**18 «Коллекции и 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.

Листинг программы:

using System;

using System.Collections.Generic;

class Program

{

static void Main(string[] args)

{

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

string result = ProcessBackspaces(input);

Console.WriteLine(result);

}

static string ProcessBackspaces(string input)

{

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

foreach (char c in input)

{

if (c == '#')

{

if (stack.Count > 0)

{

stack.Pop();

}

}

else

{

stack.Push(c);

}

}

char[] resultArray = stack.ToArray();

Array.Reverse(resultArray);

return new string(resultArray);

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| abc#d##c | ac |
|  |  |

Анализ результатов:

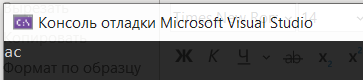


Рисунок 1.1 – Результат работы программы

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.

Листинг класса:

using System;

using System.IO;

using System.Collections.Generic;

class Program

{

static void Main(string[] args)

{

string filePath = "input.txt";

if (!File.Exists(filePath))

{

Console.WriteLine("Файл не найден.");

return;

}

string fileContent = File.ReadAllText(filePath);

string sortedContent = SortContent(fileContent);

Console.WriteLine(sortedContent);

}

static string SortContent(string content)

{

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

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

foreach (char c in content)

{

if (char.IsDigit(c))

{

digitsQueue.Enqueue(c);

}

else

{

lettersQueue.Enqueue(c);

}

}

string sortedContent = "";

while (lettersQueue.Count > 0)

{

sortedContent += lettersQueue.Dequeue();

}

while (digitsQueue.Count > 0)

{

sortedContent += digitsQueue.Dequeue();

}

return sortedContent;

}

}

Таблица 1.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| This is a test string with some numbers: 1234567890  And here are some special characters: !@#$%^&\*() | This is a test string with some numbers:  And here are some special characters: !@#$%^&\*()1234567890 |

Анализ результатов:

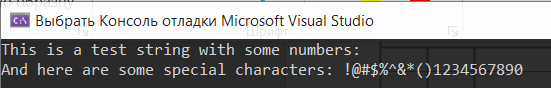


Рисунок 1.2 – Результат работы программы

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 класса:

using System;

using System.Collections.Generic;

class PrototypeClass<T> where T : class, ICloneable

{

private List<T> items;

public PrototypeClass()

{

items = new List<T>();

}

public void AddItem(T item)

{

items.Add(item);

}

public bool RemoveItem(T item)

{

return items.Remove(item);

}

public PrototypeClass<T> Clone()

{

PrototypeClass<T> clone = new PrototypeClass<T>();

foreach (T item in items)

{

clone.AddItem((T)item.Clone());

}

return clone;

}

public List<T> DisplayItems()

{

List<T> displayedItems = new List<T>();

foreach (T item in items)

{

displayedItems.Add(item);

Console.WriteLine(item.ToString());

}

return displayedItems;

}

}

class Person : ICloneable

{

public string Name { get; set; }

public int? Age { get; set; }

public Person(string name, int? age)

{

Name = name;

Age = age;

}

public object Clone()

{

return new Person(Name, Age);

}

public override string ToString()

{

return $"Name: {Name}, Age: {(Age.HasValue ? Age.ToString() : "Unknown")}";

}

}

class Program

{

static void Main(string[] args)

{

PrototypeClass<Person> prototype = new PrototypeClass<Person>();

prototype.AddItem(new Person("John", 30));

prototype.AddItem(new Person("Alice", null));

Console.WriteLine("Original items:");

List<Person> originalItems = prototype.DisplayItems();

PrototypeClass<Person> clone = prototype.Clone();

List<Person> clonedItems = clone.DisplayItems();

clone.RemoveItem(clonedItems[0]);

clone.AddItem(new Person("Bob", 25));

Console.WriteLine("\nCloned items:");

clone.DisplayItems();

}

}

Таблица 1.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Original items:  Name: John, Age: 30  Name: Alice, Age: Unknown  Name: John, Age: 30  Name: Alice, Age: Unknown  Cloned items:  Name: Alice, Age: Unknown  Name: Bob, Age: 25 |

Анализ результатов:

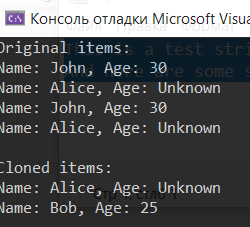


Рисунок 1.3 – Результат работы программы

Задание 4. 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 класса:

using System;

using System.Collections;

class MusicCatalog

{

private Hashtable catalog;

public MusicCatalog()

{

catalog = new Hashtable();

}

public void AddDisk(string diskName)

{

if (!catalog.ContainsKey(diskName))

{

catalog.Add(diskName, new ArrayList());

Console.WriteLine($"Disk '{diskName}' added to the catalog.");

}

else

{

Console.WriteLine($"Disk '{diskName}' already exists in the catalog.");

}

}

public void RemoveDisk(string diskName)

{

if (catalog.ContainsKey(diskName))

{

catalog.Remove(diskName);

Console.WriteLine($"Disk '{diskName}' removed from the catalog.");

}

else

{

Console.WriteLine($"Disk '{diskName}' does not exist in the catalog.");

}

}

public void AddSongToDisk(string diskName, string songName)

{

if (catalog.ContainsKey(diskName))

{

ArrayList songs = (ArrayList)catalog[diskName];

songs.Add(songName);

Console.WriteLine($"Song '{songName}' added to disk '{diskName}'.");

}

else

{

Console.WriteLine($"Disk '{diskName}' does not exist in the catalog.");

}

}

public void RemoveSongFromDisk(string diskName, string songName)

{

if (catalog.ContainsKey(diskName))

{

ArrayList songs = (ArrayList)catalog[diskName];

if (songs.Contains(songName))

{

songs.Remove(songName);

Console.WriteLine($"Song '{songName}' removed from disk '{diskName}'.");

}

else

{

Console.WriteLine($"Song '{songName}' does not exist on disk '{diskName}'.");

}

}

else

{

Console.WriteLine($"Disk '{diskName}' does not exist in the catalog.");

}

}

public void DisplayCatalog()

{

Console.WriteLine("Music Catalog:");

foreach (DictionaryEntry entry in catalog)

{

string diskName = (string)entry.Key;

ArrayList songs = (ArrayList)entry.Value;

Console.WriteLine($"Disk: {diskName}");

Console.WriteLine("Songs:");

foreach (string song in songs)

{

Console.WriteLine($"- {song}");

}

}

}

}

class Program

{

static void Main(string[] args)

{

MusicCatalog catalog = new MusicCatalog();

catalog.AddDisk("Best Hits");

catalog.AddSongToDisk("Best Hits", "Song 1");

catalog.AddSongToDisk("Best Hits", "Song 2");

catalog.AddDisk("Rock Classics");

catalog.AddSongToDisk("Rock Classics", "Song A");

catalog.AddSongToDisk("Rock Classics", "Song B");

catalog.RemoveSongFromDisk("Best Hits", "Song 2");

catalog.RemoveDisk("Rock Classics");

catalog.DisplayCatalog();

}

} Таблица 1.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Disk 'Best Hits' added to the catalog.  Song 'Song 1' added to disk 'Best Hits'.  Song 'Song 2' added to disk 'Best Hits'.  Disk 'Rock Classics' added to the catalog.  Song 'Song A' added to disk 'Rock Classics'.  Song 'Song B' added to disk 'Rock Classics'.  Song 'Song 2' removed from disk 'Best Hits'.  Disk 'Rock Classics' removed from the catalog.  Music Catalog:  Disk: Best Hits  Songs:  - Song 1 |

Анализ результатов:

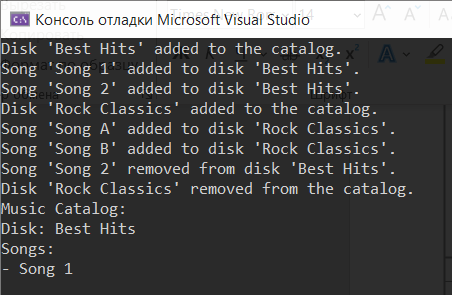


Рисунок 1.4 – Результат работы программы