**18 Коллекции. Классы-прототипы**

Задание №1. Решить задачу 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:

try

{

Console.WriteLine("Введите строку: ");

string str = Console.ReadLine();

var textStack = new Stack<char>();

for (int i = 0; i < str.Length; i++)

textStack.Push(str[i]);

Console.WriteLine(EraseCharacters(textStack));

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

string EraseCharacters(Stack<char> text)

{

int counter = 0;

string str = string.Empty;

while (text.Count > 0)

{

char c = text.Pop();

if (c == '#')

counter++;

else if (counter > 0)

counter--;

else str += c;

}

char[] chars = new char[str.Length];

chars = str.Select(x => x).ToArray();

Array.Reverse(chars);

str = new string(chars);

return str;

}

Таблица 18.1 – Входные и выходные данные

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

Анализ результата разработки библиотеки представлен на рисунке 18.1.



Рисунок 18.1 – Код разработанной библиотеки

Источник: собственная разработка

Задание №2. Решить следующую задачу с использованием класса Queue.

Дан файл, содержащий числа. За один просмотр файла напечатать элементы файла в следующем порядке: сначала все числа, из интервала [a,b], потом все числа, меньшие a, потом все числа, большие b, сохраняя исходный порядок в каждой группе чисел.

Листинг программы:

int a = 2, b = 6;

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

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

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

int number;

using (StreamReader reader = new StreamReader(@"file.txt", Encoding.UTF8))

{

while (!reader.EndOfStream)

{

number = int.Parse(reader.ReadLine());

if (number < a)

lessThenA.Enqueue(number);

else if (number > b)

moreThenB.Enqueue(number);

else

fromAToB.Enqueue(number);

}

}

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

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

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

Таблица 18.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| file.txt | 2, 6, 3, 4, 6, 2, 5  1  8, 8, 9, 8 |

Анализ результатов представлен на рисунке 18.2.



Рисунок 18.2 – Результат работы программы

Источник: собственная разработка

Задание №3. Создать класс-прототип, с обобщенным методом, с методами добавления, удаления, клонирования экземпляров класса «Животные».

В основной программе продемонстрировать:

функциональность созданного класса и всех его методов;

использование обнуляемых типов;

использование абстрактных структур (List<P> Dictionary<Key,Value>).

Листинг программы:

namespace Task3

{

public class Animal

{

public int? Age = null;

public DateTime? BirthDate = null;

public string? Name = null;

public IdInfo? IdInfo = null;

public Animal ShallowCopy()

{

return (Animal)this.MemberwiseClone(); }

public Animal DeepCopy()

{

Animal clone = (Animal)this.MemberwiseClone();

clone.IdInfo = new IdInfo(IdInfo.IdNumber);

clone.Name = Name;

return clone;

}

}

public class IdInfo

{

public int IdNumber;

public IdInfo(int idNumber)

{

this.IdNumber = idNumber;

}

}

class Program

{

static void Main(string[] args)

{

Animal p1 = new Animal();

p1.Age = 42;

p1.BirthDate = Convert.ToDateTime("1977-01-01");

p1.Name = "Jack Daniels";

p1.IdInfo = new IdInfo(666);

Animal p2 = p1.ShallowCopy();

Animal p3 = p1.DeepCopy();

Console.WriteLine("Original values of p1, p2, p3:");

Console.WriteLine(" p1 instance values: ");

DisplayValues(p1);

Console.WriteLine(" p2 instance values:");

DisplayValues(p2);

Console.WriteLine(" p3 instance values:");

DisplayValues(p3);

p1.Age = 32;

p1.BirthDate = Convert.ToDateTime("1900-01-01");

p1.Name = "Frank";

p1.IdInfo.IdNumber = 7878;

Console.WriteLine("\nValues of p1, p2 and p3 after changes to p1:");

Console.WriteLine(" p1 instance values: ");

DisplayValues(p1);

Console.WriteLine(" p2 instance values (reference values have changed):");

DisplayValues(p2);

Console.WriteLine(" p3 instance values (everything was kept the same):");

DisplayValues(p3);

var animals = new Dictionary<int, Animal>();

animals[p1.IdInfo.IdNumber] = p1;

animals[p2.IdInfo.IdNumber] = p2;

animals[p3.IdInfo.IdNumber] = p3;

}

public static void DisplayValues(Animal p)

{

Console.WriteLine(" Name: {0:s}, Age: {1:d}, BirthDate: {2:MM/dd/yy}",

p.Name, p.Age, p.BirthDate);

Console.WriteLine(" ID#: {0:d}", p.IdInfo.IdNumber);

}

}

}

Таблица 18.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 42, 1977-01-01, Jack Daniels, 666 | 32, 1900-01-01, Frank, 7878 |

Анализ результатов представлен на рисунке 18.3.

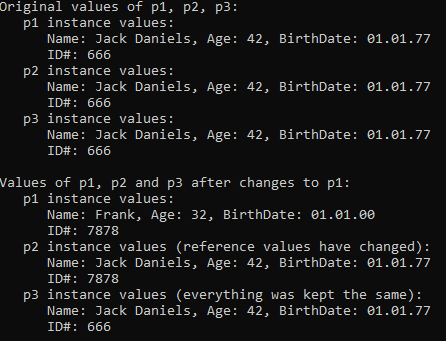
. 

Рисунок 18.3 – Результат работы программы

Источник: собственная разработка

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:

Dictionary<string, Dictionary<string, string>> catalog = new Dictionary<string, Dictionary<string, string>>();

int number = 9;

while (number != 0)

{

try

{

Console.Write("1)Создание диска\n2)Удаление диска\n3)Добавление записи\n4)Удаление записи\n5)Просмотр каталога\n6)Просмотр определенного диска\nВведите '0' для завершения программы\n");

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

if (number == 1)

{

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

Dictionary<string, string> dict = new Dictionary<string, string>();

Console.Write("Введите ключ диска: ");

string keyOfDisk = Console.ReadLine();

catalog.Add(keyOfDisk, dict);

Console.WriteLine("Диск создан");

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

}

else if (number == 2)

{

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

Console.Write("Введите ключ диска, который Вы хотите удалить: ");

string keyOfDiskDDelete = Console.ReadLine();

catalog.Remove(keyOfDiskDDelete);

Console.WriteLine("Диск удален");

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

}

else if (number == 3)

{

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

Console.Write("Введите ключ диска, в который Вы хотите добавить запись: ");

string keyOfDisk = Console.ReadLine();

var dict1 = AddMusic(keyOfDisk);

}

else if (number == 4)

{

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

Console.Write("Введите ключ диска, с которого хотите удалить запись запись: ");

string keyOfDisk = Console.ReadLine();

var dict1 = RemoveMusic(keyOfDisk);

}

else if (number == 5)

{

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

foreach (var cat in catalog)

{

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

Console.WriteLine(cat.Key);

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

foreach (var catI in catalog[cat.Key])

{

Console.Write($"ключ: {catI.Key} ");

Console.WriteLine($"название: {catI.Value}" + "\n");

}

}

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

}

else if (number == 6)

{

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

Console.Write("Введите ключ диска, который Вы хотите просмотреть: ");

string keyOfDisk = Console.ReadLine();

CheckDisk(keyOfDisk);

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

Dictionary<string, string> AddMusic(string keyOfDisk)

{

var dict1 = catalog[keyOfDisk];

Console.Write("Ключ песни: ");

string keyOfMusic = Console.ReadLine();

Console.Write("Название песни: ");

string valOfMusic = Console.ReadLine();

dict1.Add(keyOfMusic, valOfMusic);

Console.WriteLine("Песня добавлена");

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

return dict1;

}

Dictionary<string, string> RemoveMusic(string keyOfDisk)

{

var dict1 = catalog[keyOfDisk];

Console.Write("Ключ песни: ");

string keyOfMusic = Console.ReadLine();

dict1.Remove(keyOfMusic);

Console.WriteLine("Песня удалена");

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

return dict1;

}

void CheckDisk(string keyOfDisk)

{

var dict1 = catalog[keyOfDisk];

foreach (var mus in dict1)

{

Console.WriteLine($"ключ: {mus.Key}");

Console.WriteLine($"название песни: {mus.Value} \n");

}

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

}

Таблица 18.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| keyofDisk, dict1, keyofMusic | catalog |

Источник: собственная разработка

Анализ результатов представлен на рисунке 18.4.

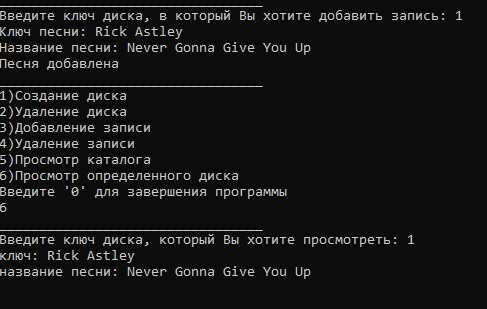


Рисунок 18.4 – Результат работы программы

Источник: собственная разработка