Массивы в C

ПОДГОТОВИЛ: БАБОШКИНА Т.В.

Массив

- Массив - это упорядоченный набор фиксированного количества некоторых значений (компонент массива). Все компоненты должны быть одного и того же типа, который называют типом компонент или базовым (для массива) типом.
- Над элементами массива можно выполнять любые действия допустимые для переменной базового типа массива, а также использовать элементы массива в качестве аргументов процедур и функций.
- При описании массива указывается диапазон номеров элементов массива и тип, к которому относится каждый его элемент. Массивы могут быть одно-, двух- и многомерными.

- Особенности массива в C# определены тем что массив в этом языке является классом Array, который относится к библиотеке FCL
- В языке C# все массивы изначально являются динамическими структурами, что позволяет менять их размер в ходе работы программы, удалять и создавать по мере надобности и другие особенности динамических конструкций.

Особенности реализации массива в C#

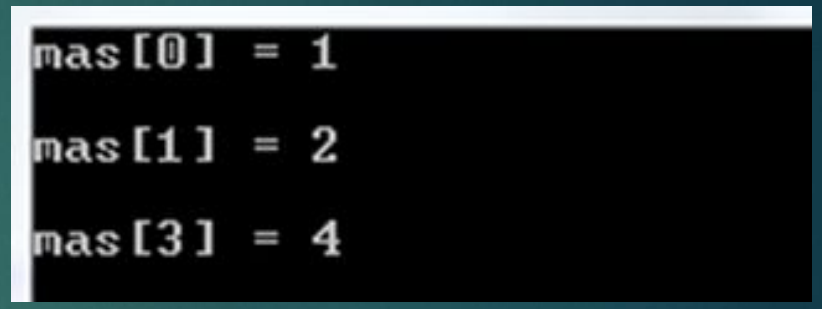
В самом простом варианте – одномерный массив с инициализацией при описании:

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

обращение к массиву

```
int [] mas = {1,2,3,4,5};
int i = 1, j = 2;
Console.WriteLine("mas[0] = {0}", mas[0]);// Используется в качестве индекса константу
Console.WriteLine("mas[{0}] = {1}",i, mas[i]); //Используется переменную
Console.WriteLine("mas[{0}] = {1}",i+j, mas[i+j]);//Используется выражение
```

Результат работы программ



Особенности реализации массива в C#

- Если же возникает попытка обратится к элементу массива, указав индекс за пределами выделенной памяти – возникнет

```
lass Program

    static void Main(string[] args)
    {
        int [] mas = {1,2,3,4,5};
        int i = 1, j = 2;
        Console.WriteLine("mas[0] = {0}\n", mas[7]);
        Console.WriteLine("mas[{0}] = {1}\n",i, mas[i]);
        Console.WriteLine("mas[{0}] = {1}\n",i+j, mas
        Console.ReadLine();

    }
```

IndexOutOfRangeException не обработано

Индекс находился вне границ массива.

Советы по устранению неполадок:

Убедитесь в правильности имен столбцов данных

Конструкция: try...catch...finally

Блок try..catch.

```
try
{
  //блок кода, в котором возможно исключение
}
catch ([тип исключения] [имя])
{
  //блок кода – обработка исключения
}
```

В конкретном случае этот блок может иметь следующий вид:

```
try
  {
    Console.WriteLine("mas[{0}] = {1}",i, mas[i]);
  }

catch(System.IndexOutOfRangeException e)
  {
    throw new System.ArgumentOutOfRangeException(
        "Parameter index is out of range.");
  }
```

Классическая структура описания массива в C# имеет вид:

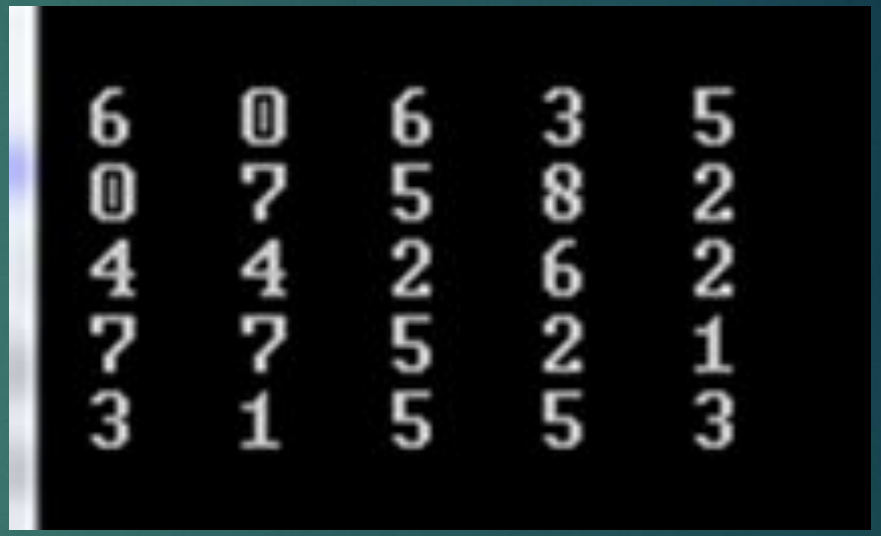
тип[] имя_массива = new тип[размер массива];

- Тогда ранее рассмотренный пример должен иметь вид:

int [] mas = new int[5] {1,2,3,4,5};

Описание двухмерного массива, заполнение его случайными числами и вывод в виде таблицы

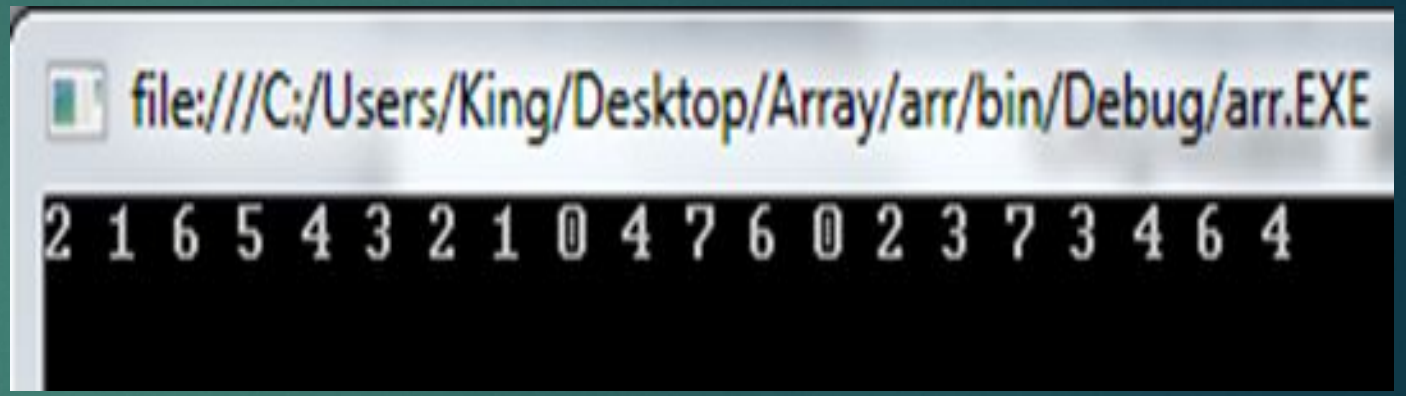
```
int[,] mas2 = new int[5, 5];
        Random rnd = new Random();
        for (int ii = 0; ii < 5; ii++)
        {
            for (int jj = 0; jj < 5; jj++)
            {
                mas2[ii, jj] = rnd.Next(9);
                Console.Write(" {0} ",
mas2[ii,jj]);
            }
            Console.WriteLine();
        }
```



Ссылки на класс Array:

```
Array mas = new int[20];
        Random rnd = new Random();
        for (int i = 0; i < 20; i++)
        {

mas.SetValue(rnd.Next(10), i);
            Console.Write("{0} ", mas.GetValue(i));
        }
        Console.WriteLine();
        Console.ReadLine();
```



Методы обращения (геттеры/сеттеры):

- IndexOf (array, object) – найти первое вхождение элемента
- LastIndexOf (array, object) – найти последнее вхождение элемента
- Sort (array) – отсортировать массив.

Цикл перебора коллекции для массива:

```
int ind1 = Array.IndexOf(mas, 2);
        int ind2 = Array.LastIndexOf(mas, 2);
        Console.Write("Первое вхождение числа 2: mas[{0}]=2\n", ind1);
        Console.Write("Последнее вхождение числа 2: mas[{0}]=2\n", ind2);
        Array.Sort(mas);
        Console.Write("Отсортированный массив:\n");
        foreach (int el in mas)
        {
            Console.Write("{0} ", el);
        }
        Console.WriteLine();
        Console.ReadLine();
```

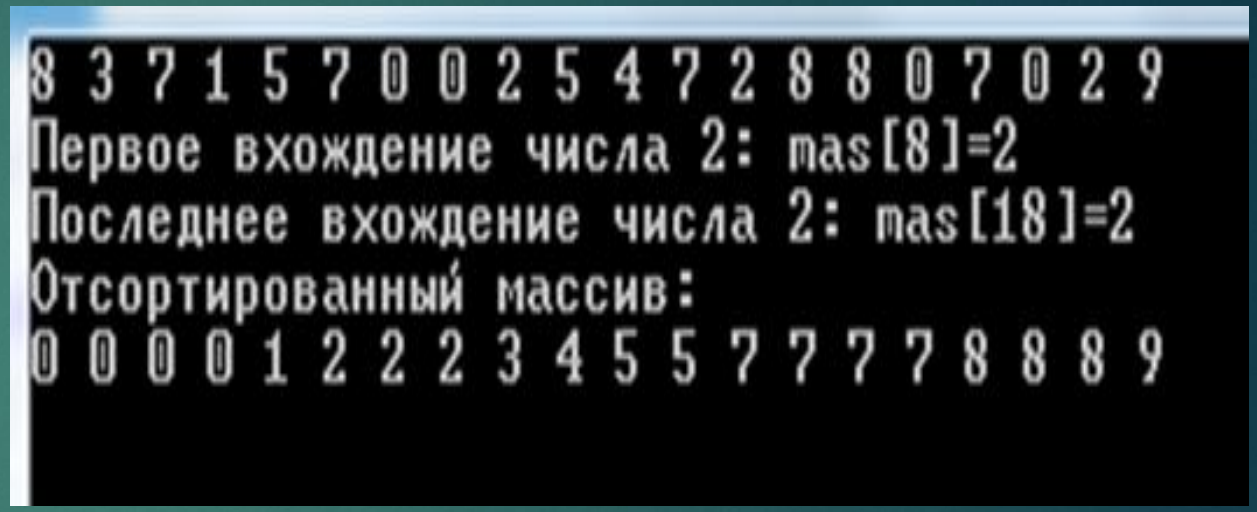
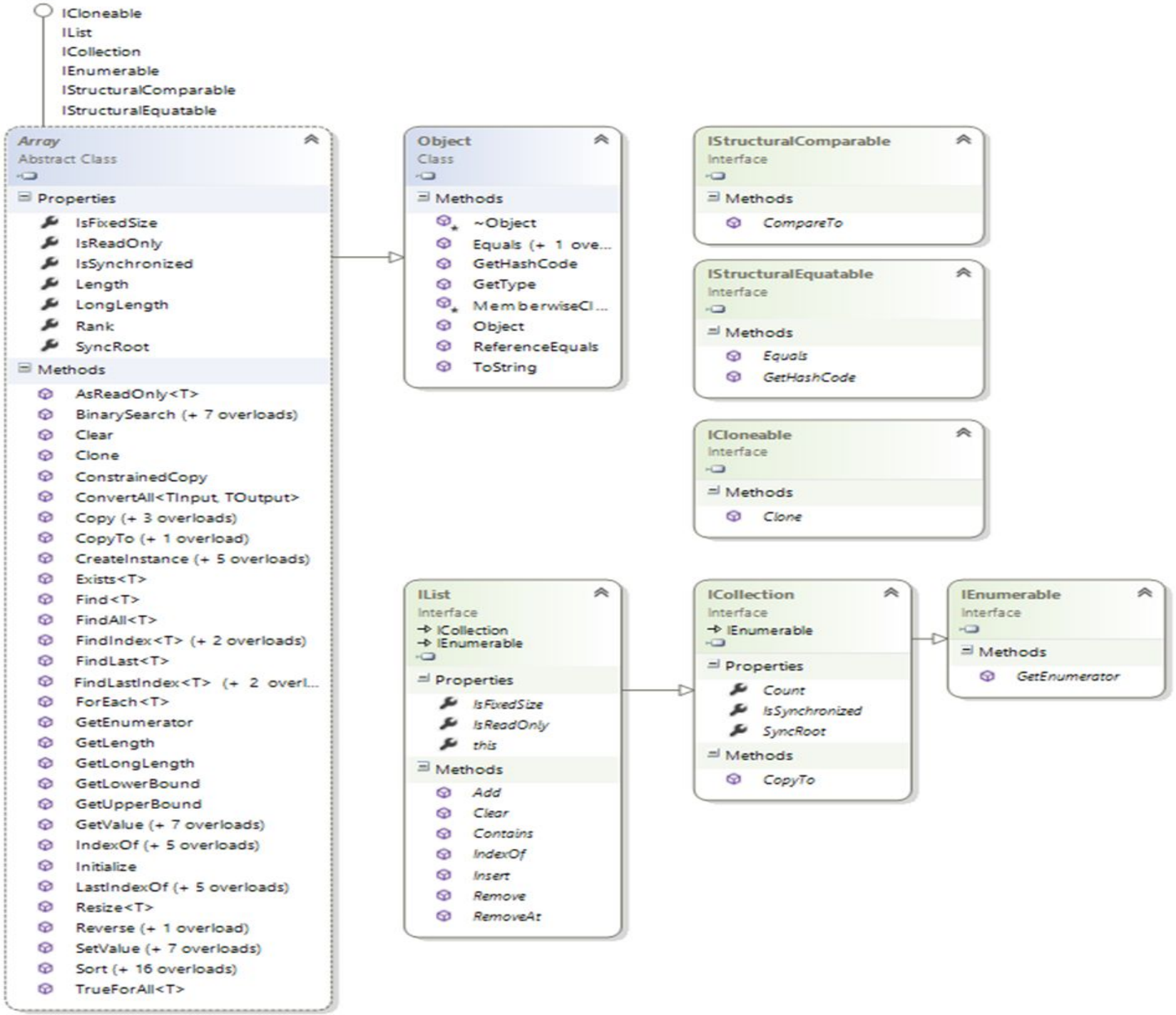


Диаграмма интерфейсов класса System.Array



Метод возвращает значения:

- больше нуля – текущий объект размещается после объекта переданного как параметр (текущий больше);
- равное нулю – текущий объект равен объекту переданного как параметр, объекты размещаются в порядке следования;
- меньше нуля – текущий объект размещается перед объектом переданным как параметр (текущий меньше).

наглядно список интерфейсов класса можно получить, написав следующий код:

```
// Использование обобщенного интерфейса IComparable

    class DataBase : IComparable<object>

    {

        public string login, password;

        public int id;

        byte sort; // Определяет по какому из полей сортировать

                // 1 - сортировка по id

                // 2 - сортировка по логину

               //При этом по id в прямом порядке, а по логину в обратном


        public DataBase() {}


        public DataBase(string login, string password, int id)

        {

            this.login = login;

            this.password = password;

            this.id = id;

        }
```

```
// Реализация интерфейса IComparable<T>

        public int CompareTo(object other)

        {

            // Проверим тип объекта obj

            DataBase db = other as DataBase;

            if (db != null)

            {

                switch (db.sort)

                {

                    case 1:

                        {

                            if (this.id > db.id)

                                return 1;

                            if (this.id < db.id)

                                return -1;

                            else

                                return 0;

                        }
```

```
                // Метод, реализующий сортировку

                public void Sort(ref DataBase[] db_object)

                {

                    try

                    {

                        Array.Sort(db_object);

                    }

                    catch (Exception ex)

                    {

                        Console.WriteLine(ex.Message);

                    }

                    foreach (DataBase d in db_object)

                        Console.WriteLine("{0}\t{1}\t{2}",d.id,d.login,d.password);

                }

                public void NumberSort(byte j, ref DataBase[] arr)

                {

                    for (int i = 0; i < arr.Length; i++)

                        arr[i].sort = j;

                }

            }
```

Далее, организовав массив объектов описанного выше класса, можно реализовать сортировку по определённым ранее правилам. Вызов метода может иметь вид:

```
DataBase[] db_arr = new DataBase[5];
        db_arr[0] = new DataBase("alex85", "sddd", 14);
        db_arr[1] = new DataBase("dm23", "12345", 2);
        db_arr[2] = new DataBase("rvklops", "ss", 5);
        db_arr[3] = new DataBase("Djeff", "sdsdsdf", 86);
        db_arr[4] = new DataBase("dff","s",15);

        DataBase sr1 = new DataBase();
        sr1.NumberSort(1, ref db_arr);//Определяем параметр сортировки по id
        sr1.Sort(ref db_arr);// вызываем метод сортировки
```

Язык C++

Дана целочисленная матрица размером n*m. Написать программу, позволяющую находить сумму наибольших значений элементов ее строк

Фрагмент решения задачи:

```
for (i=0;i<n;i++)
   {  max = -101;
      for (j=0;j<m;j++)
      {
        mas[i][j] = rand()%201 - 100;// случайное число из диапазона -100:100
        printf("%3d ",mas[i][j]);
        if (mas[i][j]>max)
         {max = mas[i][j];}
        }
        Sum+=max;
          cout<<"maxi = "<<max<<"\n";
     }
```

Задана целочисленная матрица размером n*m. Написать программу, позволяющую находить строки с наименьшей и наибольшей суммой и выводить их на печать

Первый массив

```
//Сама матрица
    int mas[n][m];
    //Массив для хранения сумм
    int Sum[n]={0};
```

Второй (линейный массив)

```
for (i=0;i<n;i++)
    {  for (j=0;j<m;j++)
       {
         mas[i][j] = rand()%201 - 100;// случайное число из диапазона -100:100
         printf("%3d ",mas[i][j]);
        Sum[i]+=mas[i][j];
         }
        cout<<"Sum_el = "<<Sum[i]<<"\n";
      }
```

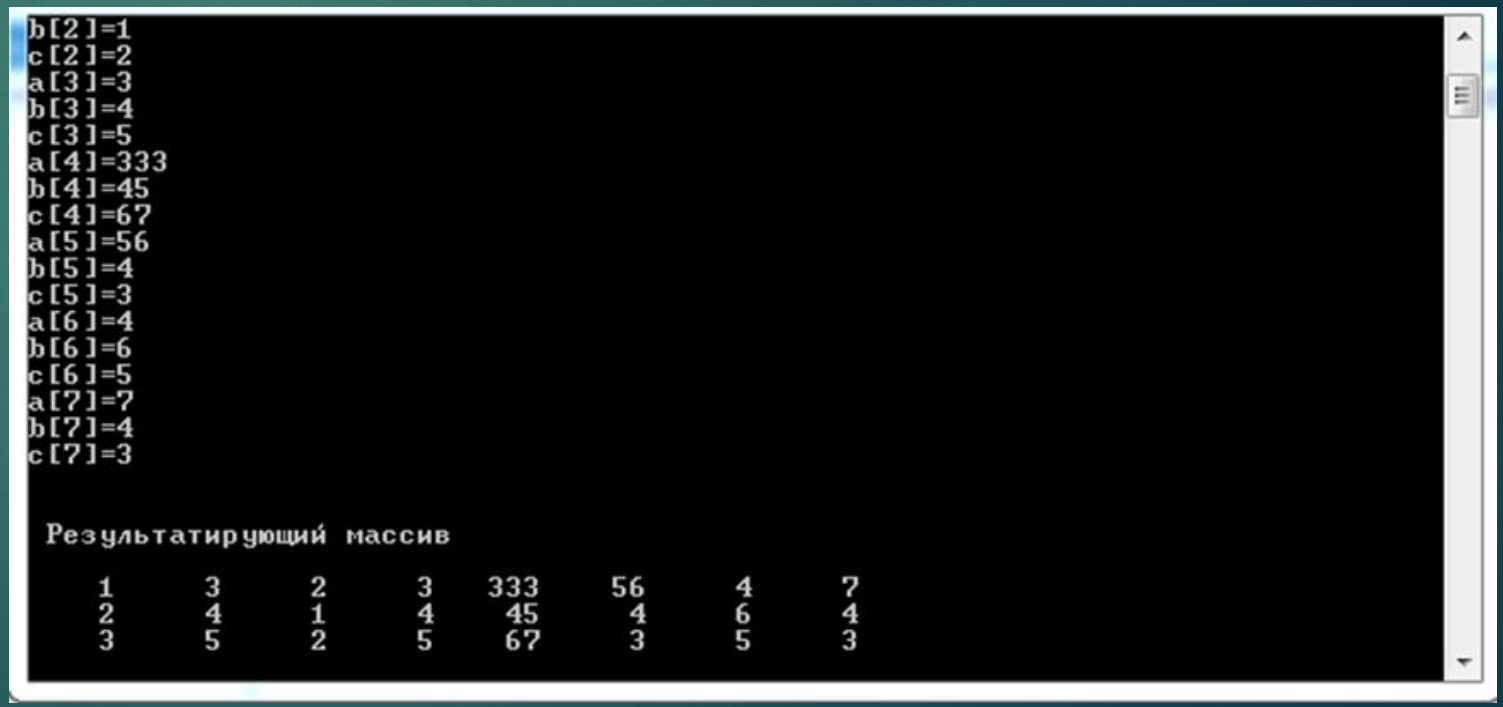
найдем максимум в линейном массиве

```
max = Sum[0];
    imax = 0;
    min = Sum[0];
    imin =0;
    for (i=1;i<n;i++)
    {
       if (Sum[i]>max)
       { max = Sum[i];
         imax = i;}
```

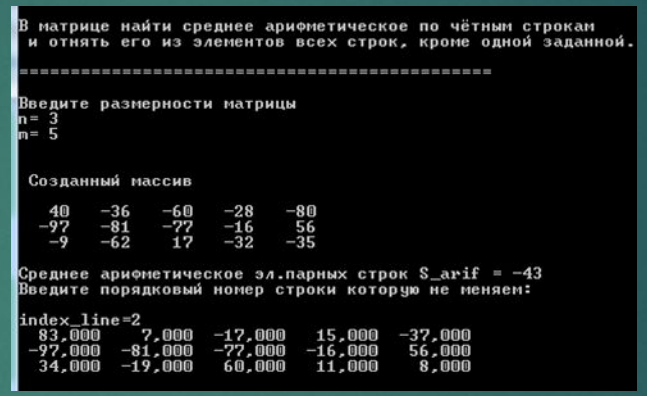
Задача 1. Даны вещественные массивы A[8], B[8], C[8]. Определить значения вещественного массива D[3][8]. В первую строку массива D записать значения массива A, во вторую - массива B, а в 3-ю - массива C.

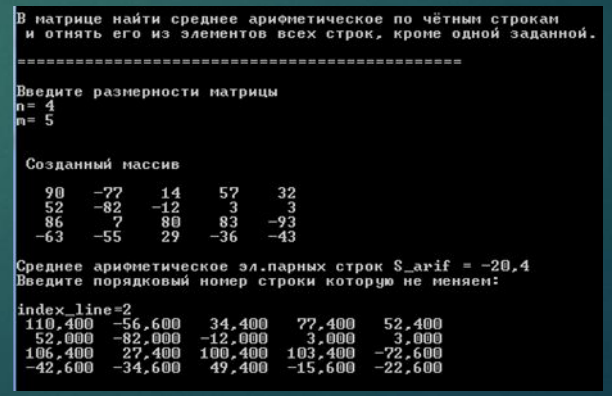
.

- double[] a = new double[8];
- double[] b = new double[8];
- double[] c = new double[8];
- double[,] d = new double [3,8];



- В матрице найти среднее арифметическое по чётным строкам и отнять его из элементов всех строк, кроме одной заданной.

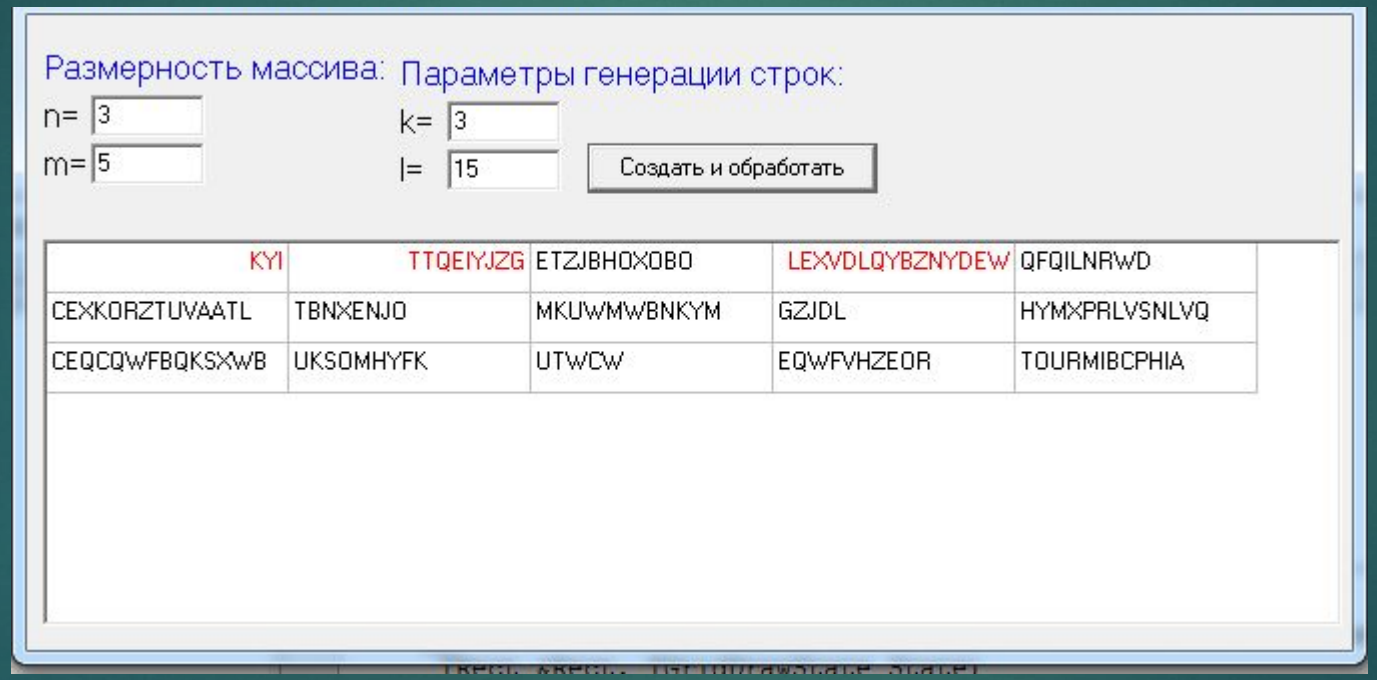




Язык C++Builder WinForms

- Сформировать двухмерный MxN массив из элементов. В качестве элементов использовать слова из K..L символов A..Z. Осуществить действия по заданному алгоритму. Вывести на экран сформированный массив, выделив цветом или мерцанием элементы – строки, имеющие среднюю гласную букву, и результат подсчета этих элементов.
- **WinForms** приложения отличаются тем, что интерфейс строится на базе оконного интерфейса ОС Windows, но обработка данных, в том числе и массивов, при этом абсолютно не отличается.

Результаты работы программы и тестовый запуск



ВЫВОДЫ

В данной курсовой работе был рассмотрен тип данных массив и особенности его реализации в различных языках. Основное внимание уделено реализации массива в языке C#.

Как показывает сравнение – во всех рассмотренных языках есть много общего при работе с массивами. В языке C# можно работать с массивами как в более ранних языках, где массив рассматривается как структура данных. Если упустить синтаксические отличия, то особой разницы при работе с массивом как с типом данных (как со структурой представления данных в памяти) в различных языках программирования нет. Но в C# массив является экземпляром класса, и если работать с массивом в C# как с объектом (экземпляром класса) то возникают особенности на всех уровнях:

- описание массива;

- выделение памяти под массив (в языках C и Паскаль для стандартного типа – статический массив достаточно только описания);

- обращение к элементам массива.

Но самым главным отличием массива в C# (за счет объектного) есть наличие множества готовых методов и интерфейсов, которые уже реализованы или наследованы классом Array. Этот факт делает массивы C# мощным и удобным инструментов для оперирования данными в ходе работы программы.

СПАСИБО ЗА ВНИМАНИЕ