Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЁТ**

**по лабораторной работе №1**

Дисциплина: «Дискретная математика и математическая логика»

Тема: Калькулятор множеств

Выполнил студент ИВТ-22-2б:

Мельников Г. В.

Проверил старший преподаватель кафедры ИТАС:

Рустамханова Г. И.

Пермь 2023

**Постановка задачи**

Разработать программу «Калькулятор множеств», которая будет содержать функционал:

1. Задать универсум (множество целых чисел [-500; 500])
2. Задать множества (A, B, C), используя способы заполнения:

* Случайное заполнение
* По условию

- знак

- чётность

- кратность n

- диапазон

* Вручную

1. Калькулятор множеств, который включает операции:

* Объединение
* Пересечение
* Разность
* Симметрическая разность
* Дополнение

**Методы решения задачи**

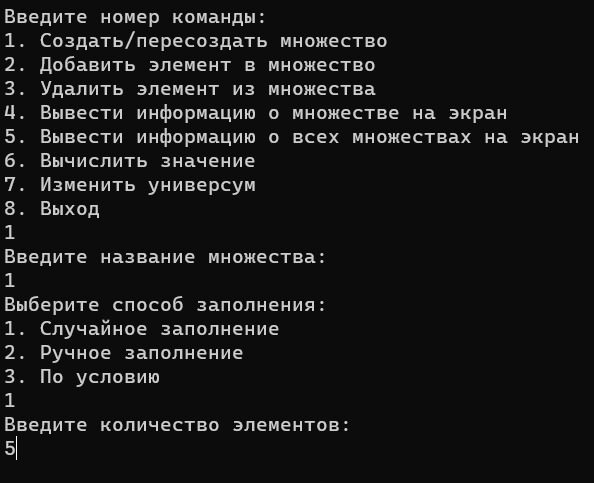
* + - 1. ChooseSetAndElement()//Выбрать множество и элемент
      2. RemoveFromSet()//Удалить из множества
      3. AddToSet()//Добавить во множество
      4. ChooseSet()//Выбрать множество – не обязательно существующее
      5. ChooseExistingSet()//Выбрать существующее множество
      6. CreateSetRandom(string nameOfSet)//Создать множество случайно
      7. CreateSetManually(string nameOfSet)//Создать множество вручную
      8. CreateSetCondition(string nameOfSet)//Создать множество по условию
      9. CreateSet()//Создать множество
      10. WriteSet()//Вывести множество
      11. WriteSets()//Вывести все множества
      12. CalculateValue()//Вычислить выражение
      13. ChangeUniverse()//Изменить универсум

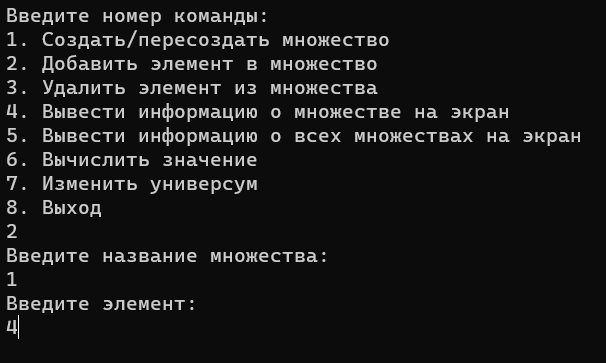
Program.cs является оболочкой (меню и отправка данных для работы с особым типом NumericalSet и хранение множеств в словаре). NumericalSet базируется на HashSet<int>, чтобы не допускать повторы.

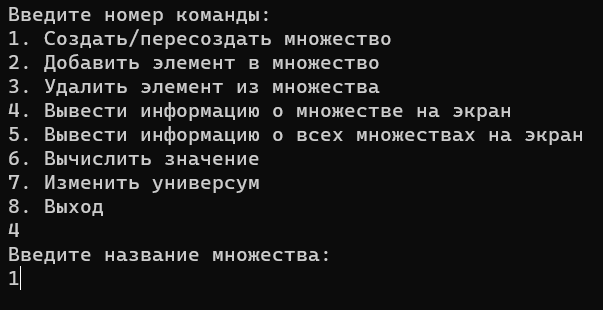
Основные методы NumericalSet.cs:

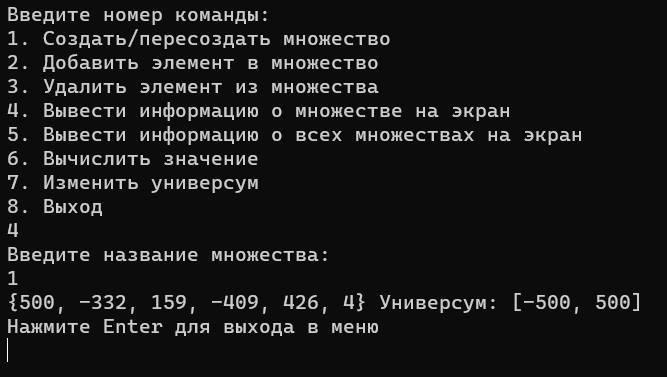
1. Объединение: Создаём результирующее множество и добавляем в него элементы с множеств a и b
2. Пересечение: Проверяем, что элемент множества a присутствует в множестве b, тогда добавляем его в результирующее множеств
3. Разность: Проверяем, что элемент множества a не присутствует в множестве b, тогда добавляем в результирующее
4. Вычисление симметричной разности: Вычисляем с помощью предыдущих методов (a - b) + (a - b)

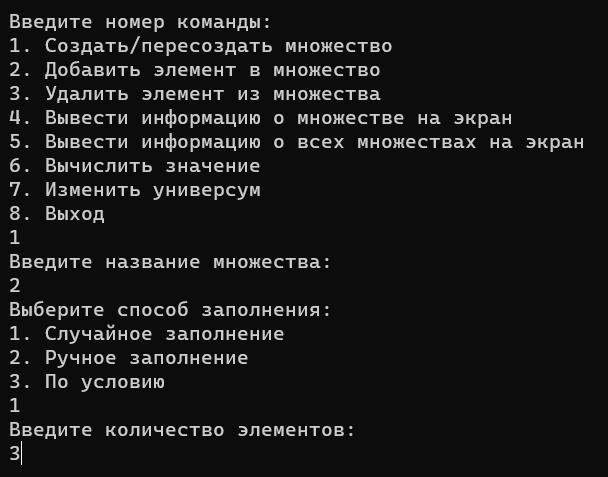
**Результаты**

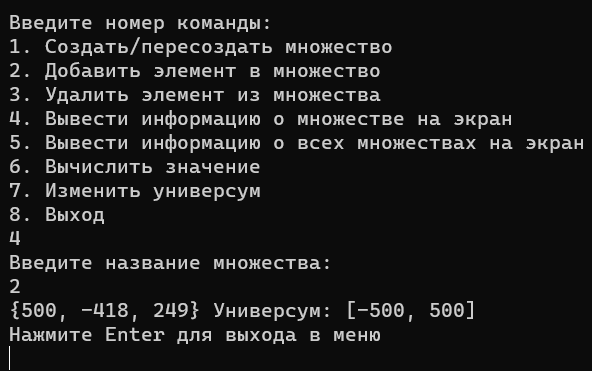


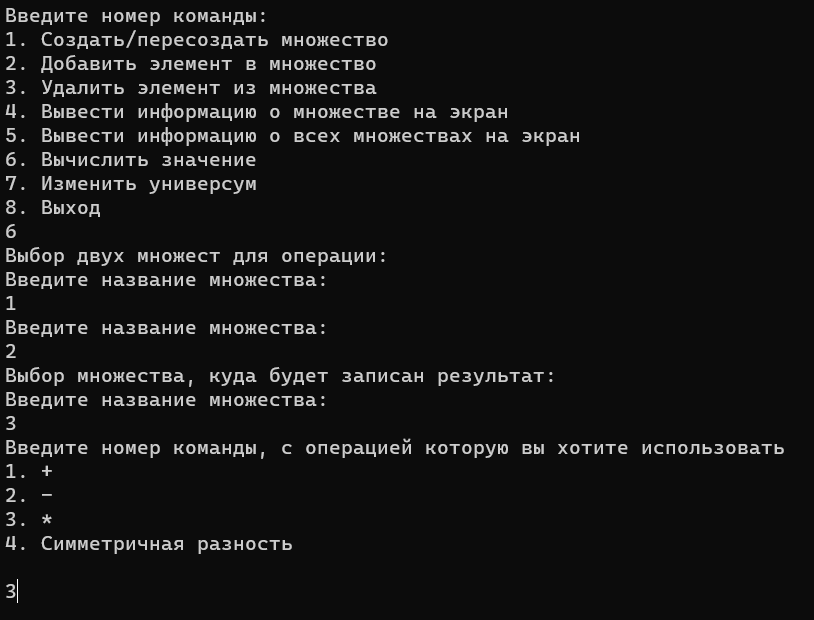


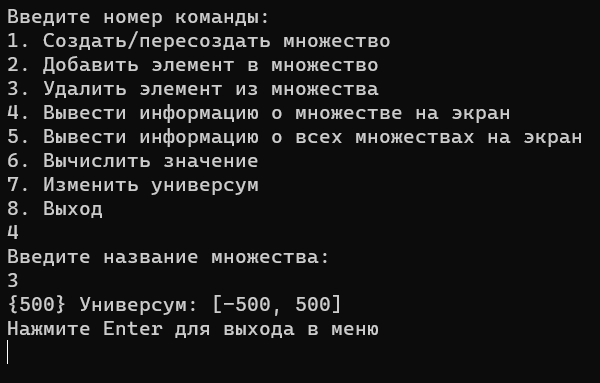












**Код программы**

**Program.cs**

using static UserInput.CustomConsoleInput;

namespace Task

{

internal class Program

{

public static Dictionary<string, NumericalSet> sets = new Dictionary<string, NumericalSet>() { };

public static void Main()

{

var answer = 0;

while (answer != 8)

{

Console.Clear();

var mainMessage = "Введите номер команды:\n" +

"1. Создать/пересоздать множество\n" +

"2. Добавить элемент в множество\n" +

"3. Удалить элемент из множества\n" +

"4. Вывести информацию о множестве на экран\n" +

"5. Вывести информацию о всех множествах на экран\n" +

"6. Вычислить значение\n" +

"7. Изменить универсум\n" +

"8. Выход";

answer = ReadInt(mainMessage, "Неверный ввод! Введите целое число от 1 до 8:", 1, 8);

try

{

switch (answer)

{

case 1:

CreateSet();

break;

case 2:

AddToSet();

break;

case 3:

RemoveFromSet();

break;

case 4:

WriteSet();

break;

case 5:

WriteSets();

break;

case 6:

CalculateValue();

break;

case 7:

ChangeUniverse();

break;

}

}

catch (Exception ex)

{

Console.WriteLine($"\n{ex}\n");

Console.WriteLine("Нажмите Enter для выхода в меню");

Console.ReadLine();

}

}

}

public static (string, int) ChooseSetAndElement()//Выбрать множество и элемент

{

var nameOfSet = ChooseExistingSet();

int element = ReadInt("Введите элемент:", "Неверный ввод! Введите число:");

return (nameOfSet, element);

}

public static void RemoveFromSet()//Удалить из множества

{

(var nameOfSet, var element) = ChooseSetAndElement();

sets[nameOfSet].Remove(element);

}

public static void AddToSet()//Добавить во множество

{

(var nameOfSet, var element) = ChooseSetAndElement();

try

{

sets[nameOfSet].Add(element);

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

Console.WriteLine("Нажмите Enter для выхода в меню");

Console.ReadLine();

}

}

public static string ChooseSet()

{

string nameOfSet = ReadString("Введите название множества:", "Неверный ввод. Введите непустую строку:");

return nameOfSet;

}

public static string ChooseExistingSet()//Выбрать множество

{

string? nameOfSet = "";

bool isCorrect = false;

do

{

nameOfSet = ReadString("Введите название множества:", "Неверный ввод. Введите непустую строку:");

isCorrect = sets.ContainsKey(nameOfSet);

if (!isCorrect)

{

Console.WriteLine("Такого множества не существует! Повторите ввод!");

}

}

while (!isCorrect);

return nameOfSet;

}

public static void CreateSetRandom(string nameOfSet)//Создать множество случайно

{

var answer = ReadInt("Введите количество элементов:", "Неверный ввод! Введите неотрицательное число:", 0);

try

{

sets[nameOfSet] = new NumericalSet().CreateRandom(answer);

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

Console.WriteLine("Нажмите Enter для выхода в меню");

Console.ReadLine();

}

}

public static void CreateSetManually(string nameOfSet)//Создать множество вручную

{

sets[nameOfSet] = new NumericalSet();

var answer = ReadInt("Введите количество элементов:", "Неверный ввод!(Учтите, что колчество элементов должно соответствовать универсуму)" +

"Введите количество элементов:", 0,

sets[nameOfSet].GetEnd() - sets[nameOfSet].GetBegin() + 1);

while (answer-- > 0)

{

var element = ReadInt("Введите элемент:",

$"Неверный ввод! Введите число от {sets[nameOfSet].GetBegin()} до {sets[nameOfSet].GetEnd()}",

sets[nameOfSet].GetBegin(), sets[nameOfSet].GetEnd());

sets[nameOfSet].Add(element);

}

}

public static void CreateSetCondition(string nameOfSet)//Создать множество по условию

{

var mainMessage = "Новое множество будет создано на основе базового.\n" +

"Выберите базовое множество:\n" +

"1. Существующее множество\n" +

"2. Универсум текущего множества";

var answer = ReadInt(mainMessage, "Неверный ввод! Число от 1 до 2:", 1, 2);

NumericalSet BaseSet = new NumericalSet();

if (answer == 1)

{

string nameOfBaseSet = ChooseExistingSet();

BaseSet = sets[nameOfBaseSet];

}

else

{

if (!sets.ContainsKey(nameOfSet))

{

Console.WriteLine("Ошибка! Множества с данным именем не существует! Повторите ввод!");

nameOfSet = ChooseExistingSet();

}

BaseSet = sets[nameOfSet].GetUniverse();

}

mainMessage = "Введите номер команды, содержащий знак,\n" +

"который вы хотитите применить для отборки элементов:\n" +

"1. >\n" +

"2. <\n" +

"3. >=\n" +

"4. <=\n" +

"5. %";

answer = ReadInt(mainMessage, "Неверный ввод! Число от 1 до 5: ", 1, 5);

var element = ReadInt("Введите элемент, сопутствующий знаку", "Неверный ввод! Введите число:");

switch (answer)

{

case 1:

sets[nameOfSet] = BaseSet.Where(x => x > element);

break;

case 2:

sets[nameOfSet] = BaseSet.Where(x => x < element);

break;

case 3:

sets[nameOfSet] = BaseSet.Where(x => x >= element);

break;

case 4:

sets[nameOfSet] = BaseSet.Where(x => x <= element);

break;

case 5:

var ost = ReadInt($"Введите чему должен быть равен остаток от деления элементов множества на {element} ",

"Неверный ввод! Введите число!");

sets[nameOfSet] = BaseSet.Where(x => x % element == ost);

break;

}

}

public static void CreateSet()

{

string nameOfSet = ChooseSet();

var mainMessage = "Выберите способ заполнения:\n" +

"1. Случайное заполнение\n" +

"2. Ручное заполнение\n" +

"3. По условию";

var answer = ReadInt(mainMessage, "Неверный ввод! Введите целое число от 1 до 3:", 1, 3);

switch (answer)

{

case 1:

CreateSetRandom(nameOfSet);

break;

case 2:

CreateSetManually(nameOfSet);

break;

case 3:

CreateSetCondition(nameOfSet);

break;

}

}

public static void WriteSet()

{

var nameOfSet = ChooseExistingSet();

if (sets.ContainsKey(nameOfSet))

{

sets[nameOfSet].WriteInformation();

}

else

{

Console.WriteLine("Такого множества не существует!");

}

Console.WriteLine("Нажмите Enter для выхода в меню");

Console.ReadLine();

}

private static void WriteSets()

{

foreach (var i in sets.Keys)

{

Console.Write($"{i}: " );

sets[i].WriteInformation();

}

Console.WriteLine("Нажмите Enter для выхода в меню");

Console.ReadLine();

}

public static void CalculateValue()//Вычислить выражение

{

Console.WriteLine("Выбор двух множест для операции:");

var a = ChooseExistingSet();

var b = ChooseExistingSet();

Console.WriteLine("Выбор множества, куда будет записан результат:");

var result = ChooseSet();

var mainMessage = "Введите номер команды, c операцией которую вы хотите использовать\n" +

"1. +\n" +

"2. -\n" +

"3. \*\n" +

"4. Симметричная разность\n";

var answer = ReadInt(mainMessage, "Неверный ввод! Число от 1 до 4: ", 1, 4);

try

{

switch (answer)

{

case 1:

sets[result] = sets[a] + sets[b];

break;

case 2:

sets[result] = sets[a] - sets[b];

break;

case 3:

sets[result] = sets[a].Cross(sets[b]);

break;

case 4:

sets[result] = sets[a].CalculateSymmetricDif(sets[b]);

break;

}

}

catch(Exception e)

{

Console.WriteLine(e.Message);

Console.WriteLine("Нажмите Enter для выхода в меню");

Console.ReadLine();

}

}

public static void ChangeUniverse()

{

var nameOfSet = ChooseExistingSet();

int begin = ReadInt("Введите начальную границу:", "Неверный ввод! Введите целое число:");

int end = ReadInt("Введите конечную границу:", "Неверный ввод! Введите конечную границу:");

try

{

sets[nameOfSet].SetUniverse(begin, end);

}

catch(Exception e)

{

Console.WriteLine(e);

Console.WriteLine("Нажмите Enter для продолжения");

Console.ReadLine();

}

}

}

}

NumericalSet.cs

namespace Task

{

public class NumericalSet

{

public HashSet<int> set = new() { };

private int begin = -500, end = 500;

public NumericalSet() { }//Конструктор без параметров

public NumericalSet(int begin, int end) => SetUniverse(begin, end);//Конструктор с параметрами

public NumericalSet(NumericalSet Copied)

{

begin = Copied.begin;

end = Copied.end;

set = Copied.set;

}//Конструктор копирования

private static bool CheckUniverse(NumericalSet first, NumericalSet second)

{

bool isOneUniverse = first.begin == second.begin

&& first.end == second.end;

return isOneUniverse;

}//Проверить универсум

public NumericalSet CreateRandom(int count)

{

if (count >= 0 && end - begin + 1 >= count)

{

var rnd = new Random();

NumericalSet result = new();

for (int i = 0; i < count; i++)

{

var newCount = end;

while (result.set.Contains(newCount))

{

newCount = rnd.Next(begin, end + 1);

}

result.set.Add(newCount);

}

return result;

}

else

{

throw new Exception("Ошибка: нельзя создать множество, задано недопустимое количество элементов");

}

}//Случайно заполнинить

public void Add(int number)

{

if (begin <= number && number <= end)

{

set.Add(number);

}

else

{

throw new Exception("Ошибка: данный элемент нельзя добавить в множество");

}

}//Добавить элемент

public void Remove(int number) => set.Remove(number);//Удалить элемент

public NumericalSet Where(Func<int, bool> predicate)

{

NumericalSet result = new();

foreach (var i in set)

{

if (predicate(i))

{

result.Add(i);

}

}

return result;

}//Отобрать по условию

public static NumericalSet operator +(NumericalSet first, NumericalSet second)

{

if (CheckUniverse(first, second))

{

NumericalSet result = new NumericalSet();

foreach (var i in first.set)

{

result.set.Add(i);

}

foreach (var i in second.set)

{

result.set.Add(i);

}

return result;

}

else

{

throw new Exception("Невозможно выполнить операцию: Множества имеют разные универсумы");

}

}//Перегрузка оператора +

public static NumericalSet operator -(NumericalSet first, NumericalSet second)

{

if (CheckUniverse(first, second))

{

NumericalSet result = new NumericalSet();

foreach (var i in first.set)

{

if (!second.set.Contains(i))

{

result.set.Add(i);

}

}

return result;

}

else

{

throw new Exception("Невозможно выполнить операцию: Множества имеют разные универсумы");

}

}//Перегрузка оператора -

public NumericalSet Cross(NumericalSet second)

{

if (CheckUniverse(second, this))

{

NumericalSet result = new NumericalSet();

foreach (var i in set)

{

if (second.set.Contains(i))

{

result.set.Add(i);

}

}

return result;

}

else

{

throw new Exception("Невозможно выполнить операцию: Множества имеют разные универсумы");

}

}//Пересечь

public NumericalSet CalculateSymmetricDif(NumericalSet second)

{

if (CheckUniverse(second, this))

{

return (this - second) + (second - this);

}

else

{

throw new Exception("Невозможно выполнить операцию: Множества имеют разные универсумы");

}

}//Вычислить симметричрую разность

public void WriteInformation()

{

Console.WriteLine("{" + string.Join(", ", set) + '}' + $" Универсум: [{begin}, {end}]");

}//Вывести информацию о множестве

public NumericalSet GetUniverse()

{

NumericalSet result = new NumericalSet();

result.set = Enumerable.Range(begin, end - begin + 1).ToHashSet();

return result;

}//Получить универсум

public int GetBegin()

{

return begin;

}

public int GetEnd()

{

return end;

}

public void SetUniverse(int begin, int end)

{

if (begin <= end && set.All(t => t >= begin && t <= end))

{

this.begin = begin;

this.end = end;

}

else

{

throw new Exception("Ошибка: заданы недопустимые границы для универсума");

}

}//Задать универсум

public static NumericalSet operator !(NumericalSet first)

{

return first.GetUniverse() - first;

}//Вычислить дополнение

}

}