# Лабораторная работа №6

Студента группы ИТ-21

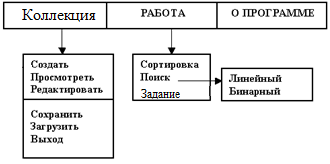
Гура Ильи Сергеевича

Выполнение \_\_\_\_\_\_\_\_\_ Защита\_\_\_\_\_\_\_\_\_

# Массивы и коллекции C#

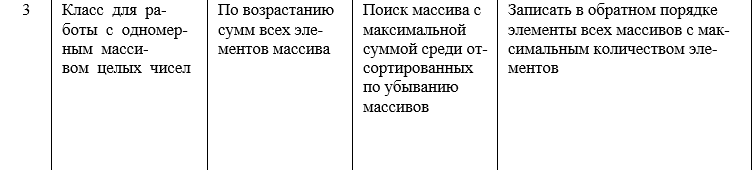
***Цель работы*:** знакомство с интерфейсами и классами коллекций C#. Получение навыков при работе с классом коллекции ArrayList. Освоение приемов организации работы в Windows Form приложении через меню.

Содержание работа:



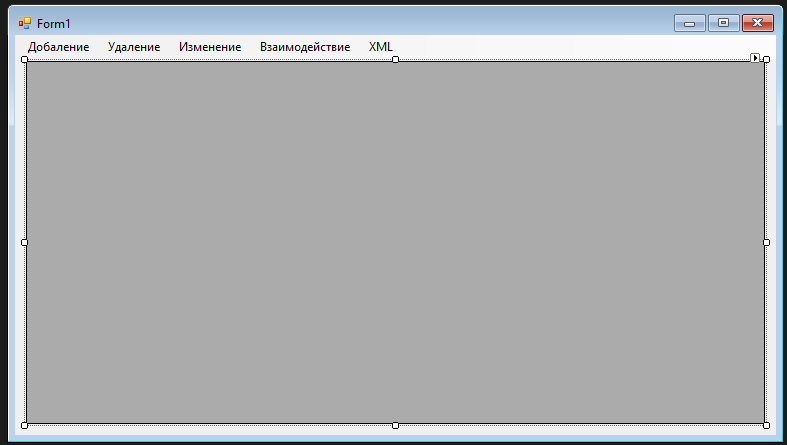
Реализовать все указанные в меню действия с коллекцией. Для манипуляции содержимым коллекции обязательно использовать двустороннюю связь с *DataGridView*. Для реализации выполнения команд: Создать, Просмотреть и Редактировать использовать одну форму, динамически изменяя свойства элементов управления.

Команды пункта меню Работа должны реализовывать способ сортировки коллекции, поиск элемента и выполнение индивидуального задания в соответствии с вариантом задания (табл. ).



# Ход работы

1. Форма



1. Основная программа

using System;

using System.Xml.Serialization;

using System.IO;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

using System.Collections;

namespace VP\_6

{

public partial class Form1 : Form

{

private const string ERR = "Ошибка";

private const string ERR\_PARS = "Введены некорректные данные";

private const string ERR\_FIND = "Необходимый объект не был найден";

private const string ERR\_LENGTH = "Необходимо соответствие длинны";

private const string ERR\_SELECTED = "Необходимо выбрать хоть одну строку";

private const string XML\_FILE\_NAME = "File.xml";

private const int maxRandom = 10;

private bool t = false;

private int Length = 5;

private ArrayList arrayOfArray = new ArrayList();

private Array AutoGeneration()

{

Array autoArray = new Array((uint)Length);

Random random = new Random();

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

{

autoArray.IntArray[i] = random.Next(10);

}

return autoArray;

}

private int MaxN()

{

int maxN = 0;

for (int i = 0; i < arrayOfArray.Count; i++)

{

Array a = arrayOfArray[i] as Array;

if (a.N > maxN)

{

maxN = (int)a.N;

}

}

return maxN;

}

/\*private void Setting()

{

Length = MaxN();

for (int i = 0; i < arrayOfArray.Count; i++)

{

(arrayOfArray[i] as Array).N = (uint)Length;

}

}\*/

private void DataGridViewOutput()

{

//Setting();

if (MaxN() == 0)

{

dataGridView1.Rows.Clear();

}

else

{

dataGridView1.ColumnCount = MaxN();

dataGridView1.RowCount = arrayOfArray.Count;

for (int i = 0; i < arrayOfArray.Count; i++)

{

int j;

Array a = arrayOfArray[i] as Array;

for (j = 0; j < a.N; j++)

{

dataGridView1.Rows[i].Cells[j].Value = a.IntArray[j];

}

}

}

}

public Form1()

{

InitializeComponent();

arrayOfArray.Add(AutoGeneration());

dataGridView1.ReadOnly = true;

DataGridViewOutput();

}

private void сортирокаToolStripMenuItem\_Click(object sender, EventArgs e)

{

arrayOfArray.Sort();

DataGridViewOutput();

}

private void удалениеToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<int> indexToDelete = dataGridView1.SelectedRows.Cast<DataGridViewRow>().Select(x => x.Index).ToList();

if (indexToDelete.Count != 0)

{

indexToDelete.Sort();

arrayOfArray.RemoveAt(indexToDelete[0]);

indexToDelete.RemoveAt(0);

DataGridViewOutput();

}

else

{

MessageBox.Show(ERR\_SELECTED, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private int[] StringToArray(string array)

{

if (array != "")

{

string[] arrayOfString = array.Split(' ');

int[] arrayOfInt = new int[arrayOfString.Length];

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

{

if (Int32.TryParse(arrayOfString[i], out int newInt))

{

arrayOfInt[i] = newInt;

}

else

{

return null;

}

}

return arrayOfInt;

}

else

{

return null;

}

}

private void самостоятельноеToolStripMenuItem\_Click(object sender, EventArgs e)

{

NewArray arrayForm = new NewArray();

DialogResult result = arrayForm.ShowDialog(this);

if (result == DialogResult.Cancel)

{

return;

}

int[] intArray = StringToArray(arrayForm.textBox1.Text);

if (intArray != null)

{

arrayOfArray.Add(new Array(intArray));

DataGridViewOutput();

}

else

{

MessageBox.Show(ERR\_PARS, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private bool TryToParse(out ArrayList newAArrayList)

{

newAArrayList = new ArrayList();

for (int i = 0; i < dataGridView1.RowCount; i++)

{

Array newArray = new Array((uint)dataGridView1.ColumnCount);

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

if (dataGridView1.Rows[i].Cells[j].Value == null)

{

newArray.IntArray[j] = 0;

}

else

{

if (Int32.TryParse(dataGridView1.Rows[i].Cells[j].Value.ToString(), out int newInt))

{

newArray.IntArray[j] = newInt;

}

else

{

return false;

}

}

}

newAArrayList.Add(newArray);

}

return true;

}

private void изменениеToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (dataGridView1.ReadOnly)

{

dataGridView1.ReadOnly = false;

изменениеToolStripMenuItem.Text = "Сохранить изменения";

добалениеToolStripMenuItem.Enabled = false;

удалениеToolStripMenuItem.Enabled = false;

взаимодействейToolStripMenuItem.Enabled = false;

xMLToolStripMenuItem.Enabled = false;

}

else

{

if (TryToParse(out ArrayList newAArrayList))

{

dataGridView1.ReadOnly = true;

arrayOfArray = newAArrayList;

DataGridViewOutput();

изменениеToolStripMenuItem.Text = "Изменение";

добалениеToolStripMenuItem.Enabled = true;

удалениеToolStripMenuItem.Enabled = true;

взаимодействейToolStripMenuItem.Enabled = true;

xMLToolStripMenuItem.Enabled = true;

}

else

{

MessageBox.Show(ERR\_PARS, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void инкрименнтМассиваToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<int> indexToInc = dataGridView1.SelectedRows.Cast<DataGridViewRow>().Select(x => x.Index).ToList();

if (indexToInc.Count != 0)

{

indexToInc.Sort();

while (indexToInc.Count != 0)

{

Array a = arrayOfArray[indexToInc[0]] as Array;

a++;

indexToInc.RemoveAt(0);

}

DataGridViewOutput();

}

else

{

MessageBox.Show(ERR\_SELECTED, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void дикриментToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<int> indexToInc = dataGridView1.SelectedRows.Cast<DataGridViewRow>().Select(x => x.Index).ToList();

if (indexToInc.Count != 0)

{

indexToInc.Sort();

while (indexToInc.Count != 0)

{

Array a = arrayOfArray[indexToInc[0]] as Array;

a--;

indexToInc.RemoveAt(0);

}

DataGridViewOutput();

}

else

{

MessageBox.Show(ERR\_SELECTED, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void массивОтсортированToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<int> indexToInc = dataGridView1.SelectedRows.Cast<DataGridViewRow>().Select(x => x.Index).ToList();

if (indexToInc.Count != 0)

{

indexToInc.Sort();

while (indexToInc.Count != 0)

{

Array a = arrayOfArray[indexToInc[0]] as Array;

if (!a)

{

MessageBox.Show("Массив отсортирован по убыванию");

t = true;

}

else

{

MessageBox.Show("Массив не отсортирован по убыванию");

}

indexToInc.RemoveAt(0);

}

DataGridViewOutput();

}

else

{

MessageBox.Show(ERR\_SELECTED, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void автоматическоеToolStripMenuItem\_Click(object sender, EventArgs e)

{

arrayOfArray.Add(AutoGeneration());

DataGridViewOutput();

}

private void линейныйToolStripMenuItem\_Click(object sender, EventArgs e)

{

NewArray arrayForm = new NewArray();

DialogResult result = arrayForm.ShowDialog(this);

if (result == DialogResult.Cancel)

{

return;

}

int[] intArray = StringToArray(arrayForm.textBox1.Text);

if (intArray != null)

{

if (intArray.Length == Length)

{

Array value = new Array(intArray);

int index = -1;

for (int i = 0; i < arrayOfArray.Count; i++)

{

Array array = arrayOfArray[i] as Array;

if (array == value)

{

index = i;

break;

}

}

if (index == -1)

{

MessageBox.Show(ERR\_FIND, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

dataGridView1.ClearSelection();

dataGridView1.Rows[index].Selected = true;

}

}

else

{

MessageBox.Show(ERR\_LENGTH, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show(ERR\_PARS, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void массиваСMaxToolStripMenuItem\_Click(object sender, EventArgs e)

{

int max = -1;

int index = -1;

for (int i = 0; i < arrayOfArray.Count; i++)

{

Array array = arrayOfArray[i] as Array;

int b = array.Summ();

if (!array && b>max) {

max = b;

index = i;

}

}

if (index == -1)

{

MessageBox.Show(ERR\_FIND, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

dataGridView1.ClearSelection();

dataGridView1.Rows[index].Selected = true;

}

}

private void обратныйПорядокToolStripMenuItem\_Click(object sender, EventArgs e)

{

uint max = 0;

for (int i = 0; i < arrayOfArray.Count; i++)

{

Array array = arrayOfArray[i] as Array;

if (array.N > max)

{

max = array.N;

}

}

for (int i = 0; i < arrayOfArray.Count; i++)

{

Array array = arrayOfArray[i] as Array;

Array arr = new Array(array.N);

if (array.N == max)

{

for(uint j = array.N ; j > 0; j--)

{

arr.IntArray[array.N - j] = array.IntArray[j-1];

}

arrayOfArray[i] = arr;

}

}

DataGridViewOutput();

}

private void бинарныйToolStripMenuItem\_Click(object sender, EventArgs e)

{

arrayOfArray.Sort();

DataGridViewOutput();

NewArray arrayForm = new NewArray();

DialogResult result = arrayForm.ShowDialog(this);

if (result == DialogResult.Cancel)

{

return;

}

int[] intArray = StringToArray(arrayForm.textBox1.Text);

int index = arrayOfArray.BinarySearch(new Array(intArray));

if (index == -1)

{

MessageBox.Show(ERR\_FIND, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

dataGridView1.ClearSelection();

dataGridView1.Rows[index].Selected = true;

}

}

private void сортировкаМассиваToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<int> indexToSort = dataGridView1.SelectedRows.Cast<DataGridViewRow>().Select(x => x.Index).ToList();

if (indexToSort.Count != 0)

{

indexToSort.Sort();

while (indexToSort.Count != 0)

{

Array a = arrayOfArray[indexToSort[0]] as Array;

a.BubbleSort();

indexToSort.RemoveAt(0);

}

DataGridViewOutput();

}

else

{

MessageBox.Show(ERR\_SELECTED, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void сОртСПИСToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<int> indexToSort = dataGridView1.SelectedRows.Cast<DataGridViewRow>().Select(x => x.Index).ToList();

if (indexToSort.Count != 0)

{

indexToSort.Sort();

int temp = 0;

int i = 0;

while (indexToSort.Count != 0)

{

Array a = arrayOfArray[indexToSort[0]] as Array;

a.sum = a.IntArray[0];

int b = a.Summ();

a.IntArray[0] = b;

indexToSort.RemoveAt(0);

//a.IntArray[0] = temp;

}

}

arrayOfArray.Sort();

List<int> indexToFirst = dataGridView1.SelectedRows.Cast<DataGridViewRow>().Select(x => x.Index).ToList();

if (indexToFirst.Count != 0) {

while (indexToFirst.Count != 0)

{

Array a = arrayOfArray[indexToFirst[0]] as Array;

a.IntArray[0] = a.sum;

indexToFirst.RemoveAt(0);

}

DataGridViewOutput();

}

else

{

MessageBox.Show(ERR\_SELECTED, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void сортToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<int> indexToSort = dataGridView1.SelectedRows.Cast<DataGridViewRow>().Select(x => x.Index).ToList();

if (indexToSort.Count != 0)

{

indexToSort.Sort();

while (indexToSort.Count != 0)

{

Array a = arrayOfArray[indexToSort[0]] as Array;

a.BubbleSortDescending();

indexToSort.RemoveAt(0);

}

DataGridViewOutput();

}

else

{

MessageBox.Show(ERR\_SELECTED, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void сохранитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

Type[] Types = new Type[1];

Types[0] = typeof(Array);

XmlSerializer serializer = new XmlSerializer(typeof(ArrayList), Types);

using (FileStream XMLFile = new FileStream(XML\_FILE\_NAME, FileMode.OpenOrCreate))

{

serializer.Serialize(XMLFile, arrayOfArray);

}

}

private void загрузитьToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

Type[] Types = new Type[1];

Types[0] = typeof(Array);

XmlSerializer deserializer = new XmlSerializer(typeof(ArrayList), Types);

using (FileStream XMLFile = new FileStream(XML\_FILE\_NAME, FileMode.OpenOrCreate))

{

arrayOfArray = (ArrayList)deserializer.Deserialize(XMLFile);

DataGridViewOutput();

}

}

private void массивОтсортированПоУбываниюToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<int> indexToInc = dataGridView1.SelectedRows.Cast<DataGridViewRow>().Select(x => x.Index).ToList();

if (indexToInc.Count != 0)

{

indexToInc.Sort();

while (indexToInc.Count != 0)

{

Array a = arrayOfArray[indexToInc[0]] as Array;

if (!a)

{

MessageBox.Show("Массив отсортирован");

}

else

{

MessageBox.Show("Массив не отсортирован");

}

indexToInc.RemoveAt(0);

}

DataGridViewOutput();

}

else

{

MessageBox.Show(ERR\_SELECTED, ERR, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

1. Класс Array

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace VP\_6

{

public class Array : IComparable

{

public int sum { get; set; }

private int[] intArray;

public int[] IntArray

{

get

{

return intArray;

}

set

{

if(value != null)

{

if(value.Length == n) {

intArray = value;

}

else

{

n = (uint)value.Length;

intArray = value;

}

}

else

{

intArray = null;

}

}

}

public override string ToString()

{

string ArrayToString = "";

if(intArray.Length !=0)

{

ArrayToString = ArrayToString + intArray[0].ToString();

}

return ArrayToString;

}

private uint n;

public uint N

{

get

{

return n;

}

set

{

if (value == 0)

{

intArray = null;

n = 0;

}

else if (value > n)

{

int[] newIntArray = new int[value];

for (uint i = 0; i < n; i++)

{

newIntArray[i] = IntArray[i];

}

n = value;

intArray = newIntArray;

}

else if (value < n)

{

int[] newIntArray = new int[value];

for (uint i = 0; i < value; i++)

{

newIntArray[i] = intArray[i];

}

n = value;

intArray = newIntArray;

}

}

}

public Array(uint n)

{

if (n != 0)

{

this.n = n;

intArray = new int[n];

}

else

{

this.n = 0;

intArray = null;

}

}

public Array(uint n, int[] IntArray)

{

if (n != 0)

{

this.n = n;

this.IntArray = new int[n];

if (n > IntArray.Length)

{

for (uint i = 0; i < IntArray.Length; i++)

{

this.IntArray[i] = IntArray[i];

}

}

else

{

for (uint i = 0; i < n; i++)

{

this.IntArray[i] = IntArray[i];

}

}

}

else

{

this.n = 0;

this.IntArray = null;

}

}

public Array(int[] IntArray)

{

n = (uint)IntArray.Length;

this.IntArray = IntArray;

}

public Array()

{

n = 0;

IntArray = null;

}

//????

public void KeyboardInput()

{

for (int i = 0; i < n; i++)

{

Console.Write("Введите {0}-й элемент массива: ", i);

intArray[i] = Convert.ToInt32(Console.ReadLine());

}

}

public void ConsoleOutput()

{

for (int i = 0; i < n; i++)

{

Console.Write("{0} ", intArray[i]);

}

Console.WriteLine();

}

public void BubbleSort()

{

for (int i = 0; i < n; i++)

for (int j = 0; j < n - 1; j++)

{

if (intArray[j] > intArray[j + 1])

{

int temp = intArray[j];

intArray[j] = intArray[j + 1];

intArray[j + 1] = temp;

}

}

}

public void BubbleSortDescending()

{

for (int i = 0; i < n; i++)

for (int j = 0; j < n - 1; j++)

{

if (intArray[j] < intArray[j + 1])

{

int temp = intArray[j];

intArray[j] = intArray[j + 1];

intArray[j + 1] = temp;

}

}

}

public int Summ()

{

int summ = 0;

for (int i = 0; i < n; i++)

summ = summ + intArray[i];

return summ;

}

public static Array operator ++(Array a)

{

for (int i = 0; i < a.n; i++)

{

a.intArray[i]++;

}

return a;

}

public static Array operator --(Array a)

{

for (int i = 0; i < a.n; i++)

{

a.intArray[i]--;

}

return a;

}

public static bool operator !(Array a)

{

for (int i = 0; i < a.n - 1; i++)

{

if (a.intArray[i] < a.intArray[i + 1])

{

return false;

}

}

return true;

}

public static Array operator \*(Array a1, Array a2)

{

uint maxN, minN;

if (a1.n > a2.n)

{

maxN = a1.n;

minN = a2.n;

}

else

{

maxN = a2.n;

minN = a1.n;

}

Array aNew = new Array(maxN);

for (uint i = 0; i < minN; i++)

{

aNew.intArray[i] = a1.intArray[i] \* a2.intArray[i];

}

return aNew;

}

public static bool operator true(Array array)

{

uint count = 0;

for (uint i = 0; i < array.n; i++)

{

if (array.intArray[i] != 0)

{

count++;

}

}

if (count != 0)

{

return true;

}

else

{

return false;

}

}

public static bool operator false(Array array)

{

uint count = 0;

for (uint i = 0; i < array.n; i++)

{

if (array.intArray[i] != 0)

{

count++;

}

}

if (count != 0)

{

return false;

}

else

{

return true;

}

}

public static bool operator ==(Array a1, Array a2)

{

if (a1.N == a2.N)

{

for (uint i = 0; i < a1.N; i++)

{

if (a1.IntArray[i] != a2.IntArray[i])

{

return false;

}

}

return true;

}

else

{

return false;

}

}

public static bool operator !=(Array a1, Array a2)

{

if (a1.N == a2.N)

{

for (uint i = 0; i < a1.N; i++)

{

if (a1.IntArray[i] != a1.IntArray[i])

{

return true;

}

}

return false;

}

else

{

return true;

}

}

public static bool operator <(Array a1, Array a2)

{

uint minN;

if (a1.N > a2.N)

{

minN = a2.N;

}

else

{

minN = a1.N;

}

for (uint i = 0; i < minN; i++)

{

if (a1.IntArray[i] > a2.IntArray[i])

{

return false;

}

else if (a1.IntArray[i] < a2.IntArray[i])

{

return true;

}

}

if (a1.N >= a2.N)

{

return false;

}

else

{

return true;

}

}

public static bool operator >(Array a1, Array a2)

{

uint minN;

if (a1.N > a2.N)

{

minN = a2.N;

}

else

{

minN = a1.N;

}

for (uint i = 0; i < minN; i++)

{

if (a1.IntArray[i] > a2.IntArray[i])

{

return true;

}

else if (a1.IntArray[i] < a2.IntArray[i])

{

return false;

}

}

if (a1.N > a2.N)

{

return true;

}

else

{

return false;

}

}

public int CompareTo(object other)

{

if (this == (other as Array)) return 0;

else if (this < (other as Array)) return -1;

else return 1;

}

}

}

Вывод: Проведено знакомство с интерфейсами и классами коллекций C#. Получены навыки при работе с классом коллекции ArrayList. Освоены приемы организации работы в Windows Form приложении через меню.

Контрольные вопросы

1. Интерфейс - средство С#, определяющее ряд методов для реализации в классе. Интерфейсы подобны абстрактным классам, но в интерфейсе вообще не предоставляется никакой реализации. В нем указывается только, что именно следует делать, но не как это делать.
2. Интерфейсы объявляются с помощью ключевого слова *interface*. Имена интерфейсов принято начинать с префикса *I* Особенности реализации интерфейса в классах
3. Коллекция является классом, поэтому необходимо объявить экземпляр класса перед добавлением в коллекцию элементов. Коллекции предоставляют более гибкий способ работы с группами объектов. В отличие от массивов, коллекция, с которой вы работаете, может расти или уменьшаться динамически при необходимости.
4. Обобщенные коллекции - это те же самые обобщенные классы. И опять же их использование перед необобщенными коллекциями имеет те же преимущества: повышение производительностиКлассы обобщенных коллекций. Необобщенные или простые коллекции определены в пространстве имен System.Collections. Их особенность состоит в том, что их функциональность, функциональные возможности описываются в интерфейсах, которые также находятся в этом пространстве имен.
5. Интерфейсы, представляющие базовую функциональность коллекций в пространстве имен *System.Collections: В среде .NET Framework поддерживаются пять типов коллекций: необобщенные, специальные, с поразрядной организацией, обобщен-ные и параллельные.*
6. В классе ArrayList определяется массив переменной длины, кото-рый состоит из ссылок на объекты и может динамически увеличивать и уменьшать свой размер Интерфейсы, реализованные в классе *System.Array*
7. Примеры методов класса *System.Array:* [*Clear*()](https://msdn.microsoft.com/ru-ru/library/system.collections.arraylist.clear(v=vs.110).aspx), [*Add*(*Object*)](https://msdn.microsoft.com/ru-ru/library/system.collections.arraylist.add(v=vs.110).aspx), [*IndexOf*(*Object*)](https://msdn.microsoft.com/ru-ru/library/7w3e62a8(v=vs.110).aspx)
8. Коллекция является классом, поэтому необходимо объявить новую коллекцию перед добавлением в неё элементов. Доступ к элементам этой коллекции осуществляется с помощью целочисленного индекса. Индексы в этой коллекции начинаются с нуля.
9. Для создания контекстного меню можно использовать класс *ContextMenu* или *ContextMenuStrip*.