Хотя в языке C# есть массивы, которые хранят в себе наборы однотипных объектов, но работать с ними не всегда удобно. Например, массив хранит фиксированное количество объектов, однако что если мы заранее не знаем, сколько нам потребуется объектов. И в этом случае намного удобнее применять коллекции. Еще один плюс коллекций состоит в том, что некоторые из них реализует стандартные структуры данных, например, стек, очередь, словарь, которые могут пригодиться для решения различных специальных задач.

Большая часть классов коллекций содержится в пространствах имен `System.Collections` (простые необобщенные классы коллекций),`System.Collections.Generic` (обобщенные или типизированные классы коллекций) и `System.Collections.Specialized` (специальные классы коллекций). Также для обеспечения параллельного выполнения задач и многопоточного доступа применяются классы коллекций из пространства имен `System.Collections.Concurrent`

ArrayList

Итак, класс **ArrayList** представляет коллекцию объектов. И если надо сохранить вместе разнотипные объекты - строки, числа и т.д., то данный класс как раз для этого подходит.

Основные методы класса:

- `int Add(object value)`: добавляет в список объект value
- `void AddRange(ICollection col)`: добавляет в список объекты коллекции col, которая представляет интерфейс ICollection - интерфейс, реализуемый коллекциями.
- `void Clear()`: удаляет из списка все элементы
- `bool Contains(object value)`: проверяет, содержится ли в списке объект value. Если содержится, возвращает true, иначе возвращает false
- `void CopyTo(Array array)`: копирует текущий список в массив array.
- `ArrayList GetRange(int index, int count)`: возвращает новый список ArrayList, который содержит count элементов текущего списка, начиная с индекса index
- `int IndexOf(object value)`: возвращает индекс элемента value
- `void Insert(int index, object value)`: вставляет в список по индексу index объект value
- `void InsertRange(int index, ICollection col)`: вставляет в список начиная с индекса index коллекцию ICollection
- `int LastIndexOf(object value)`: возвращает индекс последнего вхождения в списке объекта value

- `void Remove(object value)`: удаляет из списка объект value
- `void RemoveAt(int index)`: удаляет из списка элемент по индексу index
- `void RemoveRange(int index, int count)`: удаляет из списка count элементов, начиная с индекса index
- `void Reverse()`: переворачивает список
- `void SetRange(int index, ICollection col)`: копирует в список элементы коллекции col, начиная с индекса index
- `void Sort()`: сортирует коллекцию

Кроме того, с помощью свойства `Count` можно получить количество элементов в списке.

Посмотрим применение класса на примере.

```
using System;
using System.Collections;

namespace Collections
{
    class Program
    {
        static void Main(string[] args)
        {
            ArrayList list = new ArrayList();
            list.Add(2.3); // заносим в список объект типа double
            list.Add(55); // заносим в список объект типа int
            list.AddRange(new string[] { "Hello", "world" }); // заносим в список строковый массив

            // перебор значений
            foreach (object o in list)
            {
                Console.WriteLine(o);
            }

            // удаляем первый элемент
```

Во-первых, так как класс ArrayList находится в пространстве имен System.Collections, то подключаем его (`using System.Collections;`).

Вначале создаем объект коллекции через конструктор как объект любого другого класса: `ArrayList list = new ArrayList();`. При необходимости мы могли бы так же, как и с массивами, выполнить начальную инициализацию коллекции, например, `ArrayList list = new ArrayList(){1, 2, 5, "string", 7.7};`

Далее последовательно добавляем разные значения. Данный класс коллекции, как и большинство других коллекций, имеет два способа добавления: одиночного объекта через метод **Add** и набора объектов, например, массива или другой коллекции через метод **AddRange**

Через цикл `foreach` мы можем пройтись по всем объектам списка. И поскольку данная коллекция хранит разнородные объекты, а не только числа или строки, то в качестве типа перебираемых объектов выбран тип object: `foreach (object o in list)`

Многие коллекции, в том числе и ArrayList, реализуют удаление с помощью методов Remove/RemoveAt. В данном случае мы удаляем первый элемент, передавая в метод RemoveAt индекс удаляемого элемента.

В завершении мы опять же выводим элементы коллекции на экран только уже через цикл for. В данном случае с перебором коллекций дело обстоит также, как и с массивами. А число элементов коллекции мы можем получить через свойство `Count`

С помощью индексатора мы можем получить по индексу элемент коллекции так же, как и в массивах: `object firstObj = list[0];`