**15 Обобщения**

Задание 1. Создайте класс MyList<T>. Реализуйте возможность использования его экземпляра аналогично экземпляру класса List<T>. Минимально требуемый интерфейс взаимодействия с экземпляром, должен включать метод добавления элемента, индексатор для получения значения элемента по указанному индексу и свойство только для чтения для получения общего количества элементов.

Задание 3 (дополнительно). Создайте расширяющий метод: public static T[] GetArray<T>(this MyList<T> list) Примените расширяющий метод к экземпляру типа MyList<T>, разработанному в задании 2 для данного урока. Выведите на экран значения элементов массива, который вернул расширяющий метод GetArray().

Листинг программы:

static class MyListExtensions

{public static T[] GetArray<T>(this MyList<T> list)

{ T[] array = new T[list.Count]; for (int i = 0; i < list.Count; i++)

{array[i] = list[i];} return array;}}

class MyList<T>{private List<T> list;

public MyList() { list = new List<T>();}

public void Add(T item){ list.Add(item);}

public T this[int index] { get { return list[index]; } }

public int Count { get { return list.Count; }}}

class a1{static void Main(string[] args){

MyList<int> list = new MyList<int>(); list.Add(10); list.Add(20);

list.Add(30); int[] array = list.GetArray();

foreach (int item in array){ Console.WriteLine("Элементы массива, возвращаемые GetArray: " + item);}

Console.WriteLine("Количество элементов в списке: " + list.Count);}}

Таблица 15.1 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Элементы массива, возвращаемые GetArray: 10  Элементы массива, возвращаемые GetArray: 20  Элементы массива, возвращаемые GetArray: 30  Количество элементов в списке: 3 |

Источник: собственная разработка

Анализ результатов:

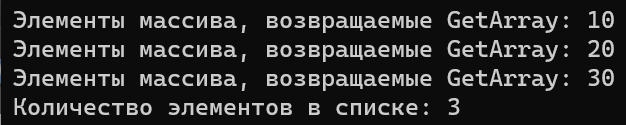


Рисунок 15.1 – Результат работы программы

Источник: собственная разработка

Задание 2. Создайте класс MyDictionary <TKey, TValue>. Реализуйте возможность использования его экземпляра аналогично экземпляру класса Dictionary.

Минимально требуемый интерфейс взаимодействия с экземпляром, должен включать метод добавления пар элементов, индексатор для получения значения элемента по указанному индексу и свойство только для чтения для получения общего количества пар элементов.

Листинг программы:

MyDictionary<string, int> dictionary = new MyDictionary<string, int>();

dictionary.Add("one", 1); dictionary.Add("two", 2); dictionary.Add("three", 3);

Console.WriteLine(dictionary["two"]); Console.WriteLine(dictionary.Count);

class MyDictionary<TKey, TValue>{

private List<TKey> keys; private List<TValue> values;

public MyDictionary(){keys = new List<TKey>();values = new List<TValue>();}

public void Add(TKey key, TValue value){ keys.Add(key);values.Add(value);}

public TValue this[TKey key] { get{int index =keys.IndexOf(key);if (index == -1)

throw new KeyNotFoundException(); return values[index];} set{

int index = keys.IndexOf(key); if (index == -1)

throw new KeyNotFoundException(); values[index] = value;}} public int Count

{ get { return keys.Count; }}}

Таблица 15.2 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 2 3 |

Источник: собственная разработка

Анализ результатов:



Рисунок 15.2 – Результат работы программы

Источник: собственная разработка

Задание 3. Создайте расширяющий метод: public static T[] GetArray<T>(this MyList<T> list) Примените расширяющий метод к экземпляру типа MyList<T>, разработанному в задании 2 для данного урока. Выведите на экран значения элементов массива, который вернул расширяющий метод GetArray().

Листинг программы:

static void Sum(int A, int N){int sum = 0;for (int i = 0; i <= N; i++)

{sum += (int)Math.Pow(A, i); Console.WriteLine($"Сумма: {sum}");

Thread.Sleep(500); }}

static void Product(int A, int N){int product = 1; for (int i = 1; i <= N; i++){

product \*= (int)Math.Pow(A, i); Console.WriteLine($"Произведение: {product}"); Thread.Sleep(500); }}

static void Main(string[] args){ Console.Write("Введите A:");

int A = int.Parse(Console.ReadLine()); Console.Write("Введите N:");

int N = int.Parse(Console.ReadLine());

Thread t1 = new Thread(() => Sum(A, N));

Thread t2 = new Thread(() => Product(A, N));

t1.Start(); t2.Start(); t1.Join(); t2.Join();

Console.WriteLine("Работа потоков завершена.");}

Таблица 15.3 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2; 8 | Сумма: 1 Произведение: 2  Сумма: 3 Произведение: 8  Произведение: 64 Сумма: 7  Сумма: 15 Произведение: 1024  Сумма: 31 Произведение: 32768  Произведение: 2097152 Сумма: 63  Произведение: 268435456 Сумма: 127  Произведение: 0 Сумма: 255  Сумма: 511  Работа потоков завершена. |

Источник: собственная разработка

Анализ результатов:

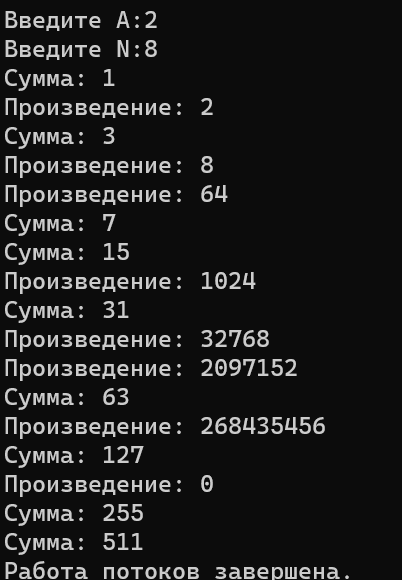


Рисунок 15.3 – Результат работы программы

Источник: собственная разработка