ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа №1

«Калькулятор множеств»

Выполнил студент группы РИС-23-3Б

Асташин Д. А.

Проверила хз потом

2024 г.

**Цель задачи**

Целью данной задачи является разработка калькулятора для работы с множествами, который позволяет решать любые логические выражения, включающие в себя операции с тремя или менее множествами. Это означает, что пользователь может вводить формулы, содержащие множества A, B, C и использовать стандартные логические операции над ними: пересечение, объединение, разность, симметрическую разность и дополнение.

**Постановка задачи**

1. **Создание множеств:**

* Множества задаются пользователем тремя способами:
* Ввод элементов самостоятельно.
* Заполнение множества с помощью генератора случайных чисел.
* Заполнение множества с условием (например, числа в заданном диапазоне или кратные определенному числу).
* Универсальное множество содержит все целые числа в диапазоне [-100, 100].
* Созданные множества сохраняют свои значения после использования, чтобы их можно было повторно использовать для других операций и выражений.

1. **Ввод логического выражения:**

* Пользователь вводит логическое выражение, содержащее операции над множествами. Выражение может включать следующие операции:
* Пересечение (&) — элементы, общие для обоих множеств.
* Объединение (|) — все элементы, которые принадлежат хотя бы одному из множеств.
* Разность (-) — элементы, присутствующие в одном множестве, но отсутствующие в другом.
* Симметрическая разность (^) — элементы, присутствующие в одном из множеств, но не в обоих одновременно.
* Дополнение (!) — элементы универсального множества, которые не входят в заданное множество.
* Программа вычисляет и выводит результат выражения в виде нового множества.

Задача состоит в том, чтобы реализовать программу, способную создавать, сохранять и выполнять операции над множествами, а также корректно обрабатывать введённые логические выражения и возвращать правильный результат.

**Реализация**

1. Создание множеств:

Множества реализованы в виде списков, которые содержат уникальные элементы в пределах универсального множества (-100 до 100). Для каждого из способов создания множеств реализована отдельная функция:

* CreateAndFillSet(string setName) — заполнение множества пользователем вручную. Пользователь задаёт количество элементов, после чего вводит их по одному.
* CreateRandomSet(string setName) — заполнение множества случайными числами. Пользователь задаёт размер множества, и программа заполняет его случайными уникальными числами в пределах универсального множества (-100 до 100).
* CreateRangeSet(string setName) — заполнение множества числами из определённого диапазона. Пользователь задаёт левую и правую границы диапазона, и программа заполняет множество числами между этими границами.
* CreateMultiplicitySet(string setName) — заполнение множества числами, кратными заданному числу. Пользователь вводит число, и программа заполняет множество элементами, кратными этому числу, из диапазона универсального множества.

В каждую из этих функций передаётся параметр — имя множества (A, B, C), чтобы идентифицировать множество, над которым выполняется операция.

1. Операции:

Каждая операция представлена в виде отдельной функции. Функции принимают списки в качестве параметров и возвращают результаты в виде списков.

* Union(List<int> setA, List<int> setB) — объединяет два множества. Проходит по каждому элементу первого списка и добавляет его в результирующий список, если этот элемент отсутствует. Аналогично обрабатывает второй список.
* Intersection(List<int> setA, List<int> setB) — пересечение двух множеств. Проверяет наличие каждого элемента первого множества во втором и добавляет их в результат, если они присутствуют.
* Difference(List<int> setA, List<int> setB) — разность двух множеств. Добавляет в результат те элементы первого множества, которых нет во втором.
* SymmetricDifference(List<int> setA, List<int> setB) — симметрическая разность. Возвращает элементы, которые содержатся либо в первом, либо во втором множестве, но не в обоих одновременно.
* Complement(List<int> setA) — дополнение. Возвращает все элементы универсального множества, которые отсутствуют в множестве setA.

Таким образом, каждая функция реализует свою операцию в полном соответствии с правилами теории множеств. Каждая из них корректно выполняет обработку элементов и возвращает результат, обеспечивая правильные математические операции для объединения, пересечения, разности, симметрической разности и дополнения. Эти функции позволяют пользователю гибко и эффективно работать с множествами, применяя их к различным задачам и выражениям.

1. Вычисление выражений

Для корректного вычисления логических выражений над множествами необходимо учитывать порядок выполнения операций и правильную расстановку скобок. Для этого были разработаны вспомогательные методы:

* EvaluateExpression(string expression, Dictionary<string, List<int>> sets) — основной метод для вычисления выражения. Он принимает два параметра: строку выражения и словарь множеств (A, B, C). Словарь связывает имена множеств с соответствующими списками чисел.
* ApplyUnaryOperator — метод для обработки унарных операторов, таких как отрицание (!).
* ApplyBinaryOperator — метод для выполнения бинарных операций, например, объединения или пересечения множеств.
* IsOperator — метод, определяющий, является ли символ оператором (например, &, |, !).
* Priority — метод, определяющий приоритет операций, что позволяет правильно обрабатывать скобки и операции с разным уровнем приоритета.

Описание работы метода EvaluateExpression

Метод EvaluateExpression работает по принципу стековой модели. Он считывает выражение символ за символом и добавляет операнды и операторы в два стека:

* operands — стек для хранения множеств (представленных в виде списков).
* operations — стек для хранения операций и скобок.

Когда программа встречает закрывающую скобку ")", она извлекает операнды и операторы из стеков и выполняет соответствующие операции. Если встретился унарный оператор - дополнение, применяется метод ApplyUnaryOperator. Для бинарных операций (например, & для пересечения или | для объединения) используется метод ApplyBinaryOperator.

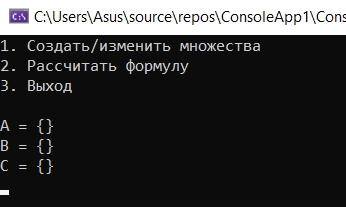
Метод также обрабатывает приоритет операций с помощью функции Priority, что позволяет выполнять действия в правильной последовательности, особенно когда выражение включает скобки.

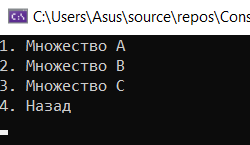
1. Интерфейс

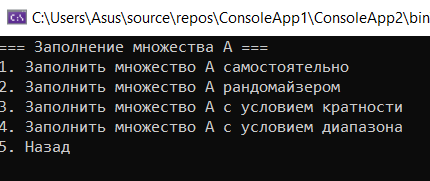
Интерфейс программы выполнен в виде текстового меню, где пользователю предлагаются основные действия: создание множества, вычисление формулы и завершение работы. Пользователь поочерёдно выбирает нужный пункт, взаимодействуя с программой через последовательные запросы.

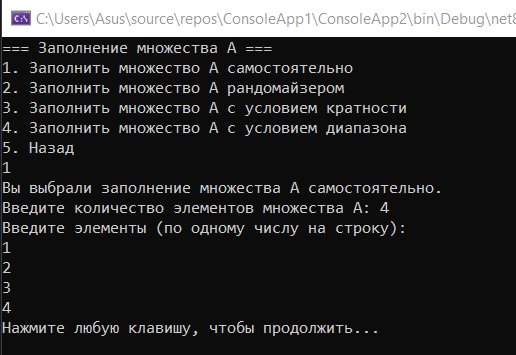
При вводе логического выражения предусмотрена проверка корректности ввода, которая учитывает возможные ошибки. Проверка осуществляется на разных уровнях: проверяется правильность скобок, корректность использования операторов и синтаксические ошибки. Например, программа отслеживает отсутствие недопустимых комбинаций символов (таких как два оператора подряд), незакрытые скобки и некорректное использование операндов (множеств).

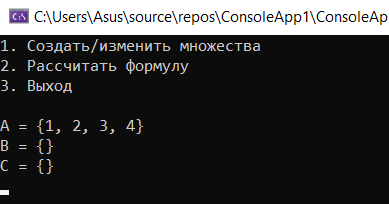
**Результаты работы**

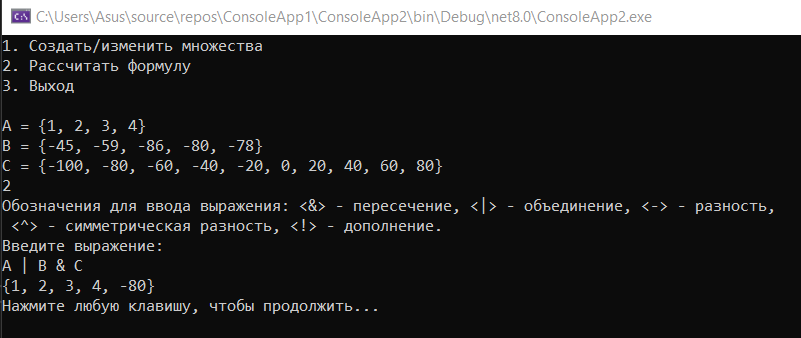
****

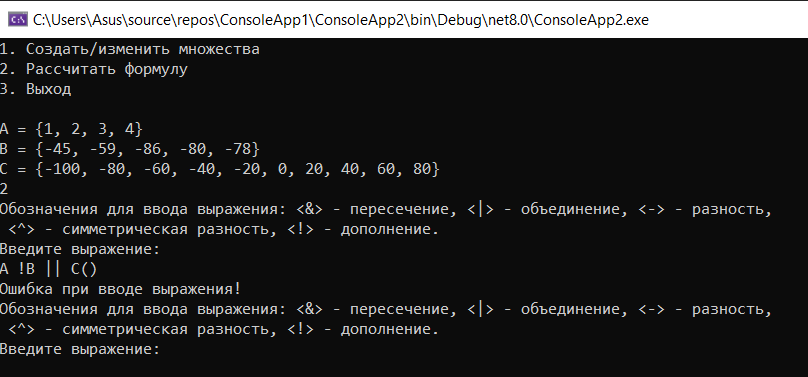
****

****

****

****

****

****

**Вывод**

Таким образом, разработан калькулятор для работы с множествами, который позволяет пользователю создавать и манипулировать наборами данных, а также вычислять результаты логических выражений. Программа поддерживает создание множеств различными способами, выполнение операций над ними, а также обработку сложных выражений с учётом приоритета операций и скобок.

Интерфейс программы в виде текстового меню обеспечивает интуитивное и простое взаимодействие с пользователем, что делает калькулятор доступным и удобным в использовании. Благодаря системе проверок, пользователю предоставляется возможность вводить выражения, не беспокоясь о синтаксических ошибках, поскольку программа заранее выявляет некорректный ввод.

[**Ссылка GitHub**](https://github.com/DimetriusAsti/2simka/tree/main/Set%20Calc)