**18 КОЛЛЕКЦИИ. КЛАССЫ-ПРОТОТИПЫ**

Задание 1. 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.

Листининг программы:

using System;

using System.Collections.Generic;

namespace task1

{

class Program

{

static void Main(string[] args)

{

Console.Write("Введите строку: ");

string line = Console.ReadLine();

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

foreach (var l in line)

{

if (l == '#')

{

if (stack.Count > 0)

stack.Pop();

}

else

{

stack.Push(l);

}

}

var array = stack.ToArray();

Array.Reverse(array);

string newLine = new string(array);

Console.WriteLine("Обработанная строка: "+newLine);

}

}

}

Таблица 18.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Привет, как твои дела? | Обработанная строка: Привет, как твои дела? |

Анализ результатов:



Рисунок 18.1 – Результат работы программы

Источник – собственная разработка

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

Листининг программы:

using System;

using System.IO;

using System.Collections.Generic;

namespace task2

{

class Program

{

static void Main(string[] args)

{

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

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

using (StreamReader sr = new StreamReader("O:\\колледж\\КПиЯП-практика\\Задания\\task18\\task2\\text.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)+ string.Join("", digits));

}

}

}

Таблица 18.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| text.txt | f dsf sdf sd  r  r  f fweff  ew fwefff  134563234232323232323 |

Анализ результатов:

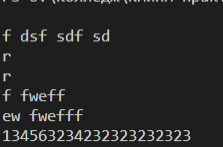


Рисунок 18.2 – Результат работы программы

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 (List&lt;P&gt; Dictionary&lt;Key,Value&gt;).

Листининг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

namespace \_1

{

class Hike

{

public string name;

public int countP;

public Hike(string name, int countP)

{

this.name = name;

this.countP = countP;

}

virtual public void PrintWeather()

{

Console.WriteLine(" В походе-" +name + " количетво человек-" + countP);

}

}

class InfoObj

{

public static void Info<tip>(tip obj)

where tip : Hike

{

obj.PrintWeather();

}

}

class Hi : IEnumerable

{

private ArrayList list;

public Hi()

{

list = new ArrayList();

}

public Hi(ArrayList a)

{

list = a;

}

public void Add(Hike m)

{

list.Add(m);

}

public Hi Clone()

{

return new Hi(list);

}

public void RemoveAt(int i)

{

list.RemoveAt(i);

}

public void Clear()

{

list.Clear();

}

public IEnumerator GetEnumerator()

{

return list.GetEnumerator();

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine(" \_\_\_\_\_Пример\_\_\_\_\_\_ ");

Hike dem = new Hike("Весельчак",13);

InfoObj.Info<Hike>(dem);

Console.WriteLine(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ");

Hi Hikelist = new Hi();

Dictionary <int, string>hike = new Dictionary<int, string>(3);

for (int i = 0; i <3; i++)

{

Console.WriteLine("Введите название похода");

string w = null;

w = Console.ReadLine();

Console.WriteLine("Введите количество человек в походе");

int g =Convert.ToInt32( Console.ReadLine());

Hikelist.Add(new Hike(w, g));

hike[i + 1] = w;

}

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

foreach (KeyValuePair < int, string> keyValue in hike)

{

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

}

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

foreach (Hike x in Hikelist)

x.PrintWeather();

Hikelist.RemoveAt(1);

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

Hi cl = (Hi)Hikelist.Clone();

foreach (Hike x in cl) x.PrintWeather();

Console.ReadLine();

}

}

}

Таблица 18.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Топорик, 50, Ножовка, 30, Молоточек, 60 | 1-Топорик  2-Ножовка  3-Молоточек  В походе-Топорик количетво человек-50  В походе-Ножовка количетво человек-30  В походе-Молоточек количетво человек-60  В походе-Топорик количетво человек-50  В походе-Молоточек количетво человек-60 |

Анализ результатов:

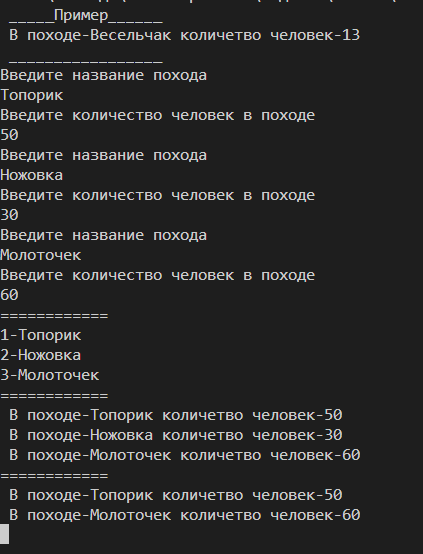


Рисунок 18.3 – Результат работы программы

Источник – собственная разработка