**19. Коллекции и классы прототипы.**

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:

Stack<string> lines = new Stack<string>();

Console.Write("Введите текст: ");

lines.Push(Console.ReadLine());

while(lines.Count != 0)

{

Console.WriteLine(ExecuteBackspaceLit(lines.Pop()));

}

string ExecuteBackspaceLit(string str)

{

while(str.IndexOf('#') != -1)

{

if(str.IndexOf('#') != 0)

{

str = str.Remove(str.IndexOf('#') - 1, 2);

}

else

{

str = str.Remove(str.IndexOf('#'), 1);

}

}

return str;

}

Таблица 19.1 – Входные и выходные данные программы задание №1

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

Анализ результатов:



Рисунок 19.1 – Результат выполнения программы задание №1

Задание №2.

Решить следующие задачи 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:

using System.Text.RegularExpressions;

string path = @"D:\PracticCSharp\Prac19\z2\Files\TextFile1.txt";

int a = Convert.ToInt32(Print("Введите a: "));

int b = Convert.ToInt32(Print("Введите b: "));

List<int> nums = GetNumsInListStr(await GetTextFile(path));

WriteList(nums);

Console.WriteLine("Числа входяшие в диапозон от a до b");

WriteList(FindNumInRange(nums, a, b));

Console.WriteLine("Меньше a:");

WriteList(FindNumsLessNum(nums, a));

Console.WriteLine("Больше b: ");

WriteList(FindNumsGreaterNum(nums, b));

string Print(string mes = "")

{

Console.Write(mes);

return Console.ReadLine();

}

async Task<List<string>> GetTextFile(string path)

{

using (StreamReader reader = new StreamReader(path))

{

string? line;

List<string> lines = new List<string>();

while ((line = await reader.ReadLineAsync()) != null)

{

lines.Add(line);

}

return lines;

}

}

void WriteList<T>(List<T> list)

{

foreach (T elem in list)

{

Console.WriteLine(elem);

}

}

List<int> GetNumsInListStr(List<string> lines)

{

List<int> nums = new List<int>();

Regex regex = new Regex(@"\d+");

foreach(string line in lines)

{

MatchCollection matches = regex.Matches(line);

foreach(var num in matches)

{

nums.Add(Convert.ToInt32(num.ToString()));

}

}

return nums;

}

List<int> FindNumInRange(List<int> nums,int a, int b) =>nums.Where(n => a <= n && n <= b).ToList();

List<int> FindNumsLessNum(List<int> nums, int num) => nums.Where(n => n <= num).ToList();

List<int> FindNumsGreaterNum(List<int> nums, int num) => nums.Where(n => n >= num).ToList();

Таблица 19.2 – Входные и выходные данные программы задание №2

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| a : 2  b : 10  В файле записано:  2 4 2345 235 5 23 5 1 3 2 0 22 24 1  12 56 32 2 45 67 3 23 59 34 1837 45 | Числа входящие в диапазон от a до b  2 4 5 5 3 2 2 3  Меньше a:  2 1 2 0 1 2  Больше b:  2345 235 23 22 24 12 56 32 45 67 23 59 34 1837 45 |

Анализ результатов:

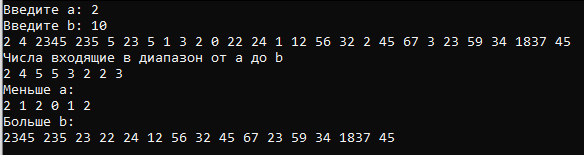


Рисунок 19.2 – Результат выполнения программы задание №2

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<P> Dictionary<Key,Value>).

Листинг программы:

Program:

using z3.Model;

Diary diary = new Diary("MakiMails",

new IdInfo(12412));

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

Console.WriteLine("Shallow copy:");

diary.Add("1 record in diary.");

Diary diary1 = diary.ShallowCopy();

diary1.Add("2 record in diary copy.");

Console.WriteLine(diary1);

Console.WriteLine(diary);

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

Console.WriteLine("Deep copy:");

Diary diary2 = diary.DeepCopy();

Console.WriteLine(diary2);

Console.WriteLine(diary);

diary2.Add("3 record in diary copy.");

Console.WriteLine(diary);

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

Console.WriteLine();

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

Console.WriteLine("Records in diary: ");

WriteDictionary(diary.Records);

Console.WriteLine();

Console.WriteLine("Records in diary1: ");

WriteDictionary(diary1.Records);

Console.WriteLine();

Console.WriteLine("Records in diary2: ");

WriteDictionary(diary2.Records);

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

void WriteDictionary(Dictionary<int,Record> d)

{

foreach (var couple in d)

{

Console.WriteLine($"Key: {couple.Key} Value: \n{couple.Value}");

}

}

Diary:

namespace z3.Model

{

internal class Diary

{

private Dictionary<int, Record> \_records = new Dictionary<int, Record>();

private string \_ownerName;

private IdInfo \_idInfo;

public Dictionary<int, Record> Records { get => \_records; }

public Diary(string ownerName, IdInfo idInfo)

{

\_ownerName = ownerName;

\_idInfo = idInfo;

}

public Diary ShallowCopy() => (Diary)this.MemberwiseClone();

public Diary DeepCopy()

{

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

clone.\_records = new Dictionary<int, Record>();

foreach (var couple in \_records)

{

clone.\_records.Add(couple.Key, couple.Value.DeepCopy());

}

clone.\_ownerName = String.Copy(\_ownerName);

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

return clone;

}

public void Add(string text)

{

DateOnly date = new DateOnly(DateTime.Now.Year, DateTime.Now.Month, DateTime.Now.Day);

Record record = new Record(date, text);

\_records.Add(\_records.Keys.Count + 1, record);

}

public override string ToString()

{

return $"Id: {\_idInfo}\n" +

$"Owner: {\_ownerName}\n" +

$"Count records: {\_records.Count}";

}

}

}

Record:

using static System.Net.Mime.MediaTypeNames;

namespace z3.Model

{

internal class Record

{

private DateOnly \_dateCreate;

private string \_text;

public DateOnly Date { get => \_dateCreate; }

public string Text { get => \_text; }

public Record(DateOnly dateCreate, string text)

{

\_dateCreate = dateCreate;

\_text = text;

}

public Record DeepCopy() => new Record( new DateOnly(\_dateCreate.Year,\_dateCreate.Month,\_dateCreate.Day),

String.Copy(\_text));

public override string ToString()

{

return $"Data: {\_dateCreate}\n" +

$"Text: {\_text}";

}

}

}

IdInfo:

namespace z3.Model

{

internal class IdInfo

{

private int \_idNumber;

public int IdNumber { get => \_idNumber; }

public IdInfo(int idNumber)

{

\_idNumber = idNumber;

}

public override string ToString()

{

return \_idNumber.ToString();

}

}

}

Таблица 19.3 – Входные и выходные данные программы задание №3

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | ------------------  Shallow copy:  Id: 12412  Owner: MakiMails  Count records: 2  Id: 12412  Owner: MakiMails  Count records: 2  ------------------  Deep copy:  Id: 12412  Owner: MakiMails  Count records: 2  Id: 12412  Owner: MakiMails  Count records: 2  Id: 12412  Owner: MakiMails  Count records: 2  ------------------  ==================  Records in diary:  Key: 1 Value:  Data: 20.03.2023  Text: 1 record in diary.  Key: 2 Value:  Data: 20.03.2023  Text: 2 record in diary copy.  Records in diary1:  Key: 1 Value:  Data: 20.03.2023  Text: 1 record in diary.  Key: 2 Value:  Data: 20.03.2023  Text: 2 record in diary copy.  Records in diary2:  Key: 1 Value:  Data: 20.03.2023  Text: 1 record in diary.  Key: 2 Value:  Data: 20.03.2023  Text: 2 record in diary copy.  Key: 3 Value:  Data: 20.03.2023  Text: 3 record in diary copy.  ================== |

Анализ результатов:

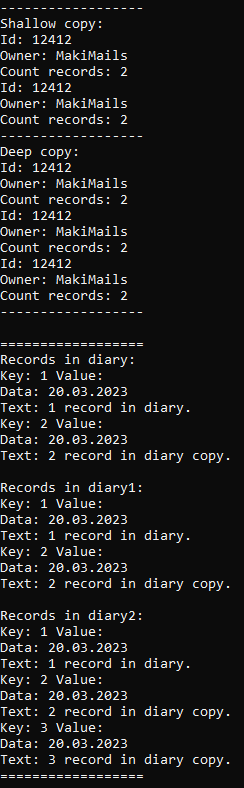


Рисунок 19.3 – Результат выполнения программы задание №3