Лекция 17. Коллекции

Козьминых Н.М.

дисциплина «Программирование»

Введение в коллекции

- Коллекция в C# представляет собой совокупность объектов
- Коллекции упрощают решение многих задач программирования на основе структур данных
- Коллекции предназначены для поддержки динамических массивов, связных списков, стеков, очередей и хеш-таблиц

Обзор коллекций

- **Необобщенные коллекции (System.Collections):** динамический массив, стек, очередь, словари. Коллекции не типизированы, поскольку в них хранятся ссылки на данные типа object.
- **Специальные коллекции (System.Collections.Specialized):** коллекции для символьных строк, специальные коллекции, в которых используется однонаправленный список.
- **Поразрядная коллекция (System.Collections) –** BitArray, поддерживающая поразрядные операции.
- **Обобщенные коллекции (System.Collections.Generic):** связные списки, стеки, очереди и словари. Коллекции являются типизированными в силу их обобщенного характера.
- **Параллельные коллекции (System.Collections.Concurrent)-**

Необобщенные коллекции

- пространство имен System.Collections;
- структуры данных общего назначения, оперирующие ссылками на объекты;
- можно хранить разнотипные данные:
  - «+» позволяют манипулировать объектом любого типа, хотя и не типизированным способом;
  - «-» не обеспечивают типовую безопасность.

Классы необобщенных коллекций

Класс	Описание
ArrayList	Определяет динамический массив, т.е. такой массив, который может при необходимости увеличивать свой размер
Hashtable	Определяет хеш-таблицу для пар «ключ-значение»
Queue	Определяет очередь, или список, действующий по принципу «первым пришел — первым обслужен».
SortedList	Определяет отсортированный список пар «ключ-значение»
Stack	Определяет стек, или список, действующий по принципу «первым пришел — последним обслужен»

Обобщенные коллекции

- пространство имен **System.Collections.Generic;**
- обобщенные коллекции типизированы;
- как правило, классы обобщенных коллекций являются не более чем обобщенными эквивалентами классов необобщенных коллекций, например, ArrayList → List, HashTable → Dictionary;

Классы обобщенных коллекций

Класс	Описание
Dictionary<Tkey, TValue>	Сохраняет пары «ключ-значение» (аналог класса Hashtable)
HashSet<T>	Сохраняет ряд уникальных значений, используя хеш таблицу
LinkedList<T>	Сохраняет элементы в двунаправленном списке
List<T>	Создает динамический массив (аналог класса ArrayList)
Queue<T>	Создает очередь (аналог класса Queue)
SortedDictionary <TKey, TValue>	Создает отсортированный список из пар «ключ-значение»
SortedList<TKey, TValue>	Создает отсортированный список из пар «ключ-значение» (аналог класса SortedList)
SortedSet<T>	Создает отсортированное множество
Stack<T>	Создает стек (аналог класса Stack)

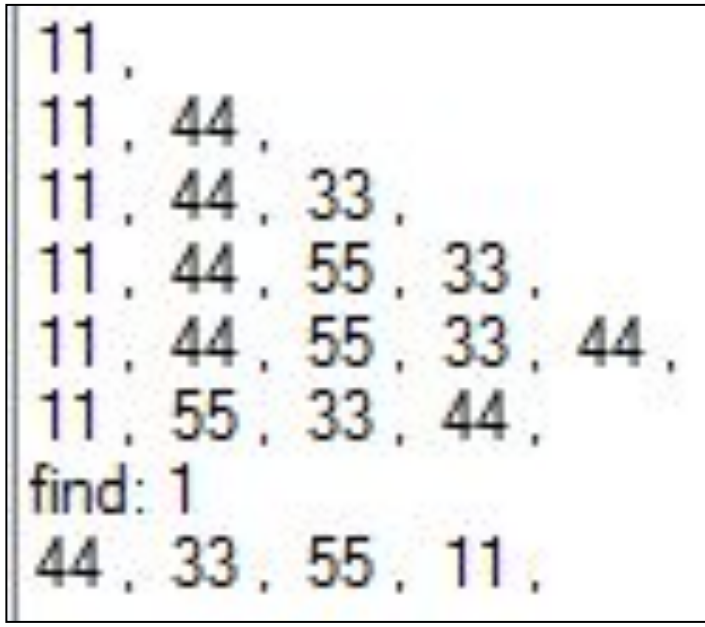
Класс ArrayList

- поддерживаются динамические массивы, расширяющиеся и сокращающиеся по мере необходимости.
- свойство Count - количество объектов, хранящихся в коллекции на данный момент

Метод	Описание
Add()	Добавляет новый элемент в конец списка
IndexOf()	Возвращает индекс первого вхождения объекта в вызывающей коллекции. Если искомый объект не обнаружен, возвращает значение -1
Insert()	Вставляет новый элемент по указанному индексу
Remove()	Удаляет первое вхождение указанного элемента из списка
Sort()	Сортирует вызывающую коллекцию по нарастающей

Класс ArrayList - пример


```
ArrayList arrLst = new ArrayList();
arrLst.Add("11"); outputArrayList(arrLst);
arrLst.Add("44"); outputArrayList(arrLst);
arrLst.Add("33"); outputArrayList(arrLst);
arrLst.Insert(2, "55"); outputArrayList(arrLst);
arrLst.Add("44"); outputArrayList(arrLst);
arrLst.Remove("44"); outputArrayList(arrLst);
listBox3.Items.Add("find: " + arrLst.IndexOf("55"));
arrLst.Reverse(); outputArrayList(arrLst);
...
private void outputArrayList(ArrayList arrLst){
   String ss = "";
   foreach (String item in arrLst){
        ss = ss + item + " ,  ";
   }
   listBox3.Items.Add(ss);
}
```



Класс Hashtable

- информация сохраняется в хеш-таблице с помощью механизма **хеширования;**
- хеш-код служит в качестве индекса, по которому в таблице хранятся искомые данные, соответствующие заданному ключу

Метод	Описание
ContainsKey()	Возвращает логическое значение true, если в вызывающей коллекции типа Hashtable содержится ключ, а иначе — логическое значение false
ContainsValue()	Возвращает логическое значение true, если в вызывающей коллекции типа Hashtable содержится значение, а иначе — логическое значение false

Класс Hashtable - пример

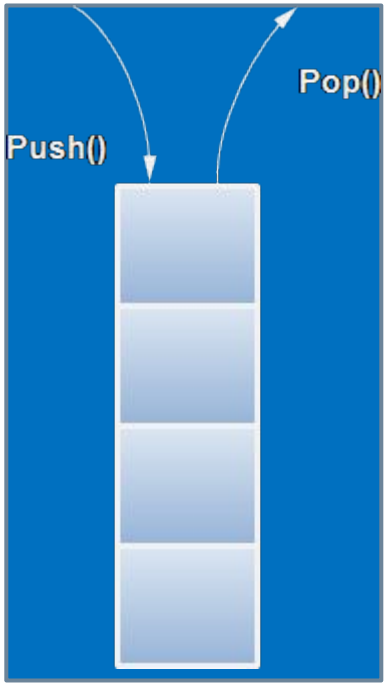

```
Hashtable hashtable = new Hashtable();
hashtable.Add("qwe", "111"); outputHashtable(hashtable);
hashtable.Add("zxc", "222"); outputHashtable(hashtable);
hashtable.Add("asd", "666"); outputHashtable(hashtable);
...
private void outputHashtable(Hashtable arrLst){
  String ss = "";
  foreach (String item in arrLst.Keys){
    ss = ss + item + " : " + arrLst[item] + " ,  ";
  }
  listBox6.Items.Add(ss);

  ss = "";
  foreach (String item in arrLst.Values){
    ss = ss + item + " ,  ";
  }
  listBox6.Items.Add(ss);
}
```

```
qwe : 111 ,
111 ,
qwe : 111 , zxc : 222 ,
111 , 222 ,
zxc : 222 , qwe : 111 , asd : 666 ,
222 , 111 , 666 ,
```

Стек: классы Stack и Stack<T>

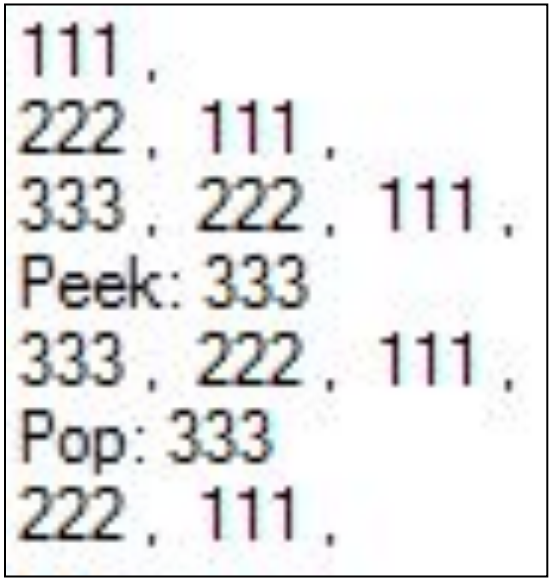
- контейнер, работающий по принципу "последний вошел, первый вышел" (last in, first out — LIFO)



Метод	Описание
Push()	Добавляет элемент в вершину стека
Pop()	Удаляет и возвращает элемент из вершины стека
Peek()	Возвращает элемент из вершины стека, не удаляя его при этом
Contains()	Метод Contains() проверяет наличие элемента в стеке и возвращает true в случае нахождения его там.

Стек: классы Stack и Stack<T> - пример


```
Stack stack = new Stack();
stack.Push("111"); outputStack(stack);
stack.Push("222"); outputStack(stack);
stack.Push("333"); outputStack(stack);
listBox5.Items.Add("Peek: " + stack.Peek());
outputStack(stack);
listBox5.Items.Add("Pop: " + stack.Pop());
outputStack(stack);
…
private void outputStack(Stack arrLst){
  String ss = "";
  foreach (String item in arrLst){
    ss = ss + item + " ,  ";
  }
  listBox5.Items.Add(ss);
}
```



Очередь: классы Queue и Queue<T>

- коллекция, в которой элементы обрабатываются по схеме "первый вошел, первый вышел" (first in, first out — FIFO

Метод	Описание
Enqueue()	Добавляет элемент в конец очереди
Dequeue()	Читает и удаляет элемент из головы очереди
Peek()	Читает элемент из головы очереди, но не удаляет его

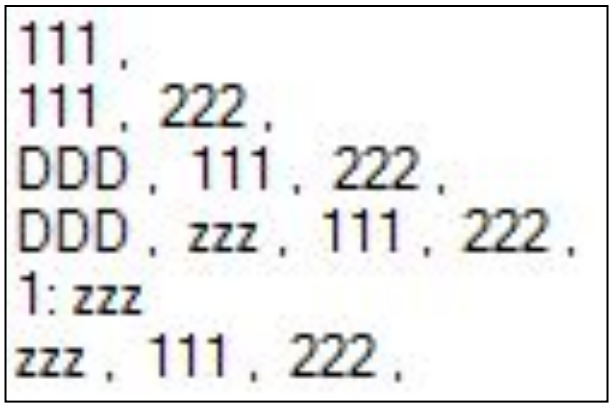
Связный список: класс LinkedList<T>

- двухсвязный список, в котором каждый элемент ссылается на следующий и предыдущий

Метод	Описание
AddAfter()	Добавляет в список узел со значением непосредственно после указанного узла
AddBefore()	Добавляет в список узел со значением value непосредственно перед указанным узлом
AddFirst(), AddLast()	Добавляют узел со значением в начало или в конец списка
Find()	Возвращает ссылку на первый узел в списке, имеющий передаваемое значение
Remove()	Удаляет из списка первый узел, содержащий передаваемое значение

Связный список: класс LinkedList<T> - пример


```
LinkedList<String> lst = new LinkedList<String>();
lst.AddFirst("111"); outputLinkedList(lst);
lst.AddLast("222"); outputLinkedList(lst);
lst.AddBefore(lst.Find("111"), "DDD");
outputLinkedList(lst);
lst.AddBefore(lst.Find("111"), "zzz");
outputLinkedList(lst);
listBox1.Items.Add("1: " + lst.ElementAt(1));
lst.Remove("DDD"); outputLinkedList(lst);
…
private void outputLinkedList(LinkedList<String> lst){
  String ss = "";
  foreach (String item in lst){
    ss = ss + item + " ,  ";
  }
  listBox1.Items.Add(ss);
}
```



Сортированный список: класс SortedList<TKey, TValue>

- коллекция, отсортированная по ключу

Метод	Описание
Add()	Добавляет в список пару «ключ-значение»
ContainsKey()	Возвращает логическое значение true, если вызывающий список содержит объект key в качестве ключа; а иначе — логическое значение false
ContainsValue()	Возвращает логическое значение true, если вызывающий список содержит значение value; в противном случае — логическое значение false
IndexOfKey(), IndexOfValue()	Возвращает индекс ключа или первого вхождения значения в вызывающем списке. Если искомый ключ или значение не обнаружены в списке, возвращается значение -1
Remove()	Удаляет из списка пару "ключ-значение" по указанному ключу key. При удачном исходе операции возвращается логическое значение true, а если ключ key отсутствует в списке — логическое значение false

Сортированный список: класс SortedList<TKey, TValue> - пример

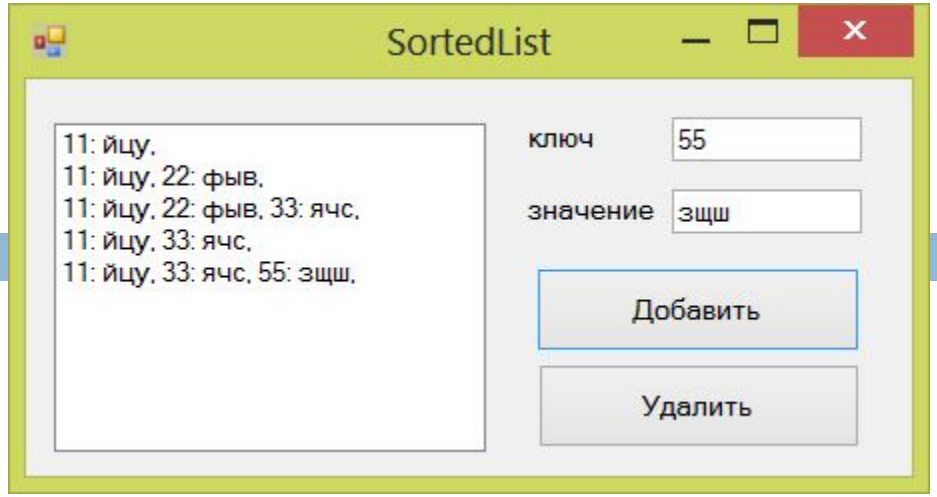

```
SortedList sortedList = new SortedList();
sortedList.Add("qwe", "111"); outputSortedList(sortedList);
sortedList.Add("asd", "333"); outputSortedList(sortedList);
sortedList.Add("zxc", "222"); outputSortedList(sortedList);

private void outputSortedList(SortedList arrLst){
  String ss = "";
  foreach (String item in arrLst.Keys){
    ss = ss + item + " : " + arrLst[item] + " ,  ";
  }
  listBox4.Items.Add(ss);

  ss = "";
  foreach (String item in arrLst.Values){
    ss = ss + item + " ,  ";
  }
  listBox4.Items.Add(ss);
}
```

```
qwe : 111 ,
111 ,
asd : 333 , qwe : 111 ,
333 , 111 ,
asd : 333 , qwe : 111 , zxc : 222 ,
333 , 111 , 222 ,
```

Пример




```
SortedList<string, string> list = new SortedList<string, string>();

 private void output()
        {
            StringBuilder result = new StringBuilder();
            foreach (string key in list.Keys)
                result.Append(key + ": " + list[key] + ", ");
            listBox.Items.Add(result.ToString());
        }
```

Пример (продолжение)


```
private void btnAdd_Click(object sender, EventArgs e)      {
    if (txtKey.Text == "")        {
        MessageBox.Show("Укажите ключ"); return;
    }
    if (txtValue.Text == "")        {
        MessageBox.Show("Укажите значение"); return;
    }
    if (list.ContainsKey(txtKey.Text))        {
        MessageBox.Show("Указанный ключ уже существует"); return;
    }
    list.Add(txtKey.Text, txtValue.Text);
    output();
}

private void btnDelete_Click(object sender, EventArgs e)    {
    if (!list.ContainsKey(txtKey.Text))        {
        MessageBox.Show("Указанного ключа уже существует"); return;
    }
    list.Remove(txtKey.Text);
    output();
}
```

Словарь: класс Dictionary<TKey, TValue>

- сложная структура данных, позволяющая обеспечить доступ к элементам по ключу;
- главное свойство словарей — быстрый поиск на основе ключей.

Метод	Описание
Add()	Добавляет в словарь пару «ключ-значение», определяемую параметрами key и value
ContainsKey()	Возвращает логическое значение true, если вызывающий словарь содержит объект key в качестве ключа; а иначе — логическое значение false
ContainsValue()	Возвращает логическое значение true, если вызывающий словарь содержит значение value; в противном случае — логическое значение false
Remove()	Удаляет ключ key из словаря. При удачном исходе операции возвращается логическое значение true, а если ключ key отсутствует в словаре — логическое значение false

Словарь: класс Dictionary<TKey, TValue> - пример


```
Dictionary<String, String> dict = new Dictionary<String, String>();
dict.Add("qwe", "111"); outputDictionary(dict);
dict.Add("asd", "666"); outputDictionary(dict);
dict.Add("zxc", "444"); outputDictionary(dict);
…
private void outputDictionary(Dictionary<String, String> arrLst){
  String ss = "";
  foreach (String item in arrLst.Keys){
    ss = ss + item + " : " +arrLst[item] + " ,  ";
  }
  listBox7.Items.Add(ss);

  ss = "";
  foreach (String item in arrLst.Values){
    ss = ss + item + " ,  ";
  }
  listBox7.Items.Add(ss);
}
```

```
qwe : 111 ,
111 ,
qwe : 111 , asd : 666 ,
111 , 666 ,
qwe : 111 , asd : 666 , zxc : 444 ,
111 , 666 , 444 ,
```

Множества: классы HashSet<T> и SortedSet<T>

- коллекция, содержащаяся только отличающиеся элементы

Метод	Описание
Add()	Добавляет элемент в набор и возвращает значение, указывающее, что элемент был добавлен успешно
Remove()	Удаляет указанный элемент из набора
Contains()	Определяет, содержит ли набор указанный элемент
IsSubsetOf()	Определяет, является ли объект SortedSet<T> подмножеством указанной коллекции

Множества: классы HashSet<T> и SortedSet<T> - пример


```
SortedSet<String> sSortSet = new SortedSet<String>();
sSortSet.Add("111"); outputSortedSet(sSortSet);
sSortSet.Add("555"); outputSortedSet(sSortSet);
sSortSet.Add("222"); outputSortedSet(sSortSet);

SortedSet<String> sSortSet2 = new SortedSet<String>();
sSortSet2.Add("111");

listBox2.Items.Add( sSortSet2.IsSubsetOf(sSortSet));
outputSortedSet(sSortSet);
…
private void outputSortedSet(SortedSet<String> sSortSet){
  String ss = "";
  foreach (String item in sSortSet){
    ss = ss + item + " ,  ";
  }
  listBox2.Items.Add(ss);
}
```

```
111 ,
111 , 555 ,
111 , 222 , 555 ,
True
111 , 222 , 555 ,
```

Спасибо за внимание

Вопросы…