера за обхождане**:
- for (int i=0; i<n; i++)
  - for (int j=0; j<m; j++)
    - a[i,j] = int.Parse(Console.ReadLine());

- **<u>Извеждане и обхождане на елементите на масив</u>** може да стане по различни начина. Например в редица по редове или в редица по стълбове. Най-естествено е обаче матрицата да се изведе в конзолата , като всички елементи от един ред се извеждат на един ред на конзолата, а различните редове на матрицата на различни редове на конзолата **– обхождане по редове**. За целта отново се използват два вложени цикъла:

```
for (int i=0; i<n; i++)
    {
     for (int j=0; j<m; j++) Console.WriteLine(a[i,j] + " ");
     Console.WriteLine();
     }
```

- **<u>Ако се налага да обхождаме масива по стълбове</u>**, т.е. Първо се извеждат всички елементи от първия стълб, после от втория и т.н., разменяме реда на операторите за цикъл, тъй като когато се обработват елементите от даден стълб, то индексът за номер на стълб не се променя:

```
for (int j=0; j<m; j++)
   {
        for (int i=0; i<n; i++)
        Console.WriteLine(a[i,j] + " "); Console.WriteLine();
   }
```

- **<u>Обхождане на диагонал на квадратен масив</u>** – не е необходимо да използваме два цикъла, тъй като, както вече видяхме, има връзка между индексите на елементите, които лежат по диагоналите. Ето как ще изглежда извеждането на елементите от главния диагонал на квадратен масив с n реда:

```
for (int i=0; i<n; i++)
 {
     Console.WriteLine(a[i,i] + " ");
 }
Console.WriteLine();
```

И извеждането на елементите от обратния диагонал:

```
for (int i=0; i<n; i++)
 {
     Console.WriteLine(a[i,n-1-i] + " ");
 }
Console.WriteLine();
```

- **<u>Обхождане на елементите, които са само под или само над главния или обратния диагонал на квадратен масив с n реда</u>**.
- Например, ако искаме да изведем елементите под главния диагонал, ще използваме свойството на индексите на всеки елемент в тази област, че номерът на стълба му винаги е по-малък от номера на реда му, т.е. Вторият му индекс е по-малък от първия. Затова ще се използват два цикъла, но вътрешния цикъл ще зависи от стойността на управляващата променлива на първия цикъл:

```
for (int i=1; i<n; i++)
 {
    for (int j=0; j<i; j++)
    Console.WriteLine(a[i,j] + " ");
    Console.WriteLine();
```

- Ето и **<u>извеждането на елементите, които са над обратния диагонал</u>** – при тях индексът на стълба отново зависи от индекса на реда като сборът им винаги е по-малък от **n-1**:

```
for (int i=1; i<n; i++)
 {
    for (int j=0; j<n-1-i; j++)
    Console.WriteLine(a[i,j] + " ");
    Console.WriteLine();
 }
```

5. Обработка на таблични данни.

- **<u>Задача1</u>**: Напишете конзолно приложение, което въвежда в двумерен масив оценките по m учебни предмети на n ученика, номерирани от 1 до n, по реда на въвеждане, и след като извърши необходимите пресмятания, извежда на конзолата номера на ученика с най-висок среден успех и номера на предмета с най-нисък среден успех. Ако има няколко такива ученици, програмата да изведе номера на този ученик, който е най-малък.

Ако вземем примера в който са дадени данни за 3 ученика и 4 предмета:

	БЕЛ	Чужд език	Матем атика	Инфор матика	ИТ
Алекса ндър	6	4	5	6	6
Диана	5	6	4	6	6
Мартин	6	6	4	5	5

тогава програмата трябва да изведе номера на Диана (2), която има най-висок среден успех и номера на предмета Математика (3), при който средния успех е най-нисък.

- Средният успех се пресмята, като се съберат всички оценки и се раздели на броят им. Тъй като в тази задача броят на предметите е еднакъв за всеки ученик, както и броят на учениците е един и същи за всеки предмет, то за да намерим най-висок или най-нисък успех не е необходимо да извършваме делението, което ще ни спести работата с дробен тип данни.
- За пресмятането на средния успех на всеки ученик ще трябва да съберем оценките му, което означава, че трябва да извършим обхождане ред по ред и преди сменянето на ред трябва да занулираме променливата, в която пресмятаме сбора от оценките на всеки ученик, а след изчислението на този сбор трябва да проверим дали той не е текущ максимум, и ако е така да съхраним номера на ученика с този сбор.

- Аналогично стоят нещата и за намиране на най-нисък успех по предмет. В този случай трябва да се извърши обхождане по стълбове и да се търси текущ минимум.
- За реализацията на задачата ще се използват няколко обхождания на масива – едно за въвеждане на елементите му (по редове), едно за намиране на най-висок успех на ученик (по редове) и едно за намиране на най-ниския успех на учебен предмет (по стълбове):

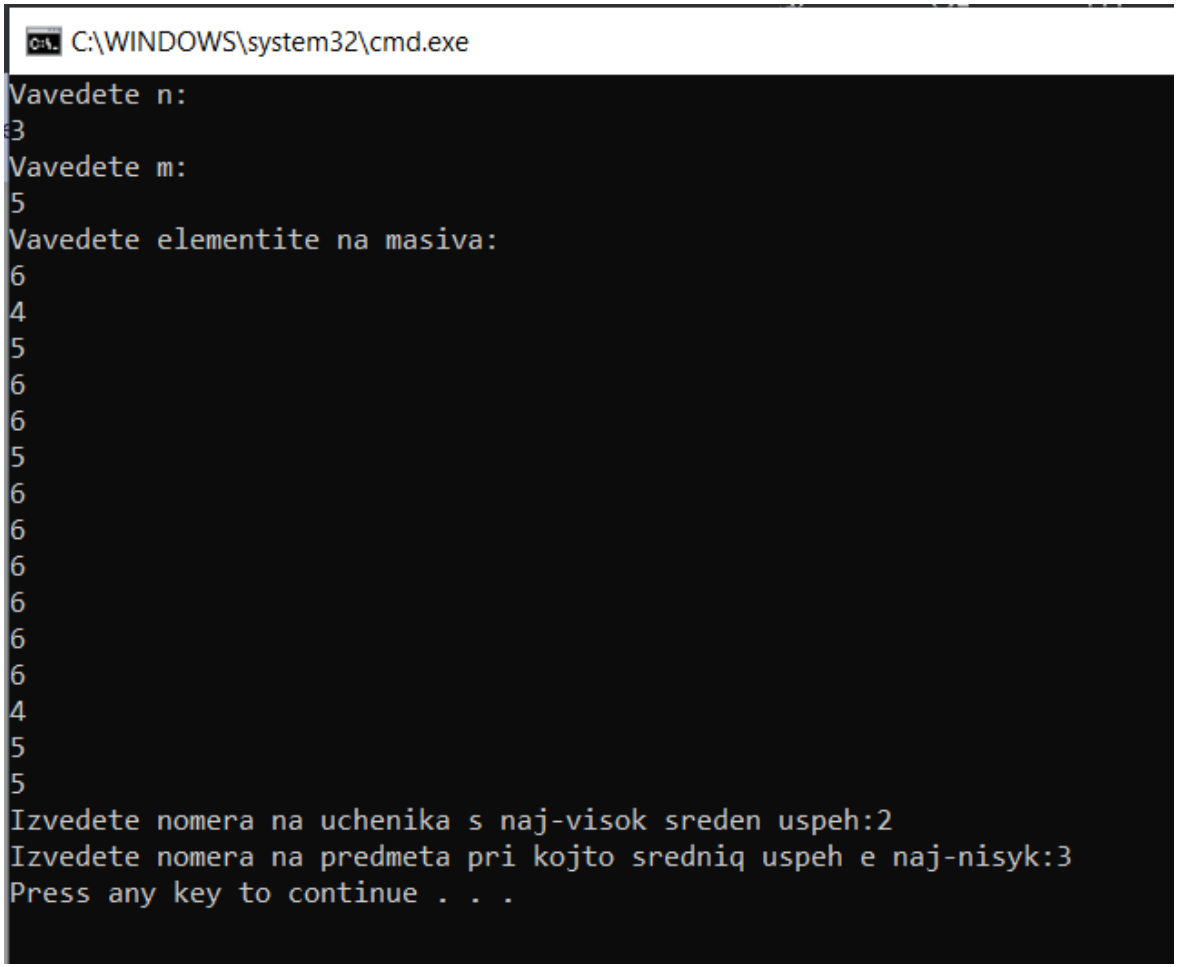
Код:

```
static void Main(string[] args)
    {
        Console.WriteLine("Vavedete n: ");
        int n = int.Parse(Console.ReadLine());
        Console.WriteLine("Vavedete m: ");
        int m = int.Parse(Console.ReadLine());
        int[,] students = new int[n, m];
        // Въвеждане на входни данни
        Console.WriteLine("Vavedete elementite na masiva: ");
        for (int i=0; i<n; i++)
            for (int j=0; j<m; j++)
            {
                students[i, j] = int.Parse(Console.ReadLine());
            }
```

```
//Намиране на номер на ученик с най-висок успех
        int max = 0, numStuden = 0;
        for (int i = 0; i<n; i++)
        {
            int sum = 0;
            for (int j = 0; j<m; j++)
            {
                sum += students[i, j];
            }
            if (sum > max)
            {
                max = sum;
                numStuden = i;
            }
        }
```

```
//Намиране на номер на предмет с най-нисък успех
        int min = 1000, numSubject = 0;
        for (int j=0; j<m; j++)
        {
           int sum = 0;
           for (int i=0; i<n; i++)
           {
              sum += students[i, j];
           }
           if (sum < min)
           {
              min = sum;
              numSubject = j;
           }
        }
        //Извеждане на номерата, но с едно повече, защото започваме от 1
        Console.WriteLine("Izvedete nomera na uchenika s naj-visok sreden uspeh:" +
(numStuden + 1));
        Console.WriteLine("Izvedete nomera na predmeta pri kojto sredniq uspeh e naj-nisyk:"
+ (numSubject + 1));
    }
  }
```

Резултат:


C:\WINDOWS\system32\cmd.exe
Vavedete n:
3
Vavedete m:
5
Vavedete elementite na masiva:
6
4
5
6
6
5
6
6
6
6
6
6
4
5
5
Izvedete nomera na uchenika s naj-visok sreden uspeh:2
Izvedete nomera na predmeta pri kojto srednig uspeh e naj-nisyk:3
Press any key to continue . . .

Графична компонента dataGridView

- Графичната компонента **dataGridView** наподобява двумерен масив.
- Тя представя не екрана двумерна таблица от клетки (екземпляри на класа **dataGridViewCell**), всяка от които може да съдържа произволна стойност (object), за разлика от двумерния масив, в който всички стойности са от един и същи тип. Мястото на клетката в таблицата се определя от номера на стълба и реда, в които се намира. Забележете още една разлика с двумерния масив, където при идентифицирането на елементите пурво задаваме реда, а после стълба (при двумерния масив).
- Така например клетка в ред I и стълб j на компонентата grid1 от типа dataGridView указваме

- Съдържанието на клетката се задава в свойствотоValue.
- Така например с оператора
- grid1[j, i].Value = „Информатика“;
- Задаваме нова стойност на клетката в стъл j и ред i, а с оператора:
- string s = (string) grid1[j, i].Value;
- или
- string s = grid1[j, i].Value.ToString();
- присвояваме на променлива от тип string стойността на клетката в стълб j и ред i.

- Вътрешно таблицата е организирана като колекция от колони(свойството **Columns**). За да редактираме колоните на колекцията трябва да отворим диалоговия прозорец **Edit Columns**. Със стрелките нагоре и надолу може да се промени позицията на маркираната колона.
- Изтриване на колона става с щракване върху бутона **Remove**, а нова колекция се добавя с щракване върху бутона **Add...**, при което се отваря диалоговия прозорец **Add Column**. В полето **Header text** се изписва наименованието на новата колона.