**ПРАКТИЧЕСКОЕ ЗАДАНИЕ 18**

**КОЛЛЕКЦИИ И КЛАССЫ ПРОТОТИПЫ**

Задание 1. Решить задачи с использованием класса Stack. Пусть символ # определен в текстовом редакторе как стирающий символ. Backspace, т.е. строка abc#d##c в действительности является строкой ac. Дан текст, в котором встречается символ #. Преобразовать его с учетом действия этого символа.

Листинг программы:

class Program

{

static void Main(string[] args)

{

string a = "abc#d##c";

Stack<char> stack = new Stack<char>();

foreach (var c in a)

{

if (c == '#')

{

if (stack.Count > 0)

stack.Pop();

}

else

{

stack.Push(c);

}

}

var array = stack.ToArray();

Array.Reverse(array);

string s = new string(array);

Console.WriteLine($"Начальная строка: {a}");

Console.WriteLine($"Конечная строка: {s}");

}

}

Таблица 18.1 – Выходные данные

|  |
| --- |
| Выходные данные |
| Начальная строка: abc#d##c  Конечная строка: ac |

Анализ результатов:



Рисунок 18.1 – Результат работы программы

Источник: собственная разработка

Задание 2. Дан текстовый файл. За один просмотр файла напечатать элементы файла вследующем порядке: сначала все символы, отличные от цифр, а затем все цифры, сохраняя исходный порядок в каждой группе символов.

Листинг программы:

class Program

{

static void Main(string[] args)

{

Queue<char> symbols = new Queue<char>();

Queue<char> digits = new Queue<char>();

using (StreamReader sr = new StreamReader(@"E:\18.txt"))

{

while (!sr.EndOfStream)

{

char s = (char)sr.Read();

if (char.IsDigit(s)) digits.Enqueue(s);

else symbols.Enqueue(s);

}

}

Console.WriteLine(string.Join("", symbols));

Console.WriteLine(string.Join("", digits));

Console.ReadKey();

}

}

}

Таблица 18.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| my  number  is  232  it  is  true | my  number  is  it  is  true  232 |

Анализ результатов:

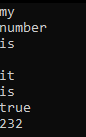


Рисунок 18.2 – Результат работы программы

Источник: собственная разработка

Задание 3. Создать класс-прототип, с обобщенным методом, с методами добавления,удаления, клонирования экземпляров класса.

В основной программе продемонстрировать:

функциональность созданного класса и всех его методов;

использование обнуляемых типов,

использование абстрактных структур

Листинг программы:

public class Program

{

static void Main(string[] args)

{

Sp SpList = new Sp();

Dictionary<string, string> city = new Dictionary<string, string>(2);

for (int i = 0; i < 2; i++)

{

Console.WriteLine("Введите вид спорта");

string s = null;

s = Console.ReadLine();

Console.WriteLine("Введите город");

string g = Console.ReadLine();

Console.WriteLine("Введите дату проведения");

string ds = Console.ReadLine();

SpList.Add(new DataS(s, g, ds));

var key = $"{i + 1}";

city[key] = g;

}

Console.WriteLine("=======================");

foreach (KeyValuePair<string, string> keyValue in city)

{

Console.WriteLine(keyValue.Key + " - " + keyValue.Value);

}

Console.WriteLine("=======================");

foreach (DataS x in SpList)

x.PrintSport();

SpList.RemoveAt(1);

Console.WriteLine("=======================");

Sp c1 = (Sp)SpList.Clone();

foreach (DataS x in c1)

x.PrintSport();

Console.ReadLine();

}

}

public class InfoObj

{

public static void Info<tip>(tip obj)

where tip : Sport

{

obj.PrintSport();

}

}

public class DataS : Sport

{

string ds;

public DataS(string s, string g, string ds) : base(s, g)

{

this.ds = ds;

}

public override void PrintSport()

{

Console.WriteLine("В " + g + " состоится игра по " + s + ": дата проведения " + ds);

}

}

}

public class Sport

{

public string s;

public string g;

public Sport(string s, string g)

{

this.s = s;

this.g = g;

}

virtual public void PrintSport()

{

Console.WriteLine("В " + g + " состоится игра по " + s);

}

}

}

public class Sp : IEnumerable

{

private ArrayList list;

public Sp()

{

list = new ArrayList();

}

public Sp(ArrayList a)

{

list = a;

}

public void Add(DataS m)

{

list.Add(m);

}

public Sp Clone()

{

return new Sp(list);

}

public void RemoveAt(int i)

{

list.RemoveAt(i);

}

public void Clear()

{

list.Clear();

}

public IEnumerator GetEnumerator()

{ return list.GetEnumerator(); }

}

}

Таблица 18.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Легкая атлетика  Введите город  Гродно  Введите дату проведения  14 февраля  Введите вид спорта  Гимнастика  Введите город  Минск  Введите дату проведения  10 марта | 1 - Гродно  2 - Минск  =======================  В Гродно состоится игра по Легкая атлетика: дата проведения 14 февраля  В Минск состоится игра по Гимнастика: дата проведения 10 марта |

Анализ результатов:

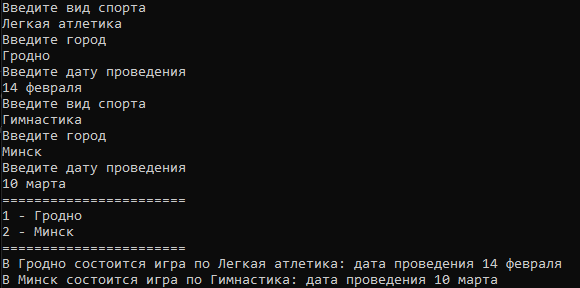


Рисунок 18.3 – Результат работы программы

Источник: собственная разработка