Лабораторная работа №1по курсу
«Базовые компоненты интернет-технологий»

Выполнил: Саврасов П.А. Группа РТ5-31

Описание задания лабораторной работы.

Разработать программу, реализующую работу с коллекциями.

Текст программы на языке C#.

Класс Program:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace LR3
{
   class Program
   {
      public static void Main(string[] args)
      {
         string P =" ";
         int Mo = 0;
         int[] FixedModes =new int[10];
         NumItem NItem = new NumItem();
         Word[] W = new Word[10];
         Code[] C = new Code[10];
         Mixed[] M = new Mixed[10];
         Number[] N = new Number[10];
         SetWordCodeMixed SWCMc = new SetWordCodeMixed();
         List<NumItem> ItemList = new List<NumItem>();


         for (int i=0; i< 4; i++)
         {
            Mo = SWCMc.TypeChoser(Mo);
            P=SWCMc.SWCM(Mo,P);
            switch(Mo)
            {
                  case 1: {W[i] = new Word(i+1,P);ItemList.Add(W[i]); break;};
                  case 2: {C[i] = new Code(i+1,P);ItemList.Add(C[i]); break;};
                  case 3: {M[i] = new Mixed(i+1,P);ItemList.Add(M[i]); break;};
                  case 4: {N[i] = new Number(i+1);ItemList.Add(N[i]); break;};
              }
            FixedModes[i]=Mo;
         }
         Console.WriteLine("\n\nНажмите любую кнопку для продолжения.");
         Console.ReadKey(true);
         Console.Clear();
         NongenericList NL = new NongenericList(ItemList);
         SimpleStack ST = new SimpleStack(ItemList);
         Dictionary Dt = new Dictionary(ItemList);
         SparseMatrixFiller SMF = new SparseMatrixFiller(ItemList,FixedModes);
         SortByLength SBL = new SortByLength(ItemList);

         Console.ReadKey(true);
```

```
    }
  }
}
```

Класс NumItem:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR3
{
/// <summary>
  /// Родительский класс NumName, включает в себя:
  ///    свойство Name(данные элемента)
  ///    параметр num(номер эл-та последовательности)
  ///    метод ToString(), выводящий информацию об элементе в консоль.
  ///      метод CompareTo() обеспечивает возможность сортировки по длине их полного
имени.
  /// </summary>
  class NumItem:IComparable
  {
      public string Name
      {
          get{return this._Name;}
          set{this._Name=value;}
       }
      public int num
      {
          get{return this._num;}
          set{this._num=value;}
       }
       string _Name;
       public int _num;
      public override string ToString()
      {
            return  "ID:" + this._num.ToString()+"   Содержание:" +  this._Name;
      }
       public int CompareTo(object obj)
       {
          NumItem NP = (NumItem)obj;
          if (this._Name.Length < NP._Name.Length) return (-1);
          else if (this._Name.Length == NP._Name.Length) {return 0;}
          else return 1;
       }
  }
}
```

Класс Word:

```
using System;

namespace LR3
{
   /// <summary>
   /// Дочерний класс Word,
   /// является элементом последовательности типа "Слово",
   /// Содержит:
   ///    numer - номер элемента в последовательности
   ///    Метод Word - заполнение наследуемого свойства Name и номера numer
   ///    Вывод информации через интерфейс IPrint.
   /// </summary>

   class Word : NumItem
   {
      public Word(int n,string C)
      {
         this.num = n;
         this.Name =" Слово: "+ C;
      }
      public void Print()
      {
         Console.WriteLine(this.ToString());
      }
   }
}
```

Класс Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace LR3

{
   /// <summary>
   /// Дочерний класс Word,
   /// является элементом последовательности типа "Код",
   /// Содержит:
   ///    numer - номер элемента в последовательности
   ///    Метод Code - заполнение наследуемого свойства Name и номера numer
   ///    Вывод информации через интерфейс IPrint.
   /// </summary>

   class Code : NumItem
   {
      public Code(int n,string C)
      {
         this.num = n;
         this.Name =" Код: "+ C;
      }
```

```
        public void Print()
        {
            Console.WriteLine(this.ToString());
        }
    }
}
```

Класс Mixed:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR3
{
    /// <summary>
    /// Дочерний класс Mixed,
    /// является элементом последовательности типа "Mixed",
    /// Содержит:
    ///     numer - номер элемента в последовательности
    ///     Метод Name - заполнение наследуемого свойства Name и номера numer
    ///     Вывод информации через интерфейс IPrint.
    /// </summary>

    class Mixed : NumItem
    {
        public Mixed(int n,string C)
        {
            this.num = n;
            this.Name =" Mixed: "+ C;
        }
        public void Print()
        {
            Console.WriteLine(this.ToString());
        }
    }
}
```

Класс Number:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace LR3
{
    /// <summary>
    /// Дочерний класс Number,
    /// является элементом последовательности типа "Номер",
    /// Содержит:
    ///     numer - номер элемента в последовательности
```

```
    ///    Вывод информации через интерфейс IPrint.
    /// </summary>

    class Number : NumItem
    {
        public Number(int n)
        {
            this.num = n;
            this.Name = " Номер: "+this.num.ToString();
        }
        public void Print()
        {
            Console.WriteLine(this.ToString());
        }
    }
}
```

Класс SetWordCodeMixed:

```
namespace LR3
{
    /// <summary>
    /// Класс заполнения элементов последовательности в зависимости от выбранного типа элемента.
    /// Содержит методы:
    ///    TypeChoser - выбор типа элемента
    ///    SWCM - заполнение элемента в зависимости от его типа
    /// </summary>
    class SetWordCodeMixed
    {
        public int TypeChoser (int Mode)
        {
            bool Err = false;
            Console.WriteLine("Тип вводимого объекта:\n  1.Имя.   2.Код.   3.Смешанный.   4.Номер.");
            while((Err==false))
            {
                string c = Console.ReadLine();
                Err=int.TryParse(c,out Mode);
                if((Err==false)||(Mode<1)||(Mode>4)) {Console.WriteLine("Ошибка!Неверно выбран тип объекта./n");  Console.WriteLine("Тип вводимого объекта:/n  1.Имя.   2.Код./n"); Err=false;}
                else Err=true;
            }
            return Mode;
        }

        public string SWCM(int Mode, string C)
        {
            switch(Mode)
            {
                case 1:
                    {
                        Console.WriteLine("Задайте Слово (должно состоять только из букв).");
                        bool Err1 = false;
                        while((Err1==false))
```

```
                {
                    C=Console.ReadLine();
                    Err1=true;
                    for (int i = 0; i < C.Length; i++)
                    {if(char.IsLetter(C[i])==false) Err1=false;}
                    if((Err1==false)) {Console.WriteLine("Ошибка!Неверно введено
слово.");Console.WriteLine("Задайте Слово (должно состоять только из букв)."); Err1=false;}
                    else Err1=true;
                }
                break;
            }
          case 2:
            {
                int a;
                Console.WriteLine("Задайте Код (должен состоять только из чисел).");
                bool Err1 = false;
                while((Err1==false))
                {
                    C=Console.ReadLine();
                    Err1=int.TryParse(C,out a);
                    if((Err1==false)) {Console.WriteLine("Ошибка!Неверно введён
код.");Console.WriteLine("Задайте Код (должен состоять только из чисел)./n"); Err1=false;}
                    else Err1=true;
                }
                break;
            }
            case 3:
            {
                Console.WriteLine("Задайте Смешанную последовательность символов и чисел.");
                C=Console.ReadLine();
                break;
            }
          }
        return C;
        }
    }
}
```

Класс SortByLength:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace LR3
{
  /// <summary>
  /// Сортировщик последовательностей элементов по длине их содержимого, включая название
типа.
  /// Содержит метод SortByLength, на вход которому передаётся список. Метод выводит
элементы
  /// списка до сортировки, после чего сортирует их, а потом выводит их снова.
  /// </summary>
  class SortByLength
```

```
    {
        public SortByLength(List<NumItem> ItemList)
        {
            Console.WriteLine("\nПеред сортировкой:");
            foreach (var x in ItemList) Console.WriteLine(x);
            ItemList.Sort();
            Console.WriteLine("\nПосле сортировки:");
            foreach (var x in ItemList) Console.WriteLine(x);
            Console.WriteLine("\n\nНажмите любую кнопку для продолжения.");
            Console.ReadKey(true);
            Console.Clear();
        }

    }
}
```

Класс NongenericList:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace LR3
{
    /// <summary>
    /// Description of Class1.
    /// </summary>
    class NongenericList
    {
        public NongenericList(List<NumItem> ItemList)
        {
            ArrayList AL = new ArrayList();
            bool AreThereCode = false;
            bool AreThereWord = false;
            bool AreThereNumber = false;
            bool AreThereMixed = false;
            foreach (var x in ItemList) AL.Add(x);
            Console.WriteLine("Сортировка элементов по типу:");
            foreach(object o in AL)
            {
                string type = o.GetType().Name;
                if (type == "Code") AreThereCode = true;
                if (type == "Word") AreThereWord = true;
                if (type == "Mixed") AreThereMixed = true;
                if (type == "Number") AreThereNumber = true;
            }
            if(AreThereWord == true)
            {
                Console.WriteLine("Слова:");
                foreach (object o in AL)
                {
                    string type = o.GetType().Name;
                    if (type == "Word") Console.WriteLine(o.ToString());
```

```
            }
        }
        if(AreThereCode == true)
        {
            Console.WriteLine("Коды:");
            foreach (object o in AL)
            {
                string type = o.GetType().Name;
                if (type == "Code") Console.WriteLine(o.ToString());
            }
        }
        if(AreThereMixed == true)
        {
        Console.WriteLine("Смешанные:");
            foreach (object o in AL)
            {
                string type = o.GetType().Name;
                if (type == "Mixed") Console.WriteLine(o.ToString());
            }
        }
        if(AreThereNumber == true)
        {
            Console.WriteLine("Номера:");
            foreach (object o in AL)
            {
                string type = o.GetType().Name;
                if (type == "Number") Console.WriteLine(o.ToString());
            }
        }
        Console.WriteLine("\n\nНажмите любую кнопку для продолжения.");
          Console.ReadKey(true);
         Console.Clear();
    }
  }
}
```

Класс SimpleStack:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR3
{
  /// <summary>
  /// Пример работы стека.
  /// </summary>
  class SimpleStack
  {
    public SimpleStack(List<NumItem> ItemList)
    {
      Stack<NumItem> TypeStack = new Stack<NumItem>();
```

```
            Console.WriteLine("Порядок записи в Стек:");
            foreach (var x in ItemList){ Console.WriteLine(x); TypeStack.Push(x);}
            Console.WriteLine("Порядок вывода стека:");
            foreach (var t in TypeStack){ Console.WriteLine(t);}
            Console.WriteLine("Пример удалениея через Pop():\n Удалён первый элемент.");
            NumItem NI = TypeStack.Pop();
            Console.WriteLine("Остались:.");
            foreach (var t in TypeStack){ Console.WriteLine(t);}
            Console.WriteLine("\n\nНажмите любую кнопку для продолжения.");
            Console.ReadKey(true);
            Console.Clear();
        }
    }
}
```

Класс Dictionary:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR3
{
    /// <summary>
    /// Класс Dictionary,строит словарь из полученного списка, имеет:
    ///     Конструктор Dictionary- обеспечивает заполнение словаря по списку.
    ///     Метод DictionaryPrint - обеспечивает вывод солваря с последуюшим удалением данных из консоли.
    /// </summary>
    class Dictionary
    {
        public static Dictionary<int, string> NameNumDictionary = new Dictionary<int, string>();
        public Dictionary(List<NumItem> ItemList)
        {
            foreach (var x in ItemList) NameNumDictionary.Add(x.num,x.Name);
            DictionaryPrint(NameNumDictionary);

        }
        public void DictionaryPrint(Dictionary<int, string> NameNumDictionary)
        {
            Console.WriteLine("Словарь:");
            foreach (KeyValuePair<int, string> v in NameNumDictionary)
            {
                Console.WriteLine(v.Key.ToString() + " -" + v.Value);
            }
            Console.WriteLine("\n\nНажмите любую кнопку для продолжения.");
            Console.ReadKey(true);
            Console.Clear();
        }
    }
}
```

Класс ItemMatrixCheckEmpty:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR3
{
   /// <summary>
   /// Description of Class1.
   /// </summary>
   class ItemMatrixCheckEmpty : IMatrixCheckEmpty<NumItem>
   {
      public NumItem getEmptyElement()
      {
         return null;
      }
      public bool checkEmptyElement(NumItem element)
      {
         bool Result = false;
         if (element == null)
         {
            Result = true;
         }
         return Result;
      }
   }
}
```

Класс SparseMatrixFiller:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace LR3
{
   /// <summary>
   /// Description of Sparse_Matrix_Filler.
   /// </summary>
   class SparseMatrixFiller
   {
      public SparseMatrixFiller(List<NumItem> ItemList,int[] FixedModes)
      {
         Matrix<NumItem> matrix = new Matrix<NumItem>(6, 6, new ItemMatrixCheckEmpty());
```

```
            int i=0;
            Console.WriteLine("\nМатрица");
            foreach (var v in ItemList)
            {
               matrix[v._num, FixedModes[i]] = v;
               i=i+1;
            }
            Console.WriteLine(matrix.ToString());
        }
    }
}
```

Класс SparseMatrix:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace LR3
{
    /// <summary>
    /// Данные в матрице записываются в таком прядке:
    ///     Строки - номер элемента.
    ///     Столбцы - тип элемента.
    /// </summary>
    public class Matrix<T>
    {
        Dictionary<string, T> _matrix = new Dictionary<string, T>();
        int maxX;
        int maxY;
        IMatrixCheckEmpty<T> checkEmpty;
        public Matrix(int px, int py, IMatrixCheckEmpty<T> checkEmptyParam)
        {
            this.maxX = px;
            this.maxY = py;
            this.checkEmpty = checkEmptyParam;
        }
         public T this[int x, int y]
         {
            set
            {
               CheckBounds(x, y);
               string key = DictKey(x, y);
               this._matrix.Add(key, value);
            }
            get
```

```
    {
        CheckBounds(x, y);
        string key = DictKey(x, y);
        if (this._matrix.ContainsKey(key))
        {
            return this._matrix[key];
        }
        else
        {
            return this.checkEmpty.getEmptyElement();
        }
    }
}
void CheckBounds(int x, int y)
{
    if (x < 0 || x >= this.maxX)
    {
        throw new ArgumentOutOfRangeException("x", "x=" + x + " выходит за границы");
    }
    if (y < 0 || y >= this.maxY)
    {
        throw new ArgumentOutOfRangeException("y", "y=" + y + " выходит за границы");
    }
}
string DictKey(int x, int y)
{
    return x.ToString() + "_" + y.ToString();
}
public override string ToString()
{
    StringBuilder b = new StringBuilder();
    for (int j = 0; j < this.maxY; j++)
    {
        b.Append("["); for (int i = 0; i < this.maxX; i++)
        {
            if (i > 0)
            {
                b.Append("\t");
            }
            if (!this.checkEmpty.checkEmptyElement(this[i, j]))
            {
                b.Append(this[i, j].ToString());
            }
            else
            {
                b.Append(" _ ");
            }
```

```
            }
            b.Append("]\n");
        }
        return b.ToString();
    }
  }
}
```

Интерфейс IPrint:

```
using System;

namespace LR3
{
  /// <summary>
  /// Обеспечивает вывод информации на консоль.
  /// </summary>
  interface IPrint
  {
    void Print();
  }
}
```

Интерфейс IMatrixCheckEmpty:

```
using System;

namespace LR3
{
  /// <summary>
  /// Description of Interface1.
  /// </summary>
  public interface IMatrixCheckEmpty<T>
  {
    T getEmptyElement();
    bool checkEmptyElement(T element);
  }
}
```

Результаты выполнения программы, экранные формы:

```
Тип вводимого объекта:
  1.Имя.   2.Код.   3.Смешанный.   4.Номер.
1
Задайте Слово (должно состоять только из букв).
1
Ошибка!Неверно введено слово.
Задайте Слово (должно состоять только из букв).
qwed
Тип вводимого объекта:
  1.Имя.   2.Код.   3.Смешанный.   4.Номер.
2
Задайте Код (должен состоять только из чисел).
1234w
Ошибка!Неверно введён код.
Задайте Код (должен состоять только из чисел)./n
1232
Тип вводимого объекта:
  1.Имя.   2.Код.   3.Смешанный.   4.Номер.
3
Задайте Смешанную последовательность символов и чисел.
fews2
Тип вводимого объекта:
  1.Имя.   2.Код.   3.Смешанный.   4.Номер.
4


Нажмите любую кнопку для продолжения.
```

```
Сортировка элементов по типу:
Слова:
ID:1   Содержание: Слово: qwdfre
ID:2   Содержание: Слово: qwsdfewqa
Коды:
ID:3   Содержание: Код: 12345
Номера:
ID:4   Содержание: Номер: 4


Нажмите любую кнопку для продолжения.
```

```
Порядок записи в Стек:
ID:1   Содержание: Слово: qwdfre
ID:2   Содержание: Слово: qwsdfewqa
ID:3   Содержание: Код: 12345
ID:4   Содержание: Номер: 4
Порядок вывода стека:
ID:4   Содержание: Номер: 4
ID:3   Содержание: Код: 12345
ID:2   Содержание: Слово: qwsdfewqa
ID:1   Содержание: Слово: qwdfre
Пример удалениея через Pop():
 Удалён первый элемент.
Остались:.
ID:3   Содержание: Код: 12345
ID:2   Содержание: Слово: qwsdfewqa
ID:1   Содержание: Слово: qwdfre


Нажмите любую кнопку для продолжения.
```

```
Словарь:
1 - Слово: qwdfre
2 - Слово: qwsdfewqa
3 - Код: 12345
4 - Номер: 4


Нажмите любую кнопку для продолжения.
```

```
Матрица
[ _       _       _       _       _ ]
[ _     ID:1   Содержание: Слово: qwdfre       ID:2   Содержание: Слово: qwsdfe
wqa       _       _       _ ]
[ _       _       _     ID:3   Содержание: Код: 12345    _       _ ]
[ _       _       _       _       _       _ ]
[ _       _       _       _     ID:4   Содержание: Номер: 4      _ ]
[ _       _       _       _       _       _ ]


Перед сортировкой:
ID:1   Содержание: Слово: qwdfre
ID:2   Содержание: Слово: qwsdfewqa
ID:3   Содержание: Код: 12345
ID:4   Содержание: Номер: 4

После сортировки:
ID:4   Содержание: Номер: 4
ID:3   Содержание: Код: 12345
ID:1   Содержание: Слово: qwdfre
ID:2   Содержание: Слово: qwsdfewqa


Нажмите любую кнопку для продолжения.
```