

Лекция 16. Списки

Курс «Программирование»

ПрИМО, 2 семестр

Щукин Александр

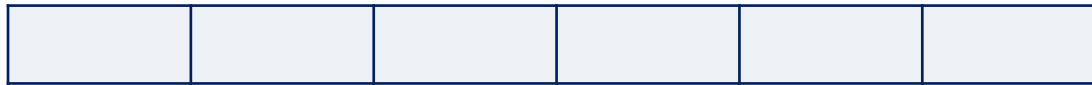
Валентинович

Понятие списка

- Что такое список — множество однотипных*, проиндексированных элементов, связанных друг с другом последовательно
- Сходство с массивом (одномерный)
 - Проиндексированные элементы (доступ по индексу)
 - Однотипные
- Отличие от массива
 - Расположение в памяти не последовательное
 - Как следствие, доступ к ячейке не за $O(1)$
 - Связанные элементы, как следствие вставка и удаление за $O(1)$

Варианты списков и использование

- Массив



- Однонаправленный список



- Варианты списков

- Однонаправленный
- Двухнаправленный
- Однонаправленный кольцевой
- Двухнаправленный кольцевой
- Стек
- Очередь

- Примеры использования

Варианты реализации

- Использовать готовые классы
 - **System.Collection.Generic** – универсальные шаблоны
 - **System.Collection**
- Самостоятельная реализация на базе массивов (использование курсоров)
- Самостоятельная реализация на базе объектов (использование ссылок)

ГОТОВЫЕ КЛАССЫ

Generic classes (using System.Collection.Generic)

- `List<TValue>`
- `Dictionary<Tkey, TValue>`
- `SortedList<Tkey, TValue>`
- `Queue<TValue>`
- `Stack<TValue>`

Regular classes (using System.Collection)

- `ArrayList`
- `SortedList`
- `Hashtable`

- `CollectionBase`
- `DictionaryBase`

Generic classes (универсальные классы)

- Универсальные классы вводят на платформе .NET Framework концепцию параметров универсального типа. Универсальный класс — это тип, спецификация которого отложена до момента объявления и создания экземпляров в клиентском коде.
- Универсальные классы и методы сочетают такие характеристики, как возможность многократного использования, типобезопасность и эффективность.
- Универсальные типы наиболее часто используются с коллекциями и методами, которые выполняют с ними операции.

List

- Объявление и создание:
 - `List<int> lstNew = new List<int>();`
- `Add(value)` - добавление, `Insert(index, value)` - вставка
- `Remove(value)`, `RemoveAt(index)` - удаление
- `Clear()` - очистка
- Доступ к элементу через `Z[index]`
- `Count` – количество элементов

List - примеры

```
// объявление, создание и инициализация строкового списка
List<string> towns = new List<string>{ "Moscow", "Minsk",
    "London", "Rome"};
```

```
// объявление и создание числового списка
List<int> nums = new List<int>();
nums.Add(32);
nums.Add(38);
nums.Add(10);
nums.Insert(2, 100);           //вставка 100 на позицию 2
nums.InsertRange(3, other);    //вставка элементов из other
```

```
// обход всех элементов
foreach(int x in nums) Console.WriteLine(x);
```


List - примеры

```
// обход всех элементов
```

```
foreach(int x in nums) Console.WriteLine(x);
```

```
// удаление всех элементов
```

```
while(nums.Count > 0)
```

```
    nums.RemoveAt(0);
```

```
// или
```

```
nums.Clear();
```

Dictionary

- Объявление и создание, ключ - `int`, значение - `string`
- `Dictionary<int, string> students = new Dictionary<int, string>();`
- `Add(key, value)`
- `Remove(key), Remove(value)`
- `Clear()`
- Доступ к элементу через `Z[key]`,
- `Count` – количество элементов
- `ContainsKey, ContainsValue` – возвращает `bool`
- `Keys` – возвращает коллекцию ключей

Dictionary - примеры

```
//  
Dictionary<int, string> students = new Dictionary<int, string>();  
students.Add(123, "Иванов");  
students.Add(124, "Сидоров");  
students.Add(125, "Иванов");  
Console.WriteLine(students[124]);  
Students[124] = "Петров";  
  
// ВЫВОД ЭЛЕМЕНТОВ  
foreach(string s in students.Values)  
    Console.WriteLine(s);  
  
// ВЫВОД ЭЛЕМЕНТОВ  
foreach (KeyValuePair<int, string> z in students)  
    Console.WriteLine(z.Value);
```

Stack

- **Push(value)** – записать значение
- **value Pop()** – извлечь значение
- **value Peek()** – взять, не удаляя
- **Clear()** - очистка
- **Count** – количество элементов

```
Stack<int> some = new Stack<int>( );  
some.Push(100);  
some.Push(200);  
Console.WriteLine(some.Pop());
```

Queue

- **Enqueue(value)** - добавить в очередь
- **value Dequeue()** - извлечь из очереди
- **value Peek()** - взять не удаляя
- **Clear()** - очистка
- **Count** - количество элементов

```
Queue<int> tasks = new Queue<int>( );  
tasks.Enqueue(100);  
tasks.Enqueue(200);  
Console.WriteLine(tasks.Dequeue( ));
```