**18 Коллекции. 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:

static void Main() { Console.Write("Введите строку: ");

string input = Console.ReadLine(); string output = RemoveBackspaces(input);

Console.WriteLine(output);}

static string RemoveBackspaces(string input) {

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

foreach (char c in input){if (c == '#' && stack.Count > 0){stack.Pop();}

else if (c != '#'){stack.Push(c);}} char[] result = new char[stack.Count];

int j = stack.Count; while (stack.Count > 0){result[--j] = stack.Pop();}

return new string(result, 0, result.Length);}

Таблица 18.1 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Медж#вееж##дь | Медведь |

Источник: собственная разработка

Анализ результатов:

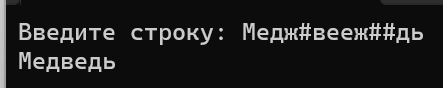


Рисунок 18.1 – Результат работы программы

Источник: собственная разработка

Задание 2. 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:

string filePath = "MyFale.txt"; Console.Write("Введите значение a: ");

int a = int.Parse(Console.ReadLine()); Console.Write("Введите значение b: ");

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

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

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

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

using (StreamReader reader = new StreamReader(filePath))

{while (!reader.EndOfStream){int number = int.Parse(reader.ReadLine());

if (number >= a && number <= b){numbersInRange.Add(number);}

else if (number < a){numbersLessThanA.Add(number);}

else if (number > b){numbersGreaterThanB.Add(number);}}}

Console.WriteLine($"Числа в интервале [{a}, {b}]:");

foreach (int number in numbersInRange){Console.WriteLine(number);}

Console.WriteLine($"Числа меньше {a}:");

foreach (int number in numbersLessThanA) {Console.WriteLine(number);}

Console.WriteLine($"Числа больше {b}:");

foreach (int number in numbersGreaterThanB) {Console.WriteLine(number);}

Таблица 18.2 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3; 5 | Числа в интервале [3, 5]:  3 4 5  Числа меньше 3:  1 2  Числа больше 5:  6 7 8 9 10 11 12 13 14 15 |

Источник: собственная разработка

Анализ результатов:

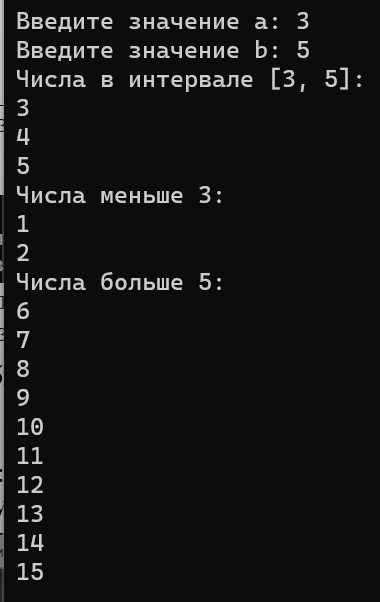


Рисунок 18.2 – Результат работы программы

Источник: собственная разработка

Задание 3. Создать класс-прототип, с обобщенным методом, с методами добавления, удаления, клонирования экземпляров класса. Тема варианта: Планы.

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 PlansProgram{class Program{static void Main(string[] args)

{Plan<string> plans = new Plan<string>();

Console.WriteLine("Введите 3 плана:"); for (int i = 0; i < 3; i++)

{ Console.Write($"План {i + 1}: "); string plan = Console.ReadLine();

plans.AddPlan(plan); } plans.ShowAllPlans();

Console.Write("Введите номер плана для удаления: ");

int index = int.Parse(Console.ReadLine()) - 1; plans.RemovePlan(index);

plans.ShowAllPlans(); Plan<string> clonedPlans = (Plan<string>)plans.Clone();

Console.WriteLine("Все планы из клонированного объекта:");

clonedPlans.ShowAllPlans(); plans.ClearPlans();

Console.WriteLine("Все планы после очистки:"); plans.ShowAllPlans();} }

public class Plan<T> : ICloneable where T : class { private List<T?> plans;

public Plan() { plans = new List<T?>(); }

public void AddPlan(T? plan) { plans.Add(plan);}

public void RemovePlan(int index) { plans.RemoveAt(index);}

public void ClearPlans() { plans.Clear();}

public void ShowAllPlans() {Console.WriteLine("Все планы на сегодня:");

for (int i = 0; i < plans.Count; i++){Console.WriteLine($"{i + 1}. {plans[i]}");}}

public object Clone(){Plan<T> clonedPlan = new Plan<T>(); foreach (T? plan in plans)

{ clonedPlan.AddPlan(plan);} return clonedPlan;}}}

Таблица 18.3 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Где; Живёт; Фазан | Все планы на сегодня:  1. Где  2. Живёт  3. Фазан  Введите номер плана для удаления: 2  Все планы на сегодня:  1. Где  2. Фазан  Все планы из клонированного объекта:  Все планы на сегодня:  1. Где  2. Фазан  Все планы после очистки:  Все планы на сегодня: |

Источник: собственная разработка

Анализ результатов:

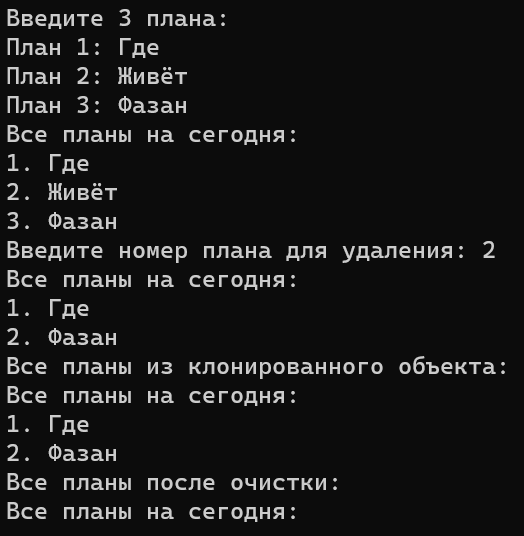


Рисунок 18.3 – Результат работы программы

Источник: собственная разработка