**Задание 18**

**ОБОБЩЕНИЯ**

**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:

static void Main(string[] args)

{

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

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

foreach (char c in s)

{

if (c == '#' && st.Count > 0)

{

st.Pop();

}

else

{

st.Push(c);

}

}

var stroka = new string(st.Reverse().ToArray());

Console.WriteLine(stroka);

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | ac |

Анализ результатов:

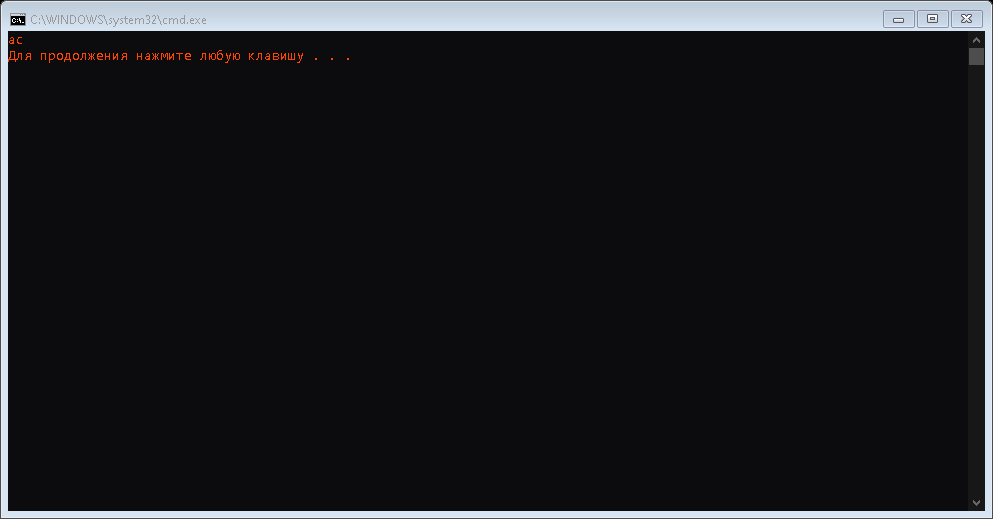


Рисунок 1.1 – Результат работы программы

**Задание №2. Решить следующие задачи с использованием класса Queue.**

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.

Листинг программы:

static void Main(string[] args)

{

StreamReader st = new StreamReader("1.txt");

Queue<string> q1 = new Queue<string>();

Queue<string> q2 = new Queue<string>();

Queue<int> q3 = new Queue<int>();

string f = st.ReadToEnd();

string notNumbers = " ";

string numbers = " ";

foreach (var i in f)

{

if (!char.IsDigit(i))

{

notNumbers += i;

}

else

{

numbers += i;

}

}

q1.Enqueue(notNumbers);

q2.Enqueue(numbers);

Console.WriteLine(String.Join(" ", q1));

Console.WriteLine(String.Join(" ", q2));

}

Таблица 1.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | .  /  ?  1234 |

Анализ результатов:

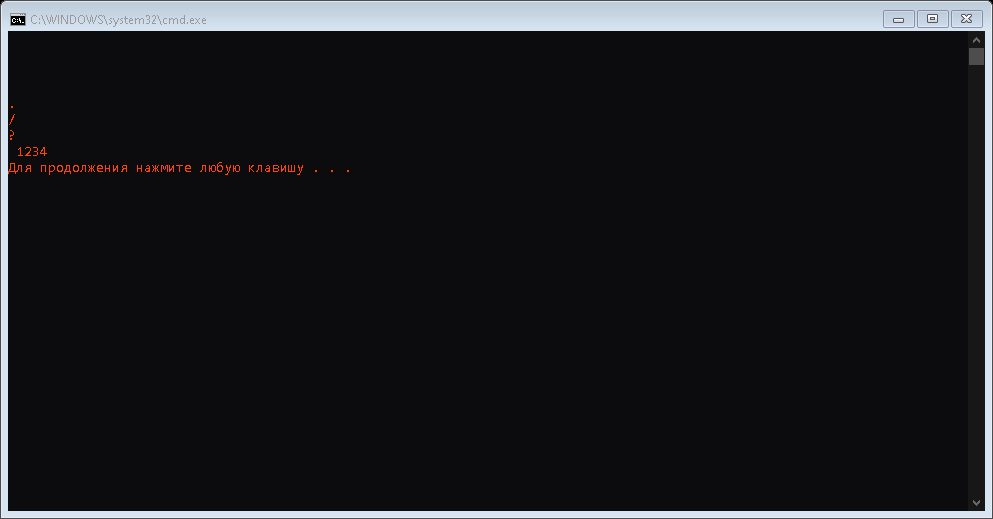


Рисунок 1.2 – Результат работы программы

**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;)**.**

Листинг программы:

static void Main(string[] args)

{

Gorod gd = new Gorod("город", 32, "страна");

InfoObj.Info<Gorod>(gd);

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

Trev TrevList = new Trev();

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

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

{

Console.WriteLine("Введите страну");

string strana = null;

strana = Console.ReadLine();

Console.WriteLine("Введите стоимость");

int stoim = int.Parse(Console.ReadLine());

Console.WriteLine("Введите город");

string gorod = Console.ReadLine();

TrevList.Add(new Gorod(gorod, stoim, strana));

c[i + 1] = gorod;

}

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

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

{

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

}

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

foreach (Gorod x in TrevList)

x.PrintTravel();

TrevList.RemoveAt(1);

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

Trev cl = (Trev)TrevList.Clone();

foreach(Gorod x in cl)

{

x.PrintTravel();

}

Таблица 1.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Польша  120  Гдыня  Россия  90  Москва  Португалия  900  Лисабон | =========================  1-Гдыня  2-Москва  3-Лисабон  ==============  Путешествие в Польша, город Гдыня стоит: 120  Путешествие в Россия, город Москва стоит: 90  Путешествие в Португалия, город Лисабон стоит: 900  -=====================-  Путешествие в Польша, город Гдыня стоит: 120  Путешествие в Португалия, город Лисабон стоит: 900 |

Анализ результатов:

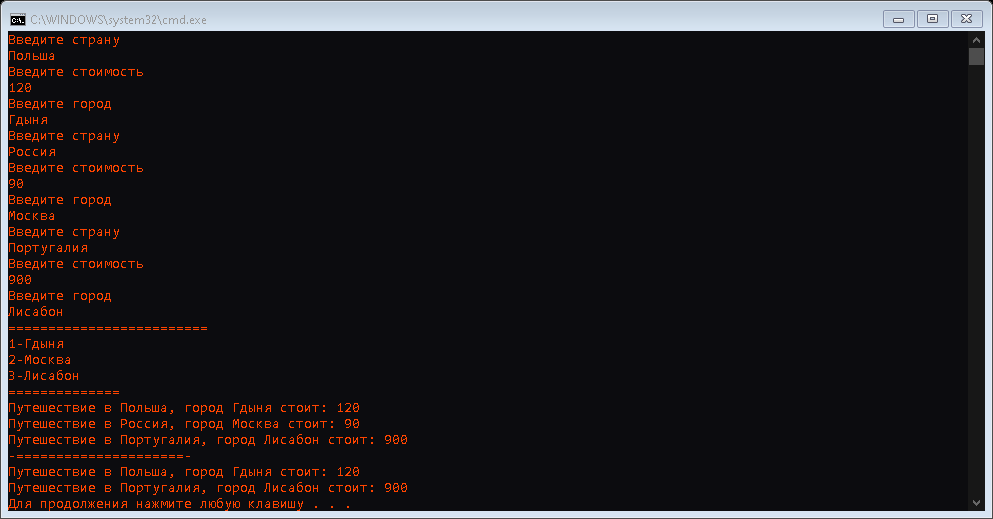


Рисунок 1.3 – Результат работы программы

**Задание №4.** Решить задачу с использованием класса HashTable. Реализовать простейший каталог музыкальных компакт-дисков, который позволяет: добавлять и удалять диски, добавлять и удалять песни, просматривать содержимое целого каталога и каждого диска в отдельности.

Листинг программы:

static void PrintHash(Hashtable ht)

{

ICollection key = ht.Keys;

foreach (var s in key)

{

Console.WriteLine(s + "\t" + ht[s]);

}

}

static void Main(string[] args)

{

Hashtable disk = new Hashtable();

Hashtable song = new Hashtable();

disk.Add(01, "Pop");

disk.Add(02, "Rock");

disk.Add(03, "Metal");

song.Add(01, "PP");

song.Add(02, "KK");

song.Add(03, "JJ");

ICollection key = disk.Keys;

Console.WriteLine("Выберите один из пунктов(1:Добавление диска, 2:Удаление диска,3:Добавление песни,4:Удаление песни,5:Просмотр дисков,6:Просмотр песен");

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

switch (n)

{

case 1:

{

Console.WriteLine("");

disk.Add(Console.ReadLine(), Console.ReadLine());

PrintHash(disk);

break;

}

case 2:

{

Console.WriteLine("Выберите номер диска для удаления");

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

if (num == 1)

{

disk.Remove(1);

}

else if (num == 2)

{

disk.Remove(2);

}

else if (num == 3)

{

disk.Remove(3);

}

else if (num == 4)

{

disk.Remove(4);

}

PrintHash(disk);

break;

}

case 3:

{

song.Add(Console.ReadLine(), Console.ReadLine());

PrintHash(song);

break;

}

case 4:

{

Console.WriteLine("Выберите номер песни для удаления");

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

if (numS == 1)

{

song.Remove(1);

}

else if (numS == 2)

{

song.Remove(2);

}

else if (numS == 3)

{

song.Remove(3);

}

else if (numS == 4)

{

song.Remove(4);

}

PrintHash(song);

break;

}

case 5:

{

PrintHash(disk);

break;

}

case 6:

{

PrintHash(song);

break;

}

default:

{

Console.WriteLine("Не правильный пункт");

break;

}

}

Таблица 1.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5 | 3 Metal  2 Rock  1 Pop |

Анализ результатов:

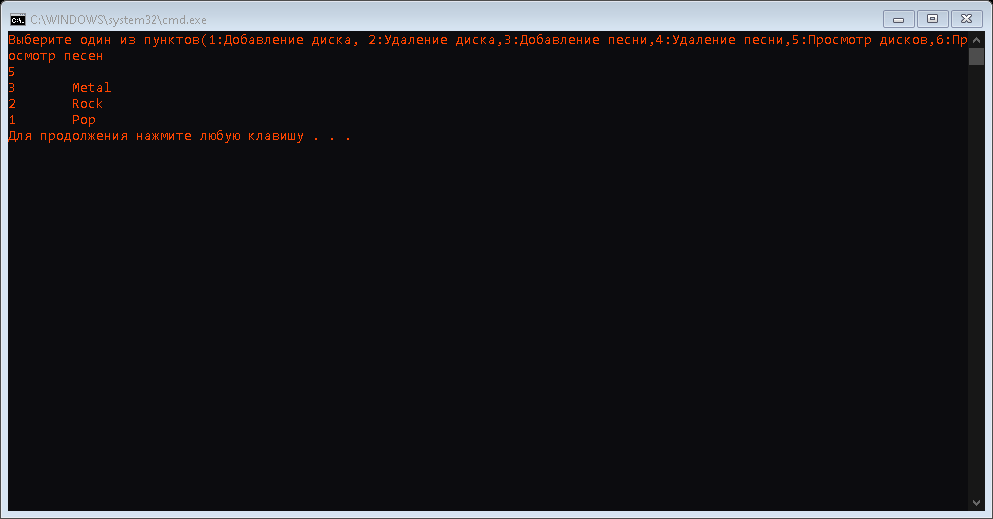


Рисунок 1.4 – Результат работы программы