Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

(ФГАОУ ВО ПНИПУ)

ОТЧЕТ

О ЛАБОРАТОРНОЙ РАБОТЕ

По дисциплине «Дискретная математика»:

РАЗРАБОТКА КАЛЬКУЛЯТОРА МНОЖЕСТВ

Выполнил студент гр. РИС-24-3б

Рунтов Иван Андреевич\_\_\_\_\_\_\_\_\_

Проверил:

Старший преподаватель\_\_\_\_\_\_\_\_\_

Рустамханова Гульшат Ильдаровна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Пермь 2025

**1. Постановка задачи**

1. Разработать консольное приложение калькулятора множеств.

2. Задается минимум три множества. Универсум – целые числа от -50 до 50.

Способы задания множеств:

* случайное заполнение (предусмотреть запрос количества элементов)
* ручной ввод (предусмотреть запрос количества элементов)
* по условия/группе условий (без запроса количества, берутся все элементы из универсума, удовлетворяющие условиям четности, знака, кратности)

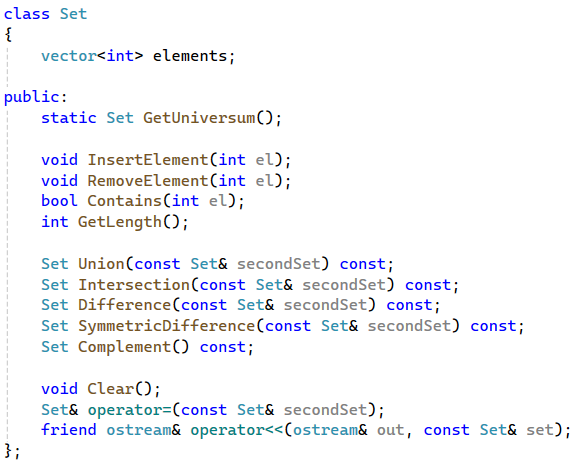
3. Реализовать основные операции над множествами:

* объединение
* пересечение
* дополнение
* разность
* симметрическая разность

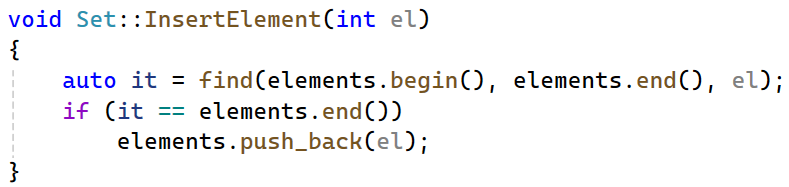
4. Реализовать ввод выражения с множествами.

**2. Описание программы**

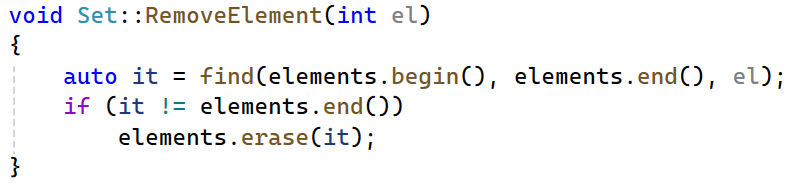
**Класс Set**



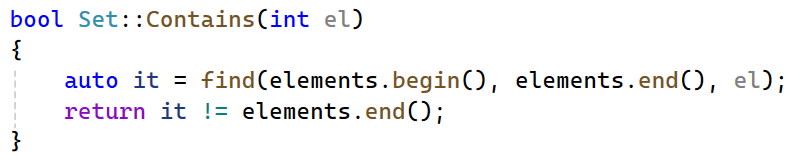
Данный класс является ключевым классом всей программы, он определяет класс множества. Множество основано на векторе.



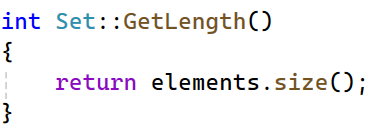
Метод InsertElement предназначен для добавления элемента в множество. Элемент добавляется, только если он уже не содержится.



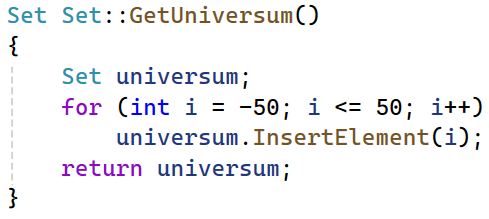
Метод RemoveElement предназначен для удаления элемента из множества. Элемент удаляется, если он содержится в множестве.



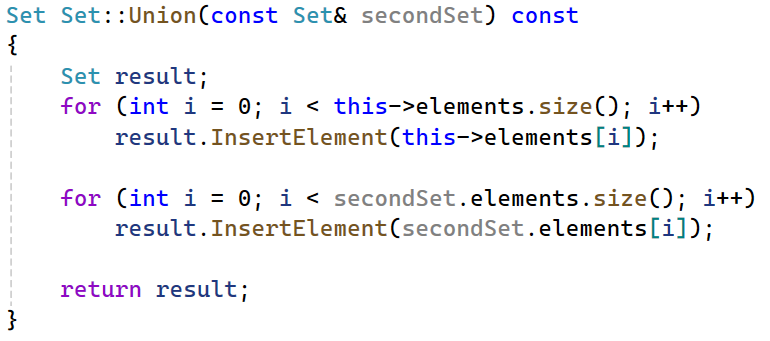
Метод Contains проверяет, содержится ли элемент в множестве.



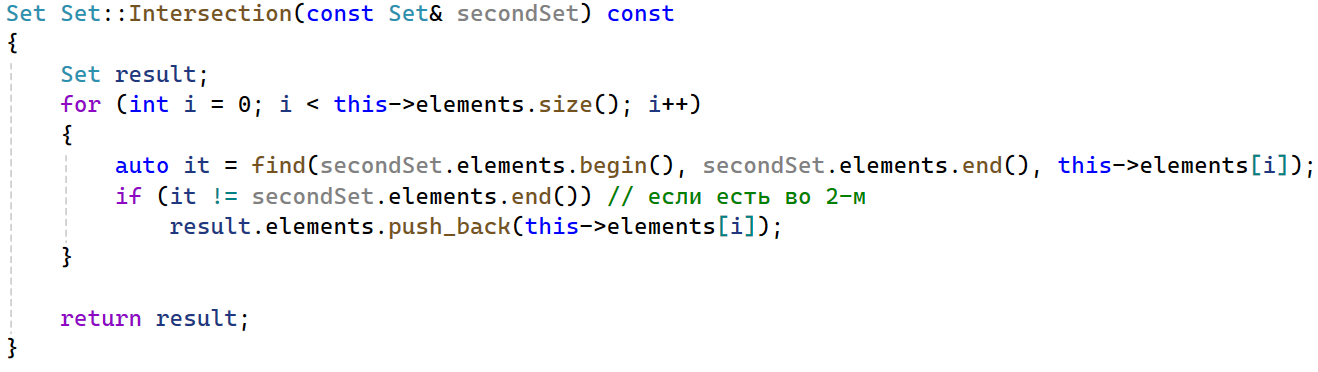
Метод GetLength возвращает размер множества.



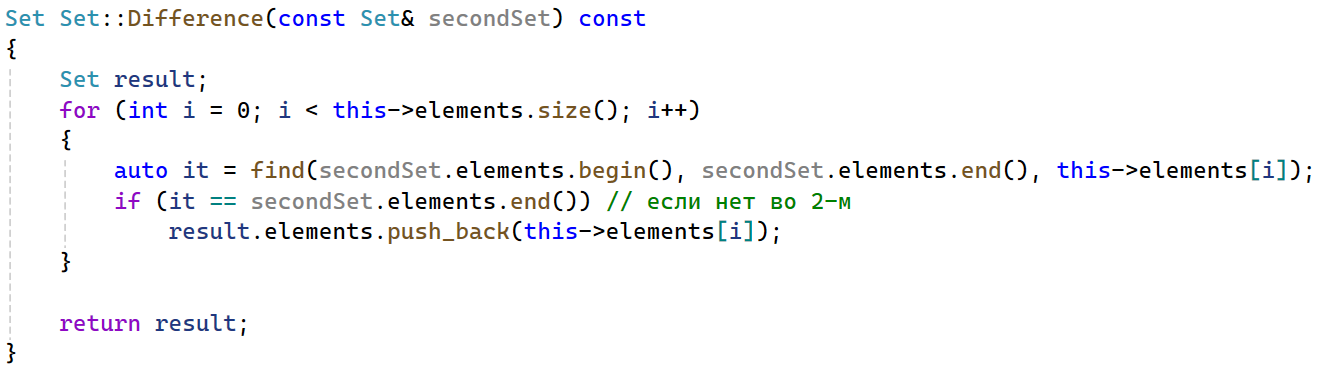
Статический метод GetUniversum создает универсум из заданного диапазона целых чисел.



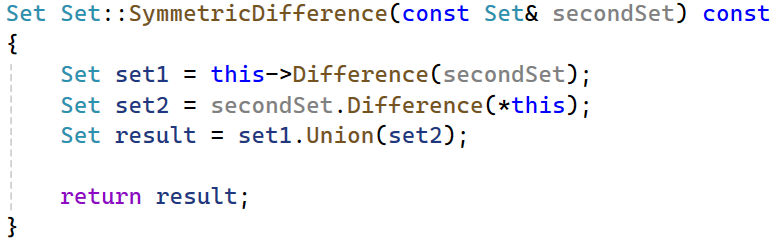
Метод Union предназначен для объединения множеств. Внутри метода создается временное множество, в которое сначала помещаются все элементы первого, а затем второго множества. Элементы не повторяются, так как это предусмотрено методом InsertElement.



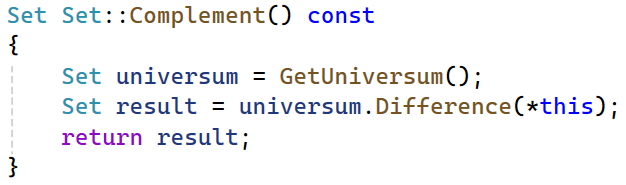
Метод Intersection предназначен для пересечения множеств. Внутри метода создается временное множество. Перебираются все элементы первого множества, проверяется, входят ли они во второе множество. Если элементы входят во второе множество, то добавляются в результирующее множество.



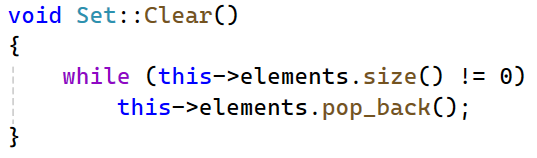
Метод Difference предназначен для разности множеств. Внутри метода создается временное множество. Перебираются все элементы первого множества, проверяется, входят ли они во второе множество. Если элементы не входят во второе множество, то добавляются в результирующее множество.



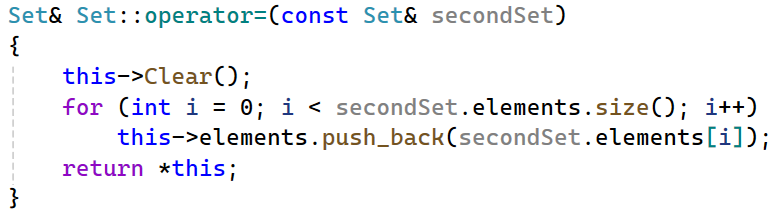
Метод SymmetricDifference предназначен для симметрической разности множеств. Результирующее множество вычисляется как объединение разности первого со вторым и разности второго с первым.



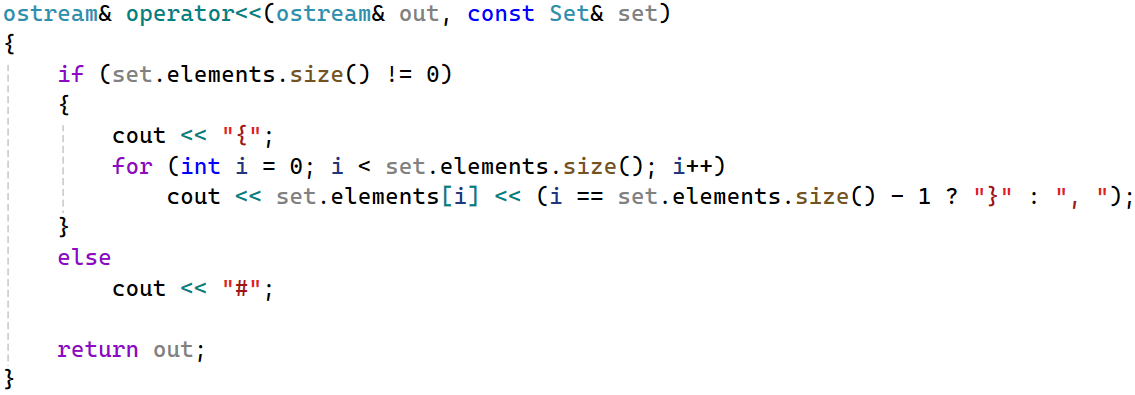
Метод Complement предназначен для вычисления дополнения множества. Результирующее множество вычисляется как разность универсума и текущего множества.



Метод Clear предназначен для очистки множества от всех элементов.

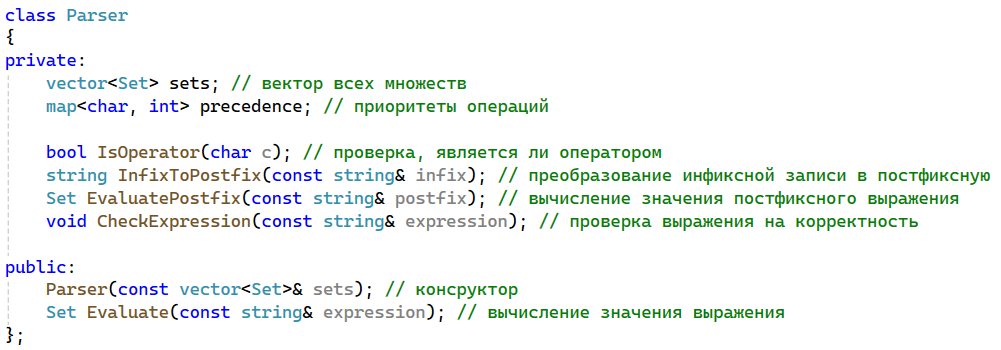


Перегрузка оператора присваивания необходима для того, чтобы можно было присваивать одним объектам класса другие объекты класса.

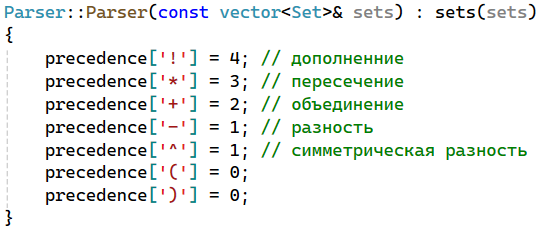


Перегрузка оператора помещения в поток необходима для удобного вывода множества в консоль. Знак «#» означает пустое множество.

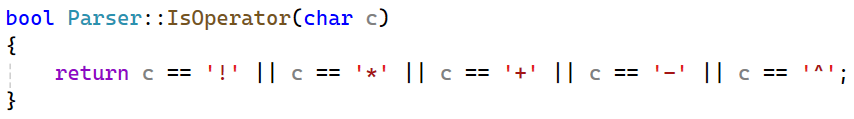
**Класс Parser**



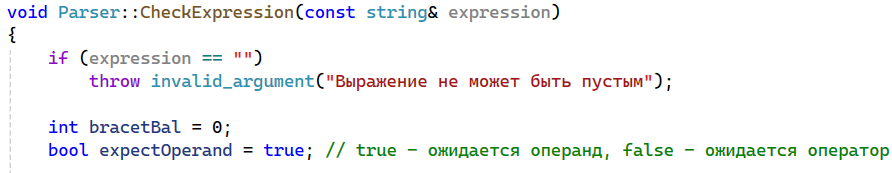
Класс Parser необходим для решения выражения, вводимого пользователем. Данный класс содержит поля вектора множеств и словаря. Вектор множеств содержит все множества, которые могут участвовать в выражении. Словарь содержит приоритеты операций по знакам.



Конструктор принимает вектор множеств, которые могут участвовать в выражении, а также заполняет приоритеты операций.

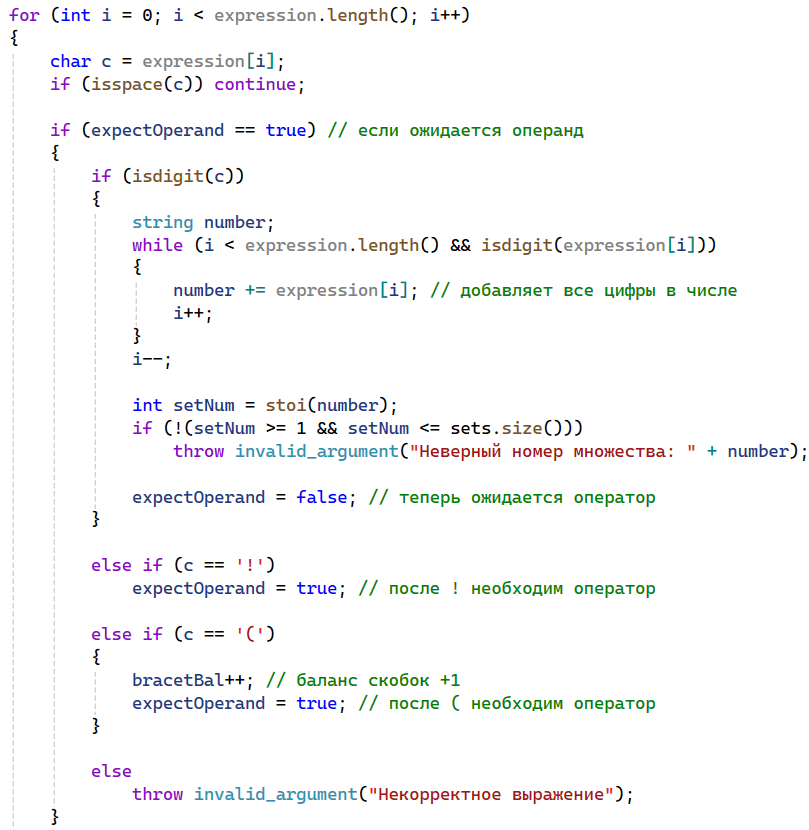


Метод IsOperator проверяет, является ли символ символом оператора.



Метод CheckExpression предназначен для проверки корректности введенного выражения. Метод принимает строку. Если строка пустая, то выводится сообщение, что выражение не может быть пустым.

Вводятся две переменные: bracetBal, необходимая для подсчета скобок, и expectOperand, которая отвечает за то, ожидается следующим операнд или оператор. Изначально expectOperand инициализировано значением true, потому что в начале выражения ожидается операнд.

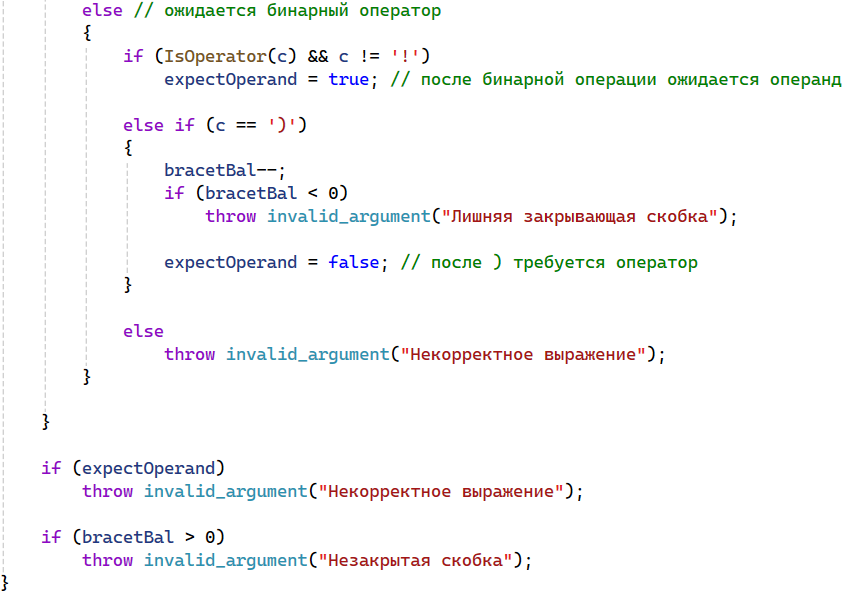


Внутри метода перебираются все символы из введенного выражения. Если символ пробельный, то он пропускается.

Если ожидается операнд, то проверяются случаи на число, поскольку в выражении участвуют номера множеств в векторе множеств. Если встретилось число, то проверяется, входит ли оно в диапазон количества множеств. Если нет, то генерируется исключение на неправильный номер множества. expectOperand изменяется на false, так как после операнда ожидается оператор.

Также, если ожидается операнд, то проверяются случаи на знак дополнения «!» и открывающую скобку «(», так как после них ожидается операнд. В данных случаях expectOperand остается true. Если встречается открывающая скобка, то bracetBal увеличивается на 1.

Если встретилось не число, не знак дополнения «!» и не открывающая скобка «(», то генерируется исключение, что введено некорректное выражение.



Если ожидается оператор, то подразумевается, что ожидается бинарный оператор. Если символ является бинарным оператором, то expectOperand меняется на true, то есть ожидается операнд.

Символ закрывающей скобки «)» не меняет переменной expectOperand, поскольку после закрывающей скобки в выражении идет оператор.

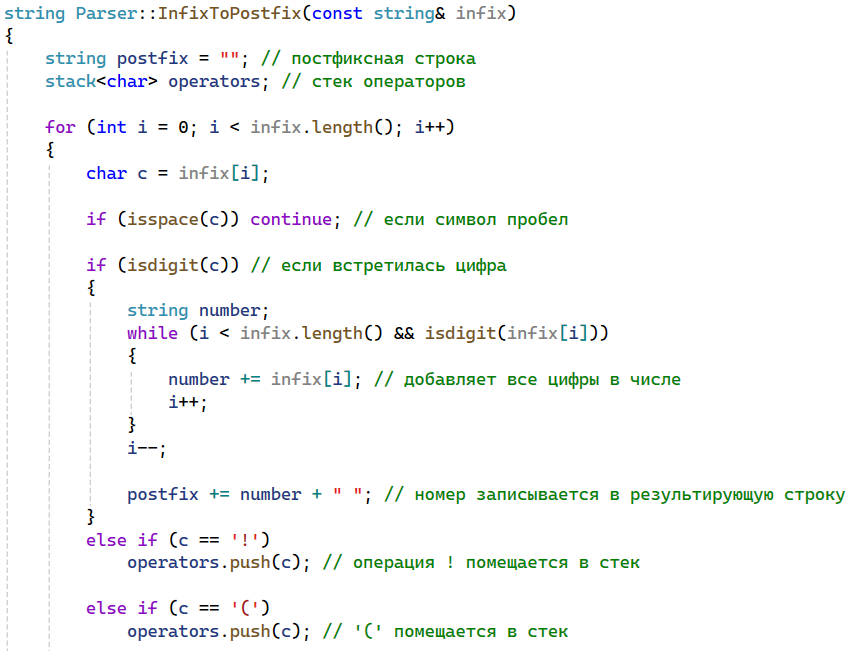
Если баланс скобок становится меньше нуля, то значит, что нашлась лишняя закрывающая скобка.

Если символ не совпал со знаком бинарной операции или знаком закрывающей скобки, то генерируется исключение, что введено некорректное выражение.

После проверки всех символов выражения происходит две проверки.

Если в конце выражения ожидается операнд, то значит, что выражение не завершено, следовательно генерируется исключение о вводе некорректного выражения.

Если bracetBal остается больше 0, то значит, что остались незакрытые скобки. Генерируется исключение, что существует незакрытая скобка.



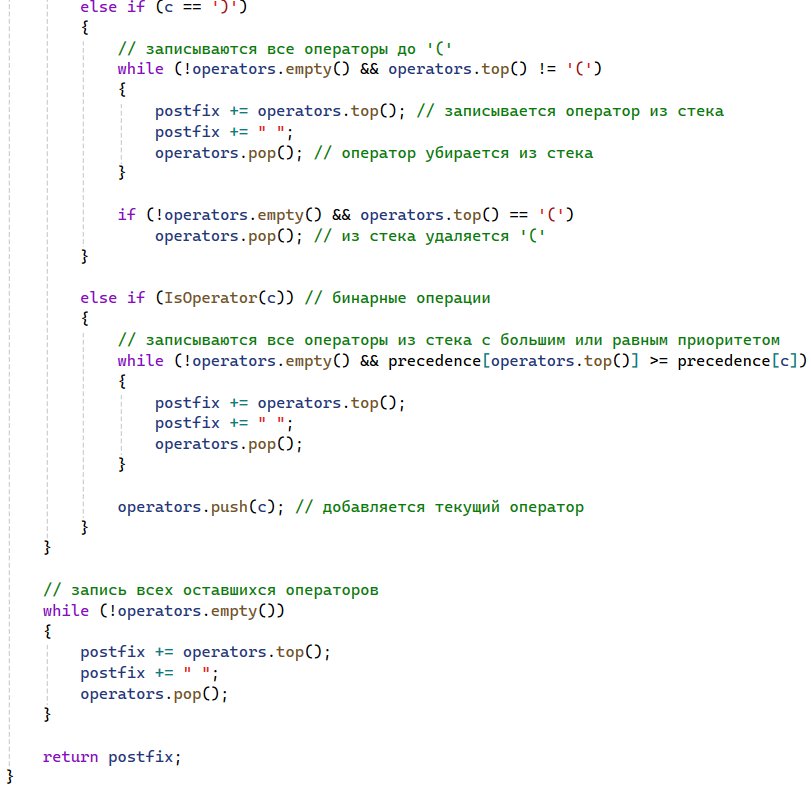
Метод InfixToPostfix принимает в качестве параметра строку, которая будет преобразована из инфиксной записи, удобной для восприятия человеком, к постфиксной записи, удобной для обработки компьютером.

Внутри метода определяются строка postfix – результирующая строка и стек operators, в который будут помещаться встречаемые операторы.

Перебираются все символы строки infix, пробельные символы пропускаются.

Если в инфиксной записи встречается число, то оно просто записывается в результирующую строку.

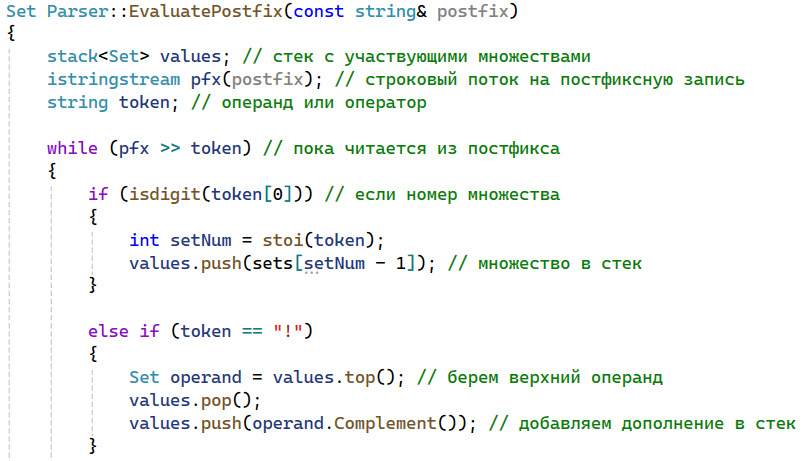
Если встречается операция дополнения или открывающая скобка, то они просто добавляются в стек.



Если встречается закрывающая скобка, то в строку записываются все операторы, уже находящиеся в стеке, пока не встретится открывающая скобка. Когда встречается открывающая скобка, то она удаляется из стека.

Если встречается символ бинарного оператора, то в результирующую строку записываются все операторы из стека, которые обладают большим или равным приоритетом. Это необходимо, чтобы выражение сохранило порядок выполнения операций. После этого в стек добавляется встреченный оператор.

После прохода по всем символам исходной строки, в результирующую строку записываются все операторы, оставшиеся в стеке.

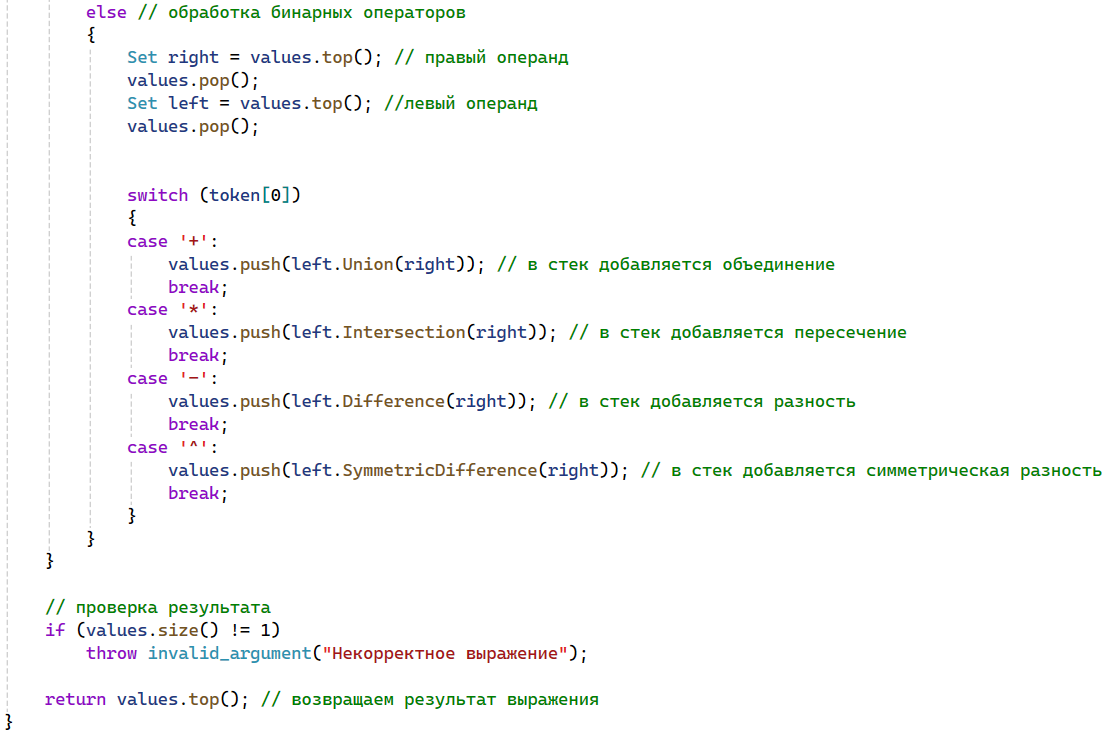


Метод EvaluatePostfix предназначен для вычисления выражения из постфиксной формы. В методе определяются стек множеств, строковый поток на постфиксную строку и строка token, которая определяет элемент постфиксной строки.

Выполняется цикл, пока в строке есть элементы.

Если встретилось число, то в стек добавляется множество из вектора множеств, находящееся под номером данного числа.

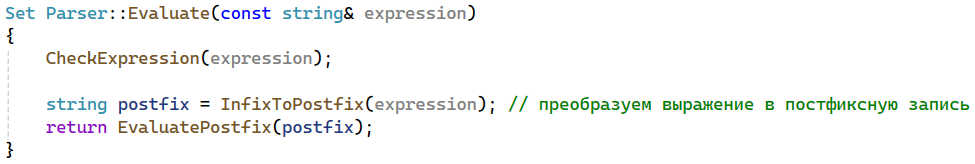
Если встретился знак дополнения «!», то из стека берется верхний операнд, удаляется из стека, а в стек кладется дополнение данного множества.



Если встречается знак бинарного оператора, то из стека достаются два верхних множества: сначала правый операнд, а потом левый операнд. Они удаляются из стека. В стек добавляется результирующее множество, соответствующее знаку операции.

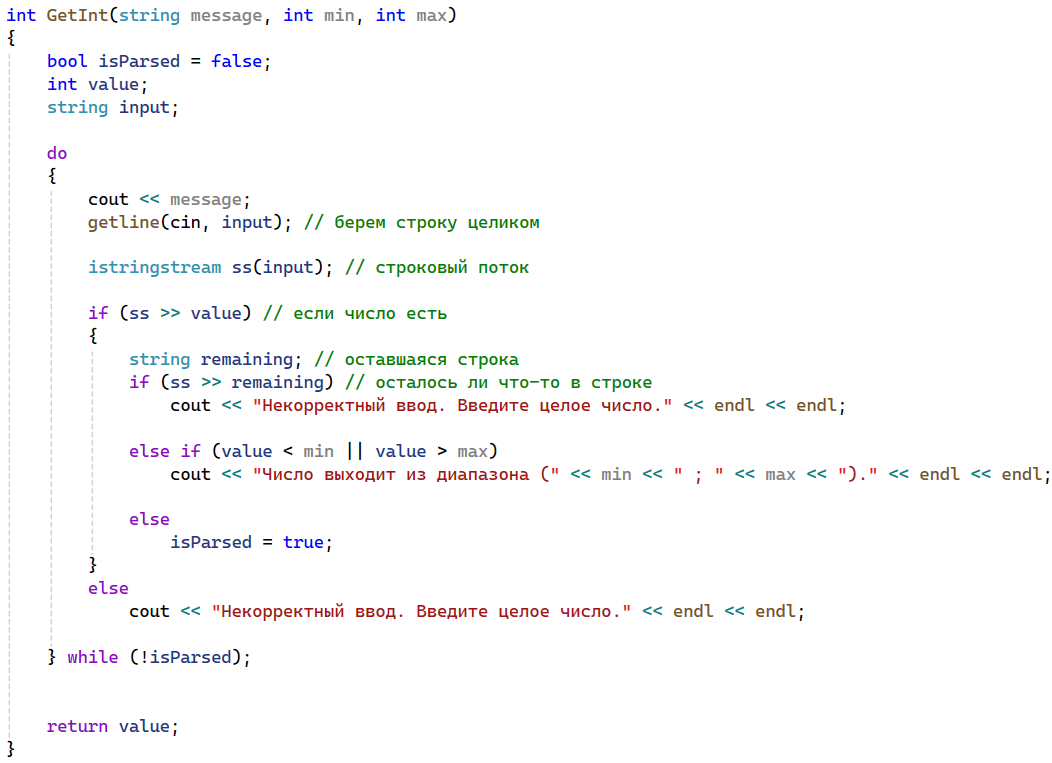
Если в конце вычислений в стеке остается больше одного множества, то генерируется исключение, что выражение некорректно.

Если все отработало успешно, то возвращается множество из стека.

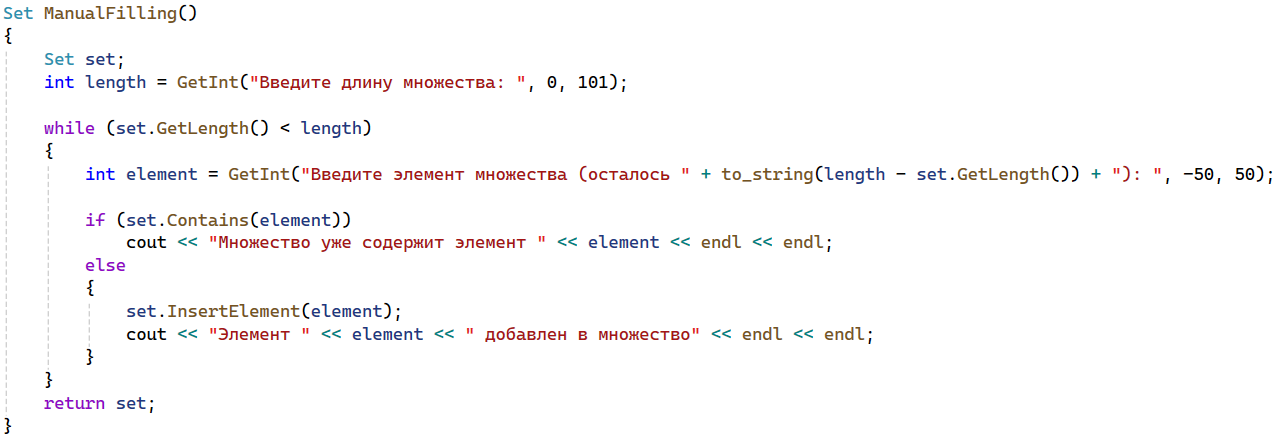


Метод Evaluate является главным методом класса Parser. Метод принимает строку выражения. Строка выражения проверяется на корректность, преобразуется в постфиксную форму. После этого рассчитывается результирующее множество.

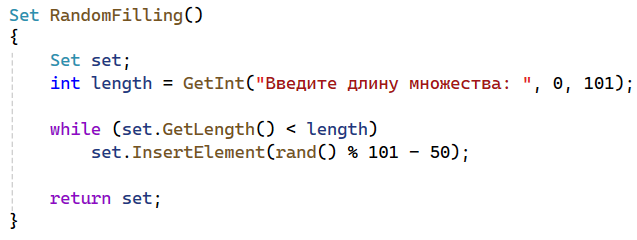
**Основной исполняемый код**



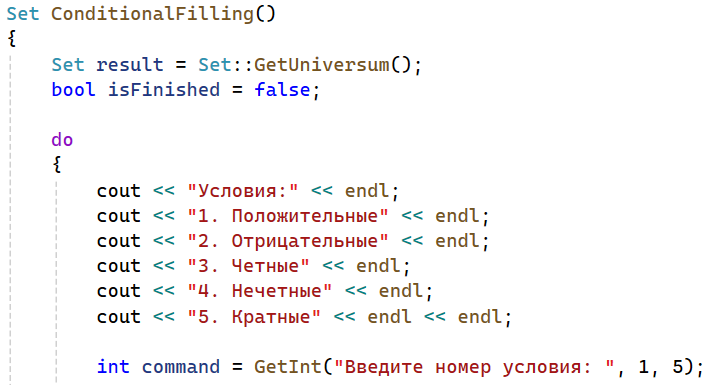
Функция GetInt предназначен для ввода целого числа с клавиатуры в пределах заданного диапазона.

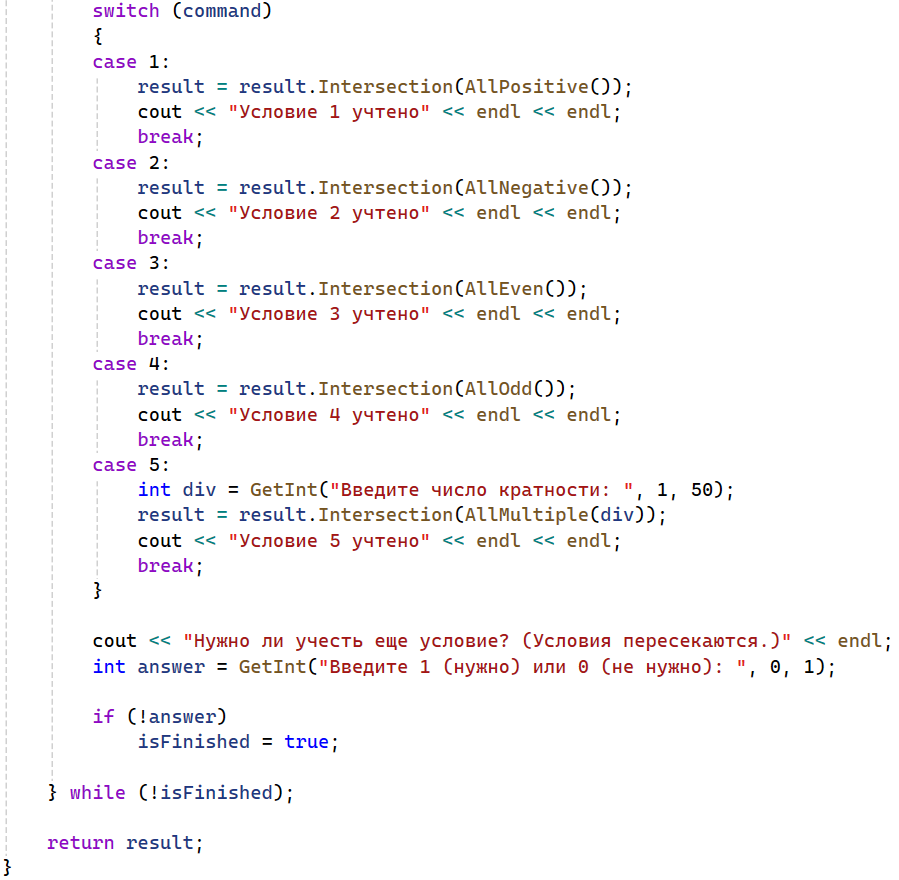


Функция ManualFilling предназначена для ручного ввода множества. Вводится длина множества. А далее заданное количество элементов, если элемент уже есть в множестве или только что добавился, то выводится сообщение.

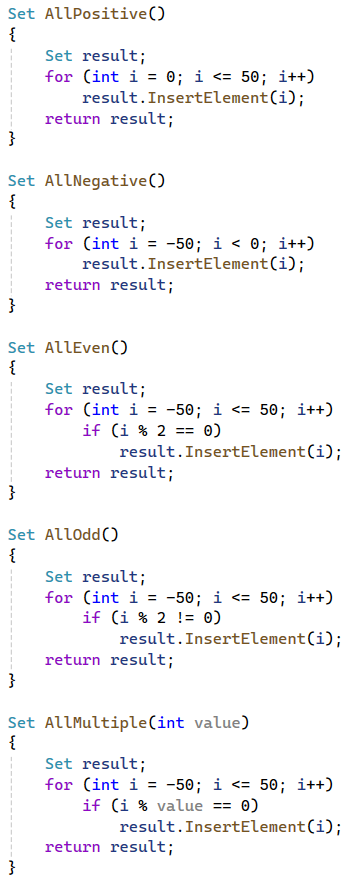


Функция RandomFilling предназначена для случайного заполнения множества. Вводится длина множества.

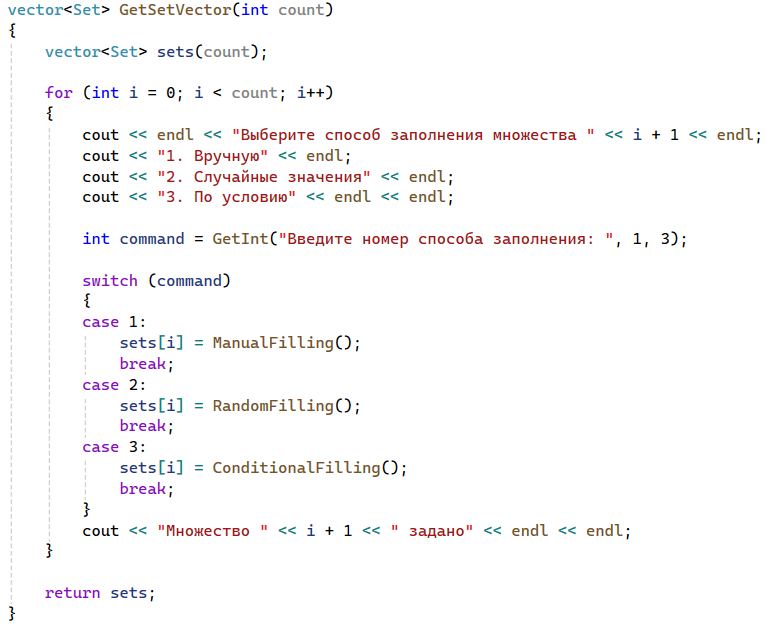




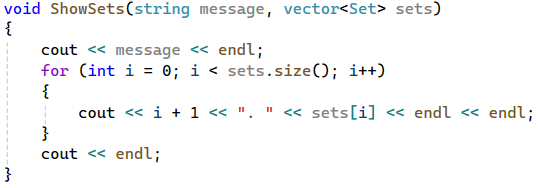
Функция ConditionalFilling предназначена для заполнения множества по условию. Результирующим множеством будет пересечение универсума с условно заданными множествами. Условия учитываются многократно, при этом они пересекаются.



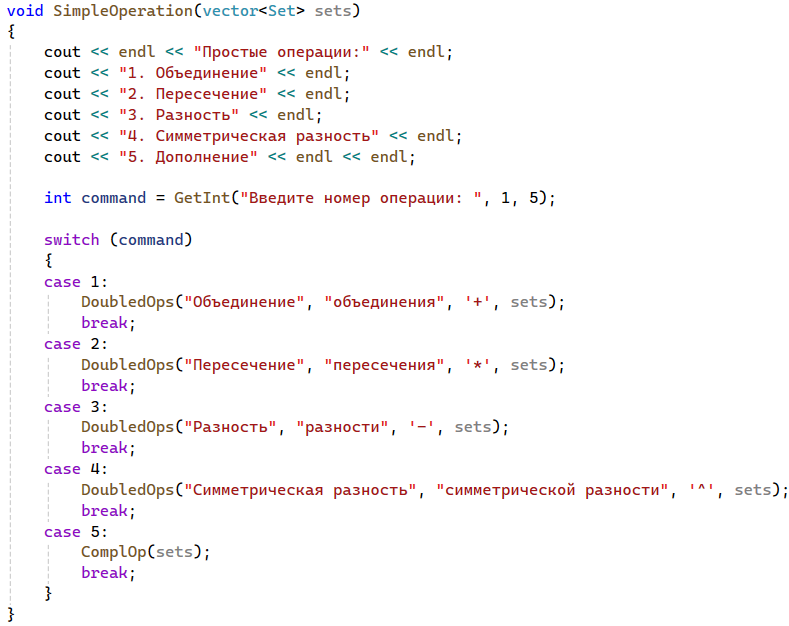
AllPositive возвращает множество всех положительных чисел, AllNegative – всех отрицательных чисел, AllEven – всех четных чисел, AllOdd – всех нечетных чисел и AllMultiple – всех чисел, кратных какому-либо числу.



Функция GetSetVector используется для создания вектора множеств с помощью различных способов заполнения. Функция принимает в качестве параметра количество множеств.



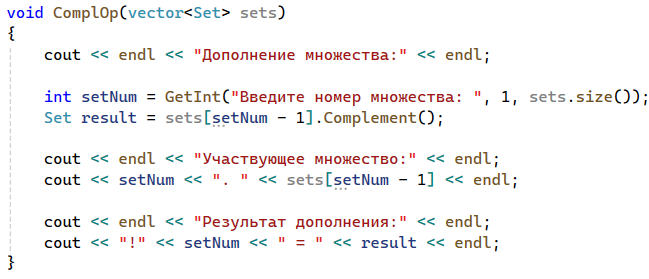
Функция ShowSets используется для отображения множеств из вектора в консоль.



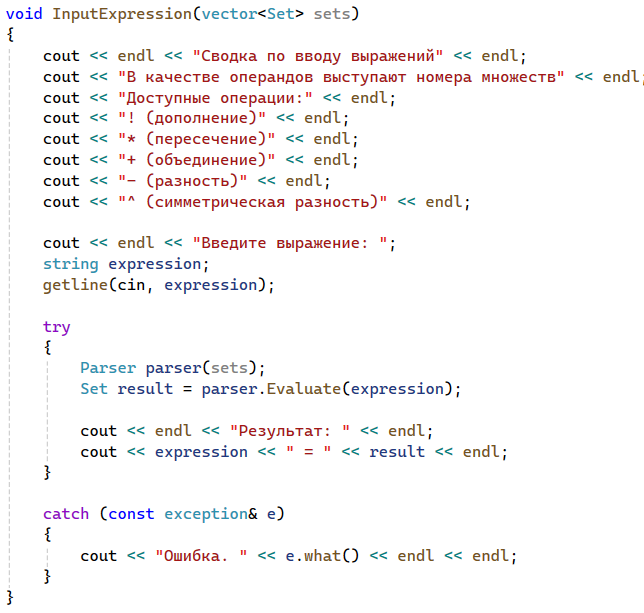
Функция SimpleOperation позволяет получить результат выполнения одной операции с одним или двумя множествами. Задается номер команды с помощью GetInt и выбирается соответствующая операция.



Функция DoubledOps предназначена для расчета бинарных операций с двумя множествами. Функция принимает две строки с названием операции, символ операции и вектор множеств. Запрашиваются номера участвующих множеств и по символу операции рассчитывается результирующее множество.

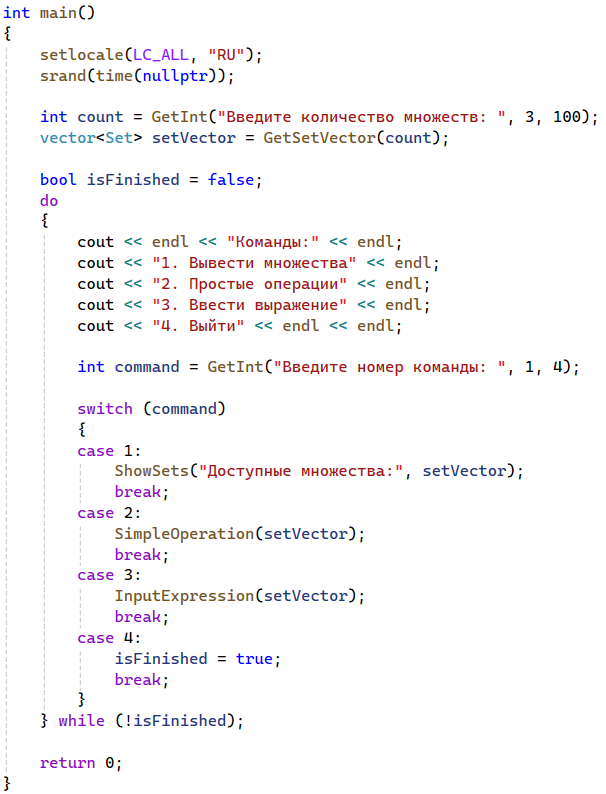


Функция ComplOp предназначена для расчета дополнения. Она принимает только вектор множеств. Запрашивается номер множества и выводится результат дополнения.



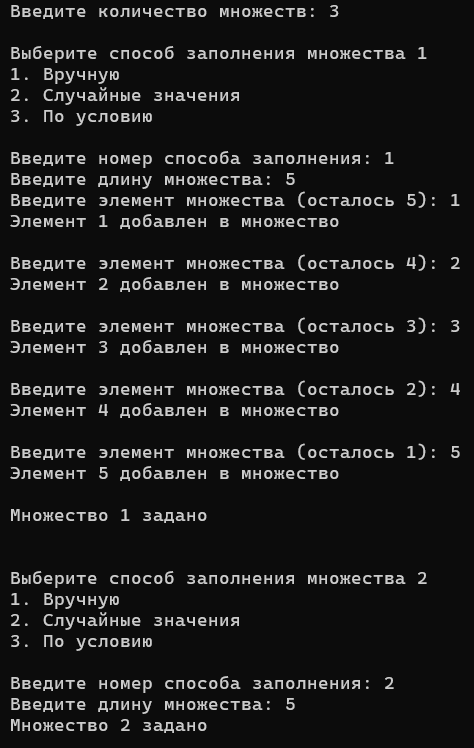
Функция InputExpresion предназначена для ввода выражения. Она принимает вектор множеств. Считывается выражение в строку. В контролируемом блоке try создается объект класса Parser. В результирующее множество записывается результат метода Evaluate объекта parser.

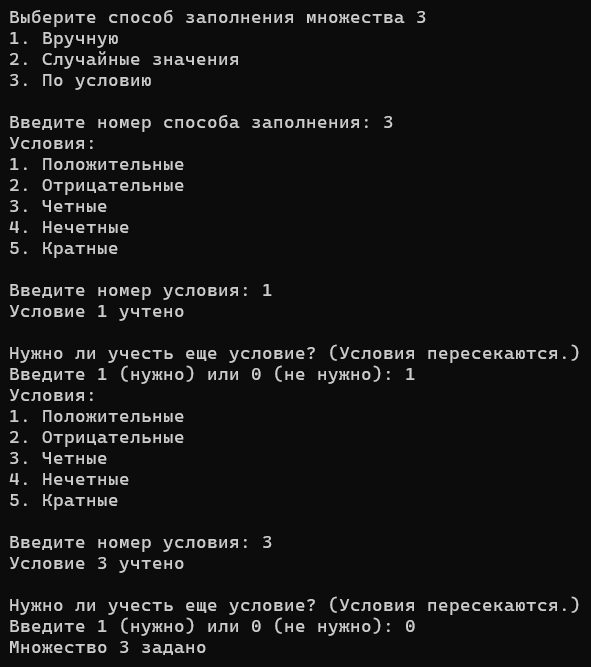
Если выражение задано неверно, то происходит обработка ошибки с выводом сообщения.



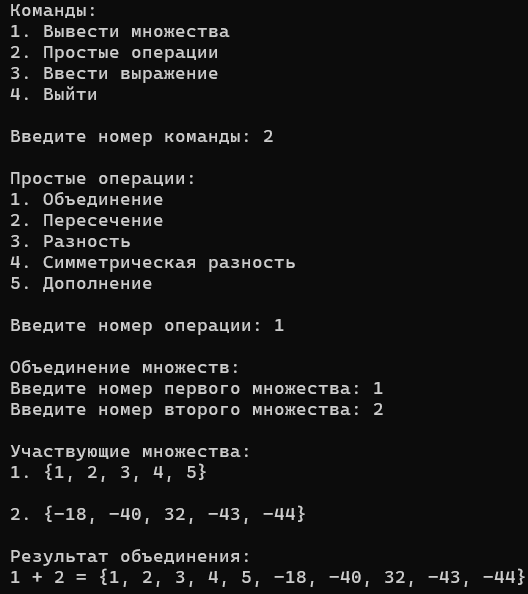
В основной функции main запрашивается количество множеств, множества заполняются. Далее запрашивается номер команды, и выполняется соответствующая функция, или выполнение программы завершается.

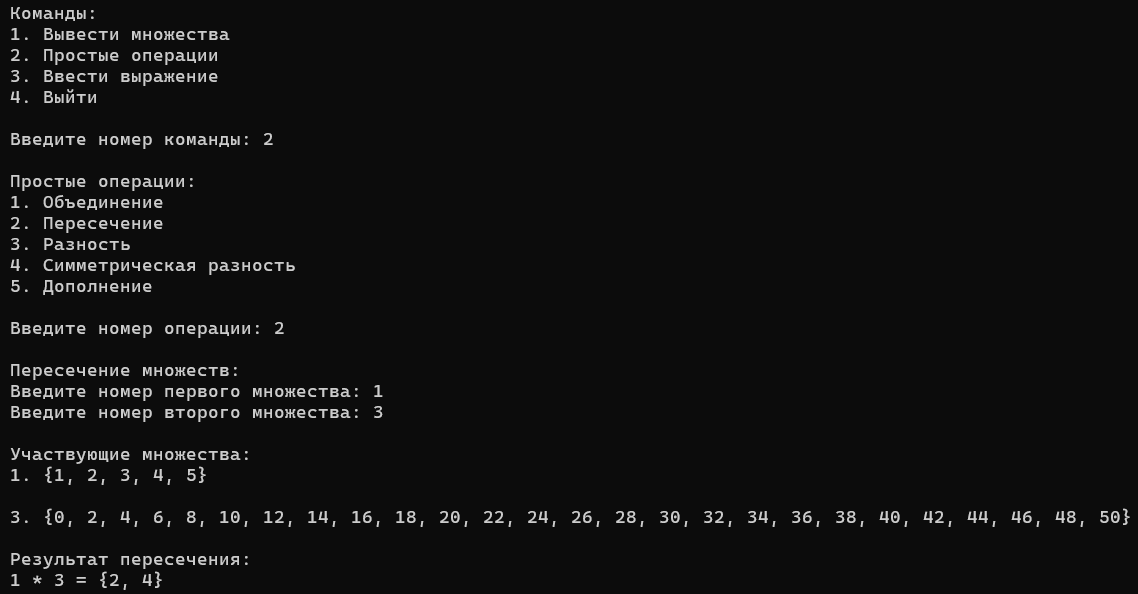
**3. Пример работы программы**

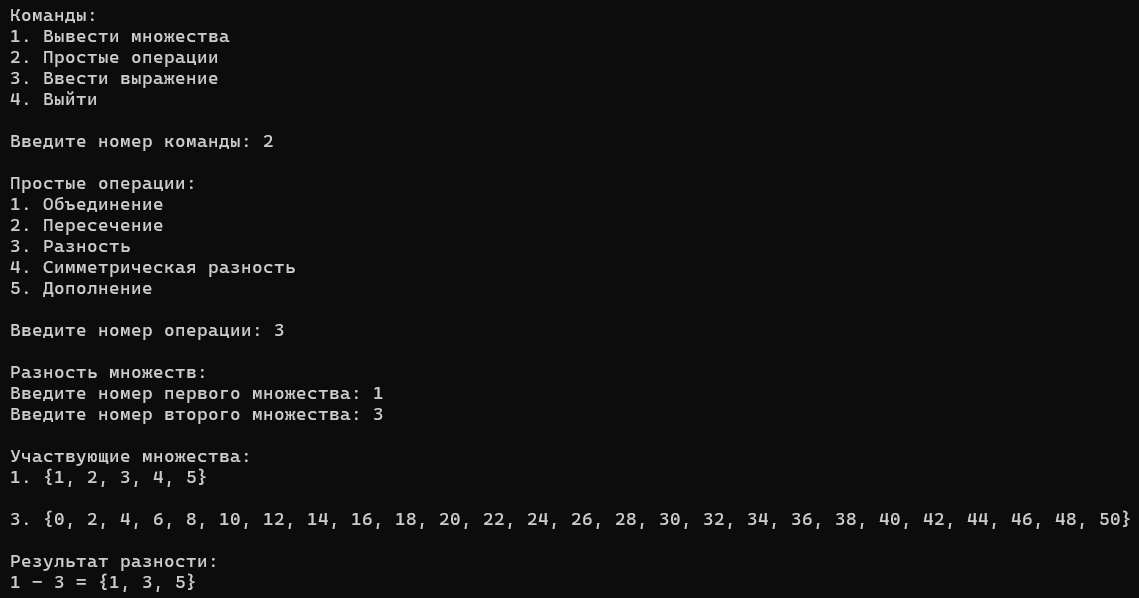


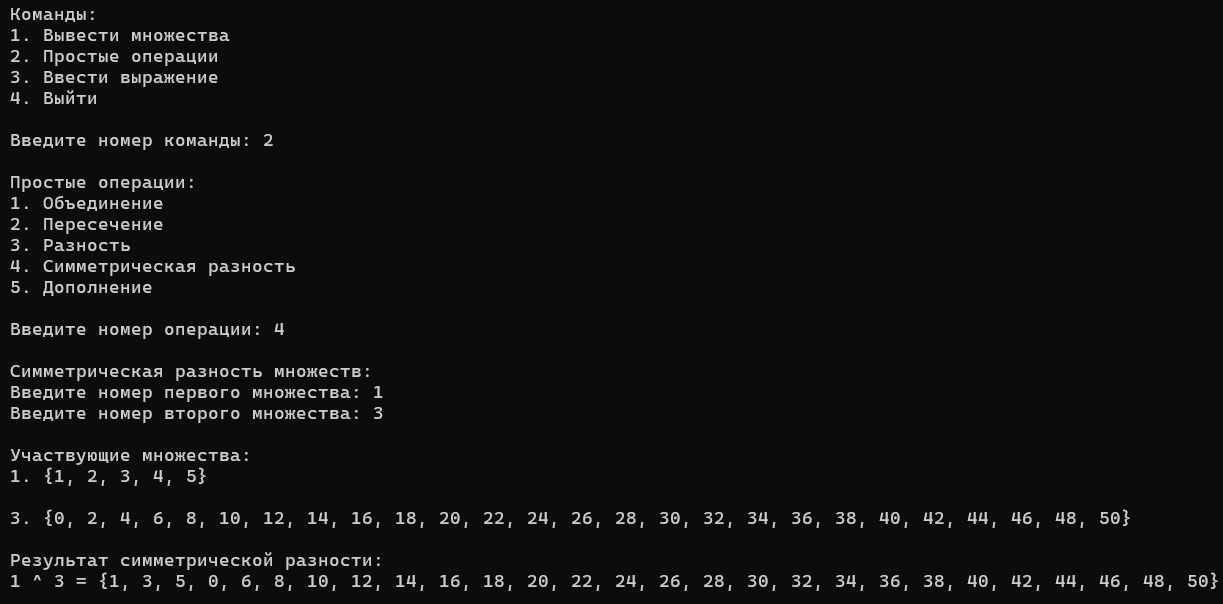


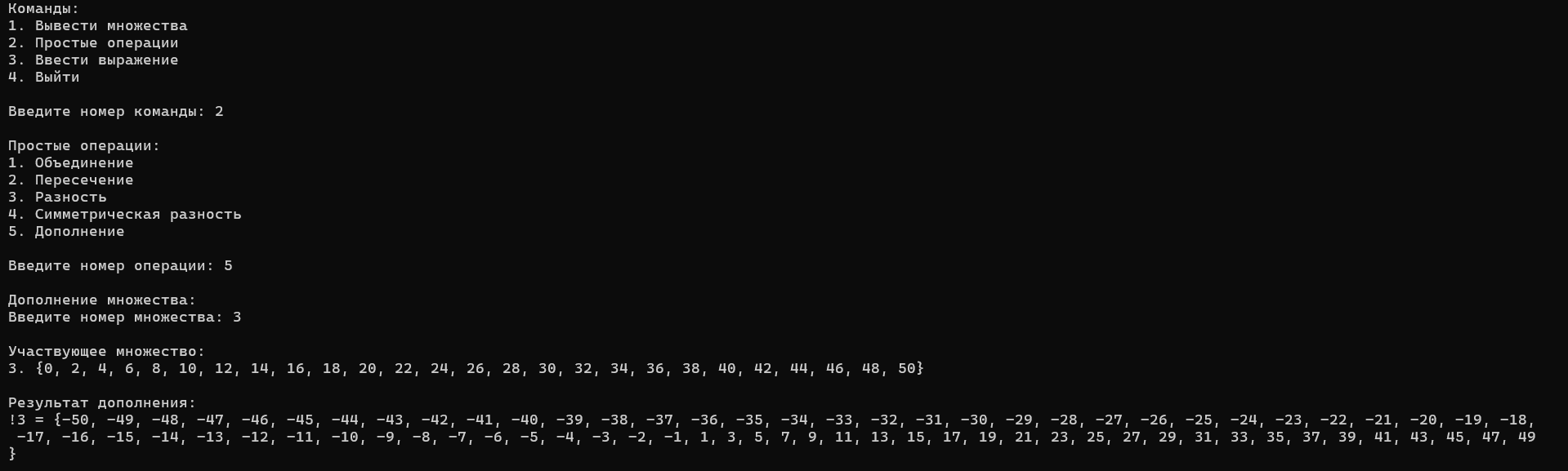


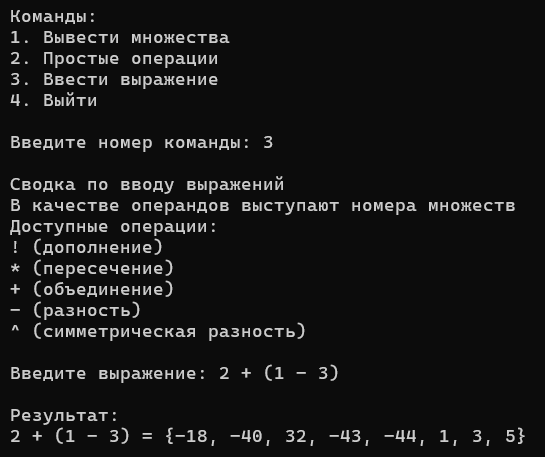


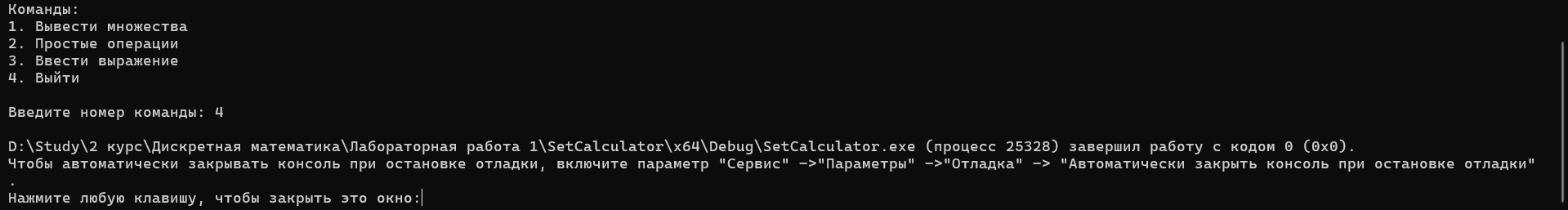












Ссылка на GitHub: <https://github.com/RuntovIvan/DiscreteMathematics>