**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("Enter text (symbol # - deleting symbol): ");

string text = Console.ReadLine();

var StackText = new Stack<char>();

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

StackText.Push(text[i]);

Console.WriteLine(DeleteLetters(StackText));

}

catch (Exception e)

{

Console.WriteLine($"An error occured - {e.Message}");

}

string DeleteLetters(Stack<char> text)

{

string textD = string.Empty;

int i = 0;

while (text.Count > 0)

{

char c = text.Pop();

if (c == '#')

i++;

else if (i > 0)

i--;

else textD += c;

}

char[] letters = new char[textD.Length];

letters = textD.Select(x => x).ToArray();

Array.Reverse(letters);

textD = new string(letters);

return textD;

}

Таблица 18.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| abc#d#e | ae |

Источник: собственная разработка

Анализ результатов представлен на рисунке 18.1.

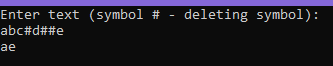


Рисунок 18.1 – Результат работы программы

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 = 5, b = 10, number;

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

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

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

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

{

while (!sr.EndOfStream)

{

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

if (number < a)

Alower.Enqueue(number);

else if (number > b)

Bhigher.Enqueue(number);

else

intervalAB.Enqueue(number);

}

}

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

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

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

Таблица 18.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| file.txt | 5, 8, 9  2, 1, 4, 3  19, 34, 55, 11 |

Источник: собственная разработка

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 «Растения».

В основной программе продемонстрировать:

функциональность созданного класса и всех его методов;

использование обнуляемых типов;

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

Листинг программы:

public class Plant

{

public string? Name = null;

public string? Variety = null;

public string? Reproduction = null;

public IdInfo? IdInfo = null;

public Plant ShallowCopy()

{

return (Plant)this.MemberwiseClone();

}

public Plant DeepCopy()

{

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

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

clone.Name = Name;

return clone;

}

}

public class IdInfo

{

public int ID;

public IdInfo(int identifier)

{

this.ID = identifier;

}

}

class Program

{

static void Main(string[] args)

{

Plant p1 = new Plant();

p1.Name = "Ulva lactuca";

p1.Variety = "Seaweed";

p1.Reproduction = "Vegetative";

p1.IdInfo = new IdInfo(1);

Plant p2 = p1.ShallowCopy();

Plant 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.Name = "Polytrichum commune";

p1.Variety = "Moss";

p1.Reproduction = "disputes";

p1.IdInfo.ID = 2;

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 plants = new Dictionary<int, Plant>();

plants[p1.IdInfo.ID] = p1;

plants[p2.IdInfo.ID] = p2;

plants[p3.IdInfo.ID] = p3;

}

public static void DisplayValues(Plant p)

{

Console.WriteLine($" Plant name: {p.Name}\nVariety: {p.Variety}, Reproduction: {p.Reproduction}");

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

}

}

Таблица 18.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Ulva lactuca, Seaweed, Vegetative, 1 | Polytrichum commune, Moss, disputes, 2 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 18.3.

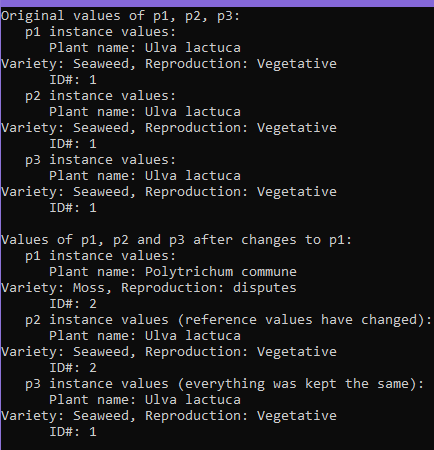
.

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

string choice = " ";

while (choice != "Exit")

{

try

{

Console.Write("1 - Create Disc\n2 - Delete Disc\n3 - Add record\n4 - Delete record\n5 - Show catalog\n6 - Show specific disc\n'Exit' - close program\n");

choice = Console.ReadLine();

if (choice == "1")

{

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

Console.Write("Enter disc key: ");

string DiscKey = Console.ReadLine();

DiscCatalog.Add(DiscKey, KeyDict);

Console.WriteLine("Disc created!\n");

}

else if (choice == "2")

{

Console.Write("Delete disc(Enter key to delete): ");

string DeleteKey = Console.ReadLine();

DiscCatalog.Remove(DeleteKey);

Console.WriteLine("Disk deleted!\n");

}

else if (choice == "3")

{

Console.Write("Add record on disc (Enter disc key): ");

string DiscKey = Console.ReadLine();

var record = AddRecord(DiscKey);

}

else if (choice == "4")

{

Console.Write("Remove record from disc (Enter disc key): ");

string DiscKey = Console.ReadLine();

var record = DeleteRecord(DiscKey);

}

else if (choice == "5")

{

foreach (var catalog in DiscCatalog)

{

Console.WriteLine(catalog.Key);

foreach (var catalogS in DiscCatalog[catalog.Key])

{

Console.Write($"Key: {catalogS.Key} ");

Console.WriteLine($"Name: {catalogS.Value}" + "\n");

}

}

}

else if (choice == "6")

{

Console.Write("Show disc(enter disc key): ");

string DiscKey = Console.ReadLine();

DiskShow(DiscKey);

}

}

catch (Exception e)

{

Console.WriteLine($"Ann error occured - {e.Message}");

}

}

Dictionary<string, string> AddRecord(string keyOfDisk)

{

var Record = DiscCatalog[keyOfDisk];

Console.Write("Record key: ");

string RecordKey = Console.ReadLine();

Console.Write("Record name: ");

string RecordName = Console.ReadLine();

Record.Add(RecordKey, RecordName);

Console.WriteLine("Record added!\n");

return Record;

}

Dictionary<string, string> DeleteRecord(string keyOfDisk)

{

var Record = DiscCatalog[keyOfDisk];

Console.Write("Record key: ");

string RecordKey = Console.ReadLine();

Record.Remove(RecordKey);

Console.WriteLine("Record deleted!\n");

return Record;

}

void DiskShow(string keyOfDisk)

{

var Dshow = DiscCatalog[keyOfDisk];

foreach (var rec in Dshow)

{

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

Console.WriteLine($"Record name: {rec.Value} \n");

}

}

Таблица 18.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| DiskKey, RecordKey, RecordName, | DiskCatalog |

Источник: собственная разработка

Анализ результатов представлен на рисунке 18.4.

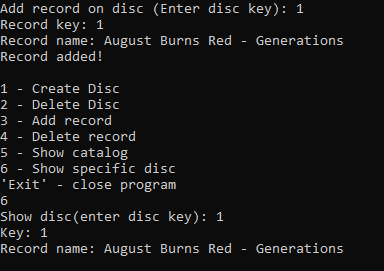


Рисунок 18.4 – Результат работы программы

Источник: собственная разработка