Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования

**«Пермский национальный исследовательский
политехнический университет»**

Электротехнический факультет
Кафедра «Информационные технологии и автоматизированные системы»
направление подготовки: 09.03.01– «Информатика и вычислительная
техника»

Лабораторная работа № 1
по дисциплине
«Дискретная математика и математическая логика»
на тему
«Создание калькулятора множеств»

Выполнил студент гр. ИВТ-23-1б
Бакин Владислав Артемович

Проверил:
ст. преп. кафедры ИТАС
Рустамханова Г.И.

(оценка) (подпись)

(дата)

г. Пермь, 2024

Содержание

Цель и задачи работы

Цель: разработать калькулятор множеств.

Задачи:

1. Калькулятор должен предоставлять возможность задать как минимум 3 множества.
2. Калькулятор должен быть представлен в универсуме из целых чисел от -50 до 50.
3. Реализовать возможность задать множество следующими способами:
    1. заполнение случайными значениями,
    2. ввод пользователем с клавиатуры,
    3. совокупность условий.
4. Предусмотреть возможность задать выражение по формуле.

Этапы выполнения

1 Выбор технологического стека

Для выполнения данной работы был выбран язык программирования C++, фреймворк Qt, для создания графического интерфейса, средство автоматизации сборки программного обеспечения из исходного кода CMake.

2 Структура кода

Исходный код по данной лабораторной работе, а также всех последующих, можно посмотреть в репозитории на GitHub: https://github.com/Meidori/Discrete_Mathematics_Labs_2024-2025.

Главное окно описано в классе SetCalculator (рис. 1).

```
17  class SetCalculator : public QMainWindow
18  {
19      Q_OBJECT
20
21  public:
22      SetCalculator(QWidget *parent = nullptr);
23      ~SetCalculator();
24
25      void add();
26      void del();
27      void edit();
28      void updateOutput();
29
30      QVector<int> unionSets(const QVector<int>& set1, const QVector<int>& set2);
31      QVector<int> intersectionSets(const QVector<int>& set1, const QVector<int>& set2);
32      QVector<int> differenceSets(const QVector<int>& set1, const QVector<int>& set2);
33      QVector<int> symmetricSets(const QVector<int>& set1, const QVector<int>& set2);
34      QVector<int> complementSets(const QVector<int>& universalSet, const QVector<int>& subset);
35
36      void calcByFormula();
37      QVector<QString> processExpression(QString input);
38      void save();
39
40  private:
41      Ui::SetCalculator *ui;
42      QVector<int> universum;
43      std::unordered_map<QString, QVector<int>*> sets;
44
45  };
```

Рисунок 1 – Класс SetCalculator

Поле universum является QVector, содержащий элементы типа int. universum хранит все элементы универсума, которые инициализируются в конструкторе класса. Хэш-таблица (std::unordered_map) sets содержит пары ключ-значение, где ключ – это QString имя множества, а значение QVector<int> элементы множества.

Метод add отвечает за добавление множества в хэш-таблицу, del – за удаление. Метод edit позволяет редактировать содержание множества. updateOutput автоматически обновляет QTextEdit, содержащий информацию о множествах.

Методы unionSets, intersectionSets, differenceSets, symmetricSets, complementSets отвечают за выполнение операций над множествами.

calcByFormula, proccessExpression предназначены для обработки формулы (выражения) и её вычисления. Метод save позволяет сохранить полученное вычисление как новое множество.

Класс EditSets (рис. 2) описывает окно редактирования множеств.

```
17  class EditSets : public QDialog
18  {
19      Q_OBJECT
20
21  public:
22      explicit EditSets(std::unordered_map<QString, QVector<int>*>& sets, QString setName, QWidget *parent = nullptr);
23      ~EditSets();
24
25      void input();
26      void conditional();
27      void rand();
28      void clear();
29
30  private:
31      Ui::EditSets *ui;
32
33      std::unordered_map<QString, QVector<int>*>& sets;
34      QString setName;
35  };
```

Рисунок 2 – Класс EditSets

Класс InputElements (рис. 3) – окно ручного ввода элементов множества, класс Conditional (рис. 4) – окно заполнения множества по условиям.

```
17  class EditSets : public QDialog
18  {
19      Q_OBJECT
20
21  public:
22      explicit EditSets(std::unordered_map<QString, QVector<int>*>& sets, QString setName, QWidget *parent = nullptr);
23      ~EditSets();
24
25      void input();
26      void conditional();
27      void rand();
28      void clear();
29
30  private:
31      Ui::EditSets *ui;
32
33      std::unordered_map<QString, QVector<int>*>& sets;
34      QString setName;
35  };
```

Рисунок 3 – Класс EditSets

```
class Conditional : public QDialog
{
    Q_OBJECT

public:
    explicit Conditional(std::unordered_map<QString, QVector<int>*>& sets, QString setName, int size, QWidget *parent = nullptr);
    ~Conditional();

    void positive();
    void negative();
    void even();
    void odd();
    void multiplicity();
    void range();

private:
    Ui::Conditional *ui;

    std::unordered_map<QString, QVector<int>*>& sets;
    QString setName;
    int size;
};
```

Рисунок 4 – Класс Conditional

3 Демонстрация работы калькулятора

Для добавления множества пользователь должен ввести название множества и нажать кнопку добавить (рис. 5). Будет создано пустое множество с заданным именем. Множество можно удалить, нажав соответствующую кнопку.

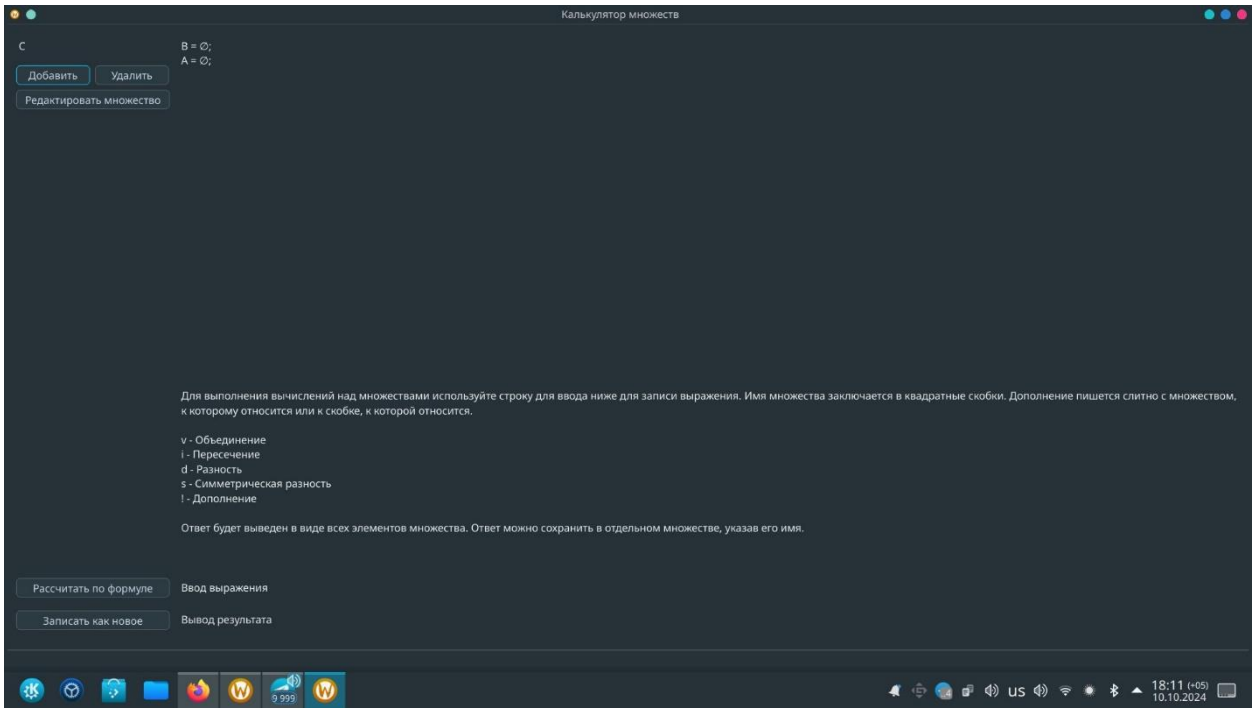

Рисунок 5 – Добавление множества

Для редактирования элементов множества нужно ввести его имя и нажать кнопку «Редактировать множество» (рис. 6).

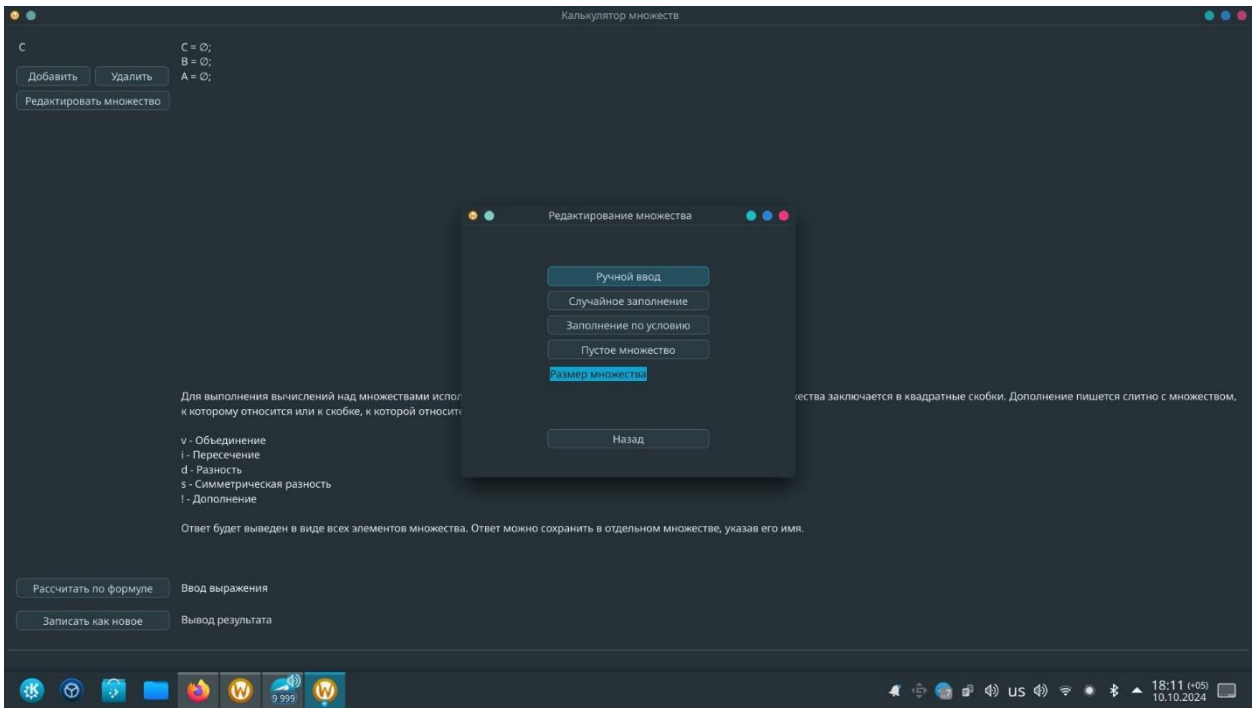

Рисунок 6 – Окно редактирования множества

В открывшемся окне появляется меню для выбора способа заполнения множества. Прежде чем выбрать способ задания элементов, нужно указать размер множества. Процесс ручного ввода для множества размером 10 показан на рисунках 7 и 8.

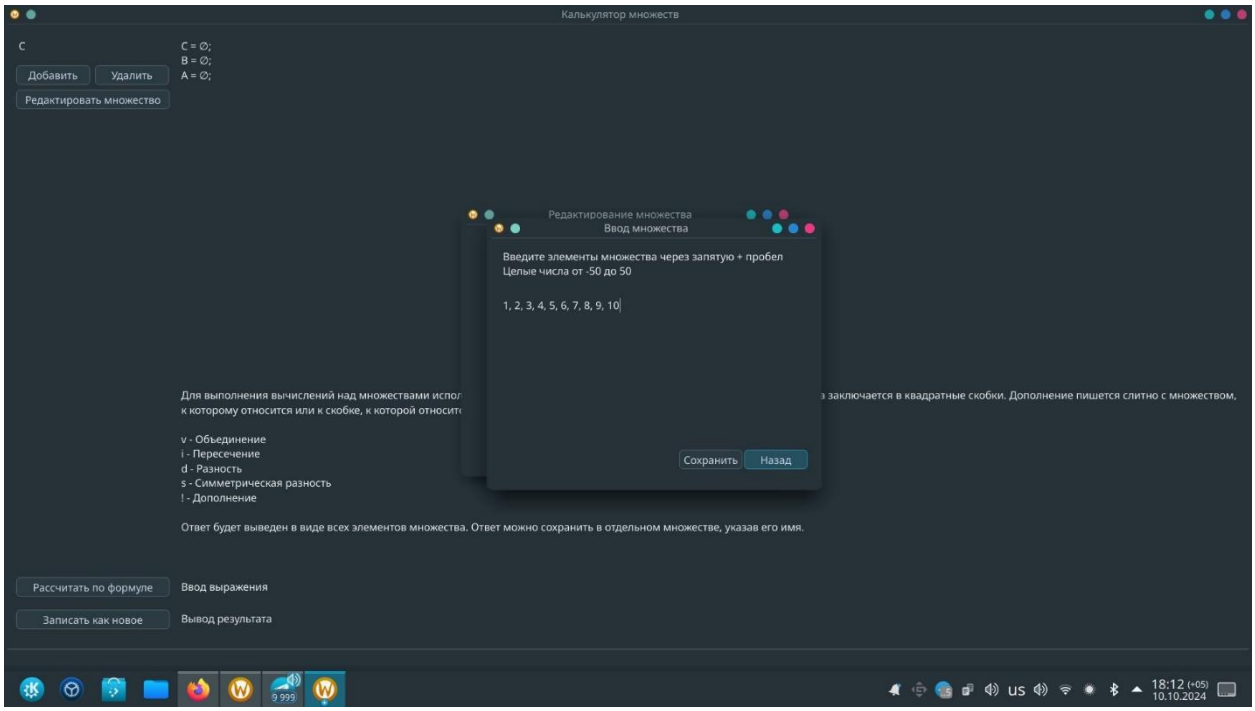

Рисунок 7 – Меню ручного ввода

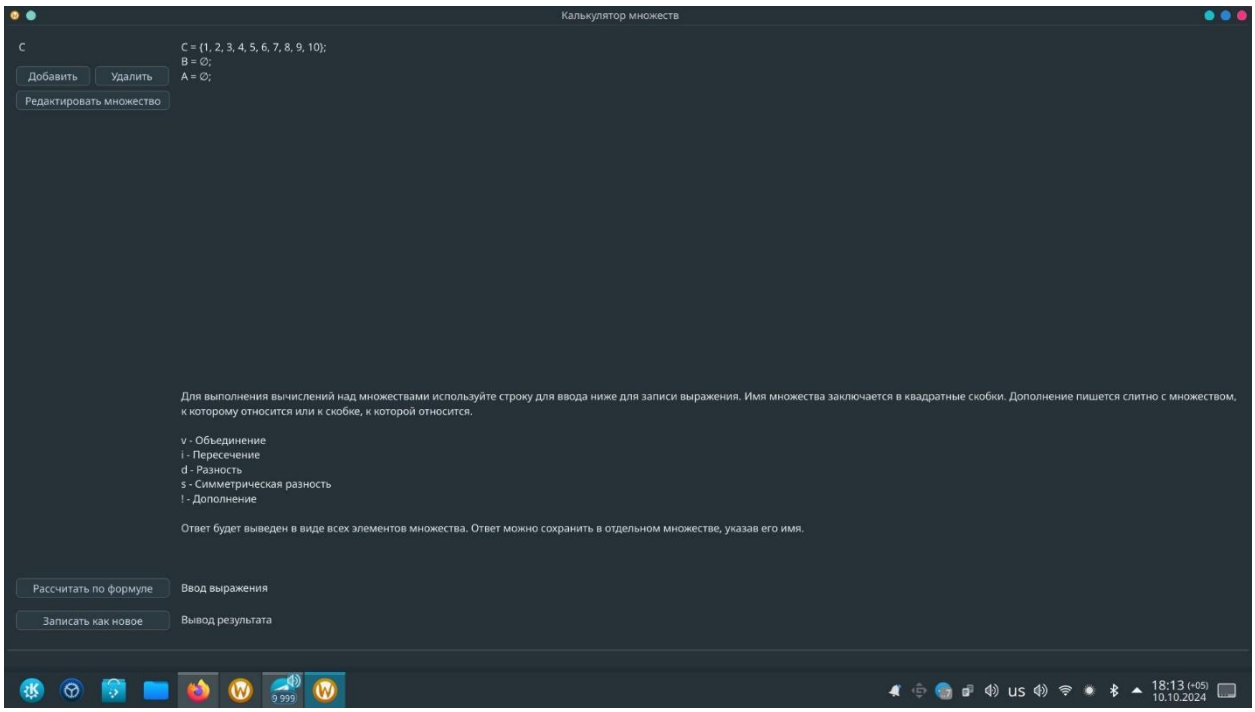

Рисунок 8 – Результат ручного ввода

Процесс случайного заполнения показан на рисунках 9 и 10.

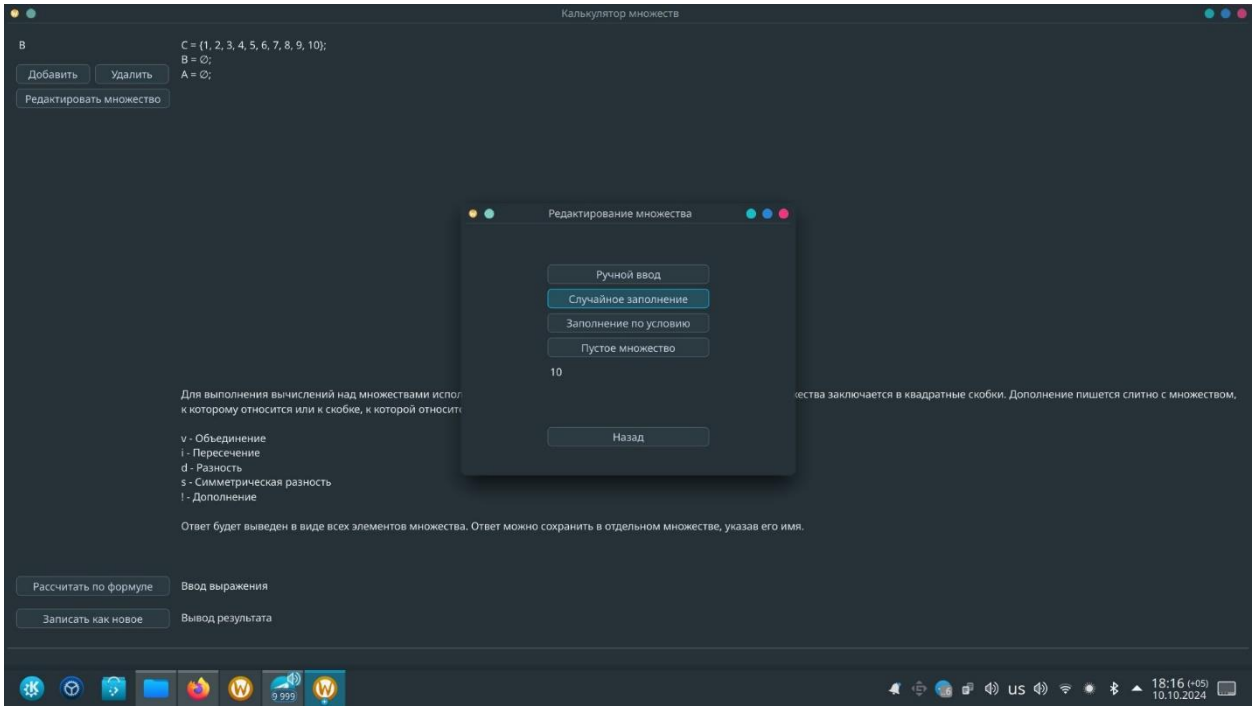

Рисунок 9 – Выбор случайного заполнения в меню

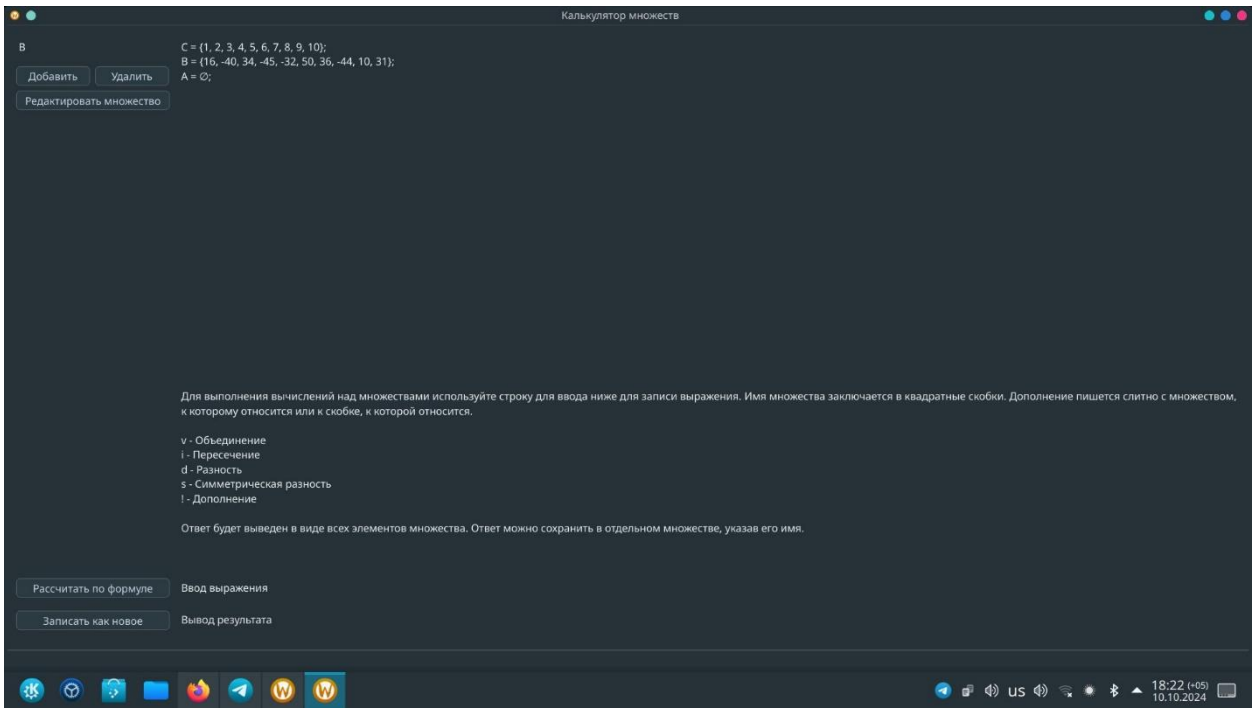

Рисунок 10 – Результат случайного заполнения

Действия для заполнения множества по условию изображены на рисунках 11-13.

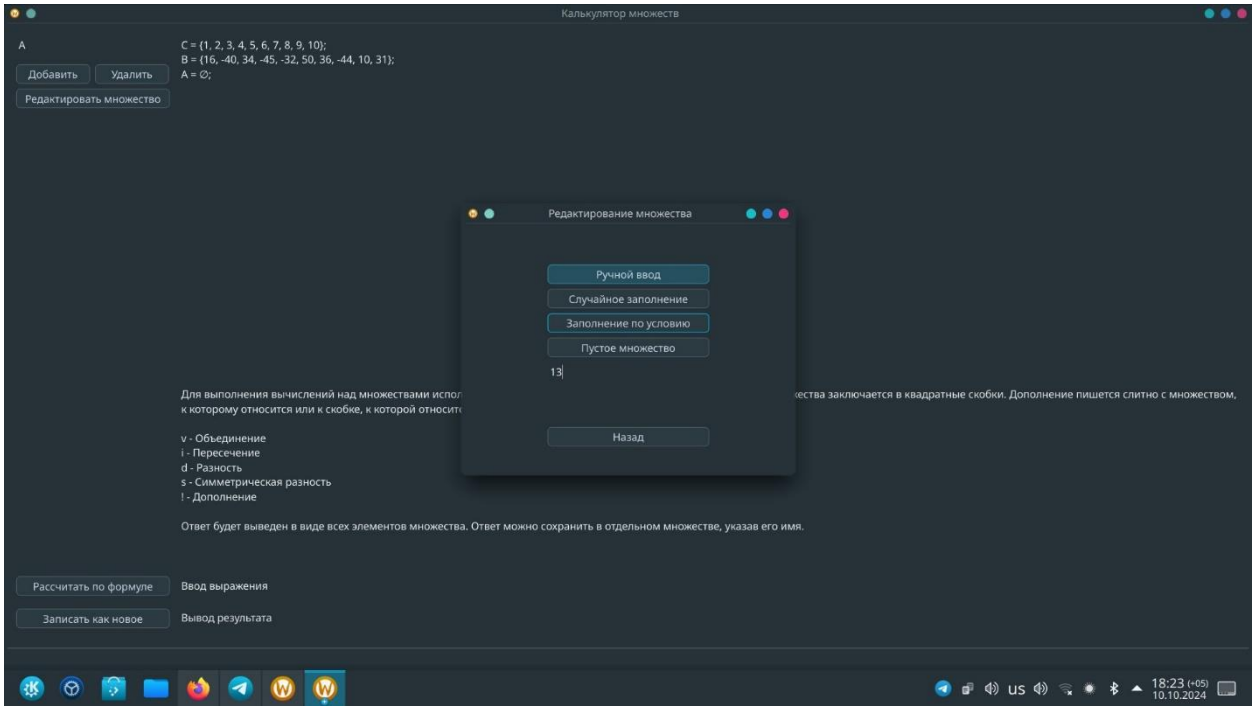

Рисунок 11 – Выбор заполнения по условию

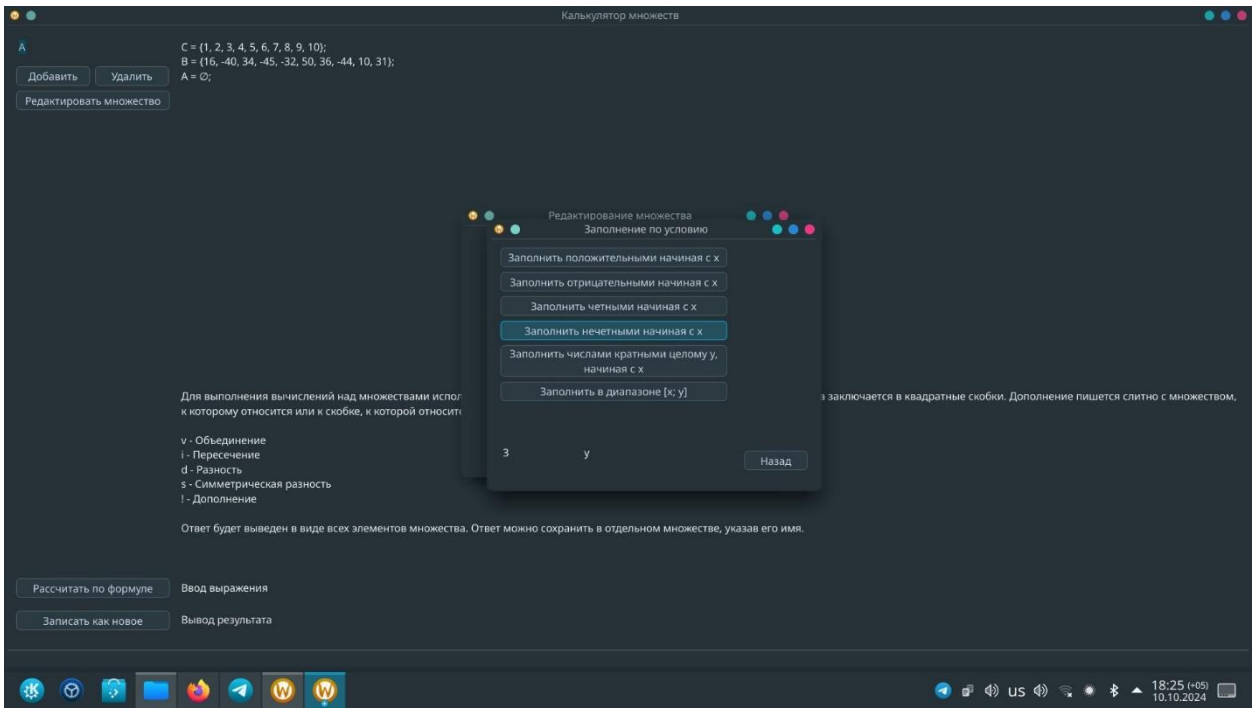

Рисунок 12 – Меню заполнения по условию

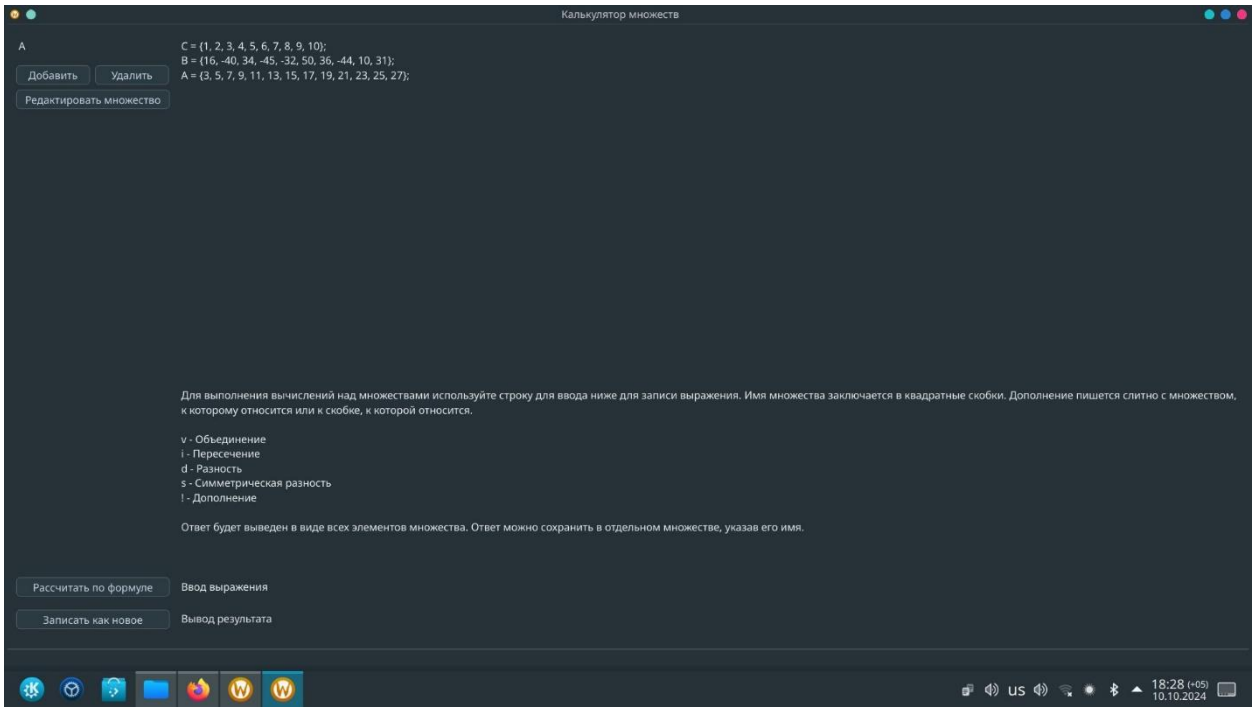

Рисунок 13 – Результат заполнения по условию

Также можно через меню «Редактировать множество» можно сделать множество пустым (рис. 14, 15)

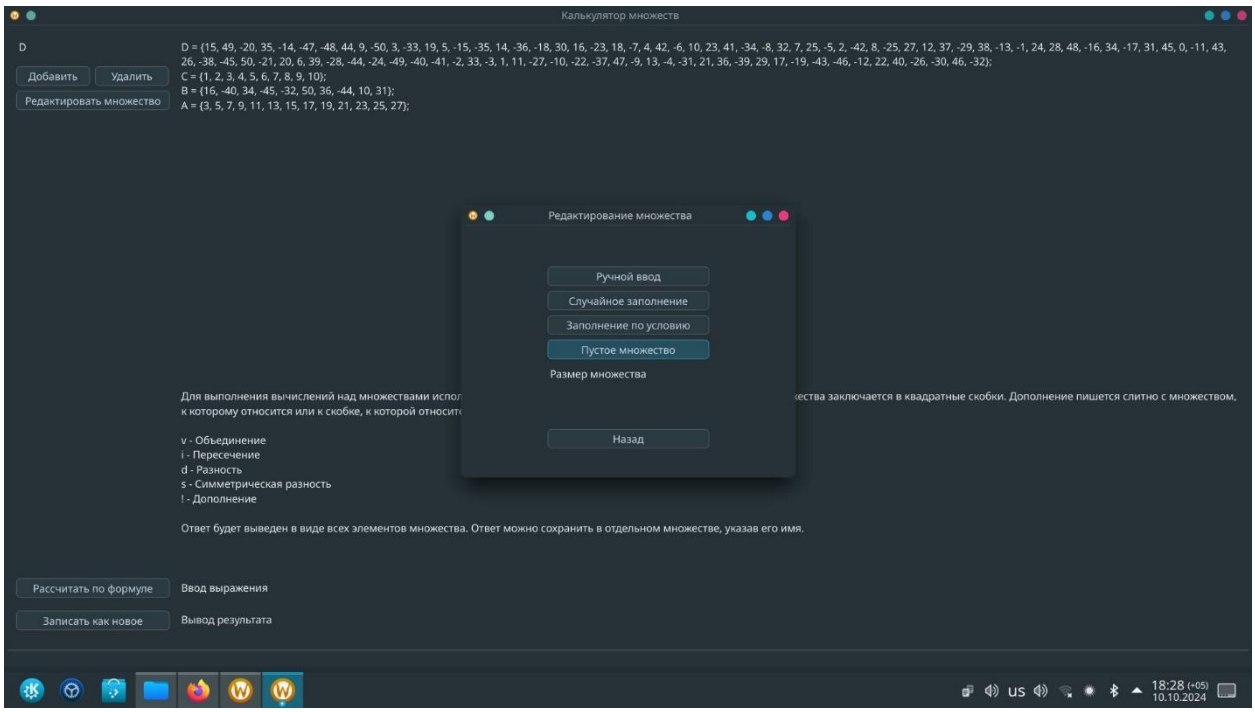

Рисунок 14 – Выбор операции «Пустое множество»

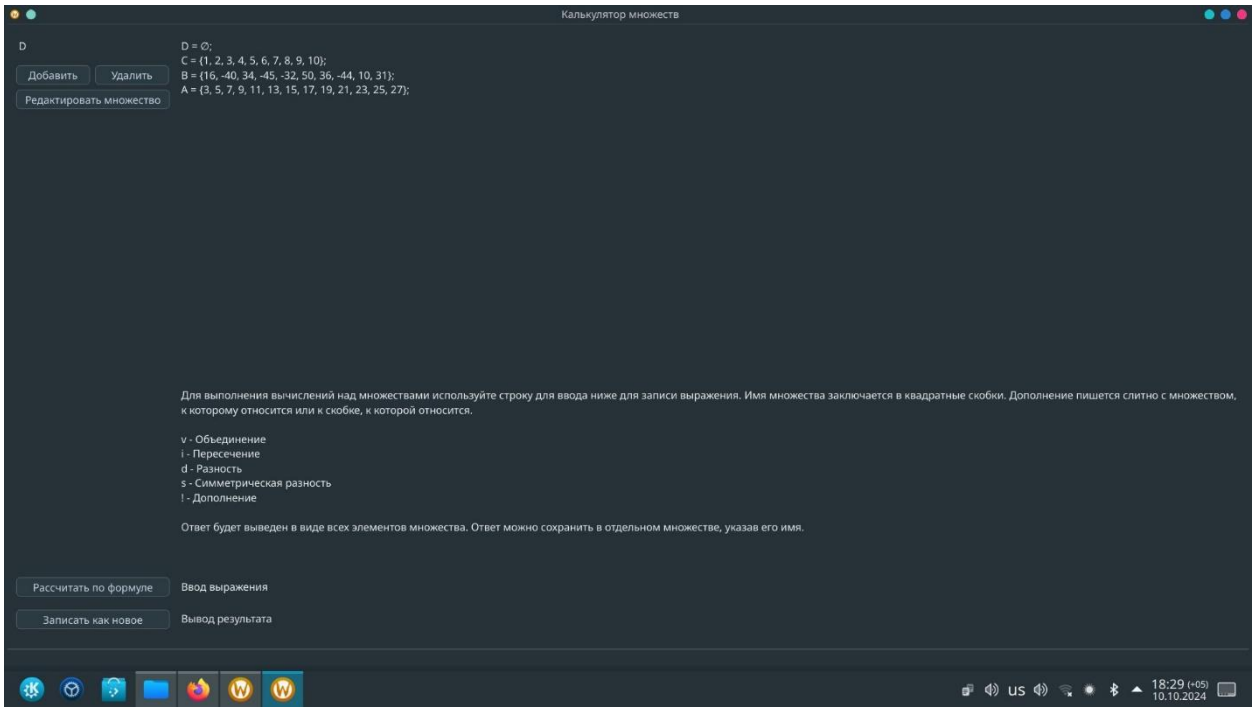

Рисунок 15 – Результат операции «Пустое множество»

Для выполнения операций над множествами используется QTextEdit, в котором пользователь должен ввести имена множеств в квадратных скобках и действие между ними (для дополнения – ввести «!» перед квадратными скобками). Затем после нажатия на кнопку «Рассчитать по формуле» в другом QTextEdit будет выведен результат вычисления. Пользователь может сохранить результат в виде нового множества, если введет название для нового множества и нажмет на кнопку «Записать как новое».

Операция объединения показана на рисунке 16.

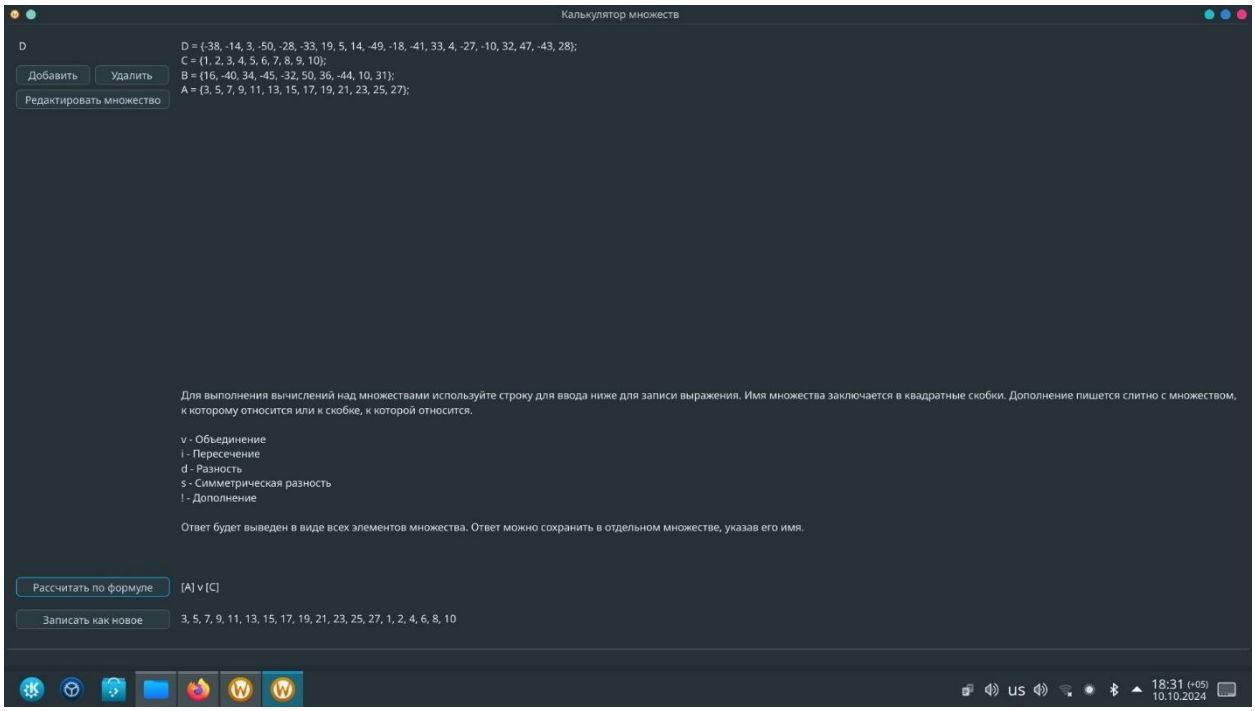

Рисунок 16 – Объединение

Операция пересечения показана на рисунке 17.

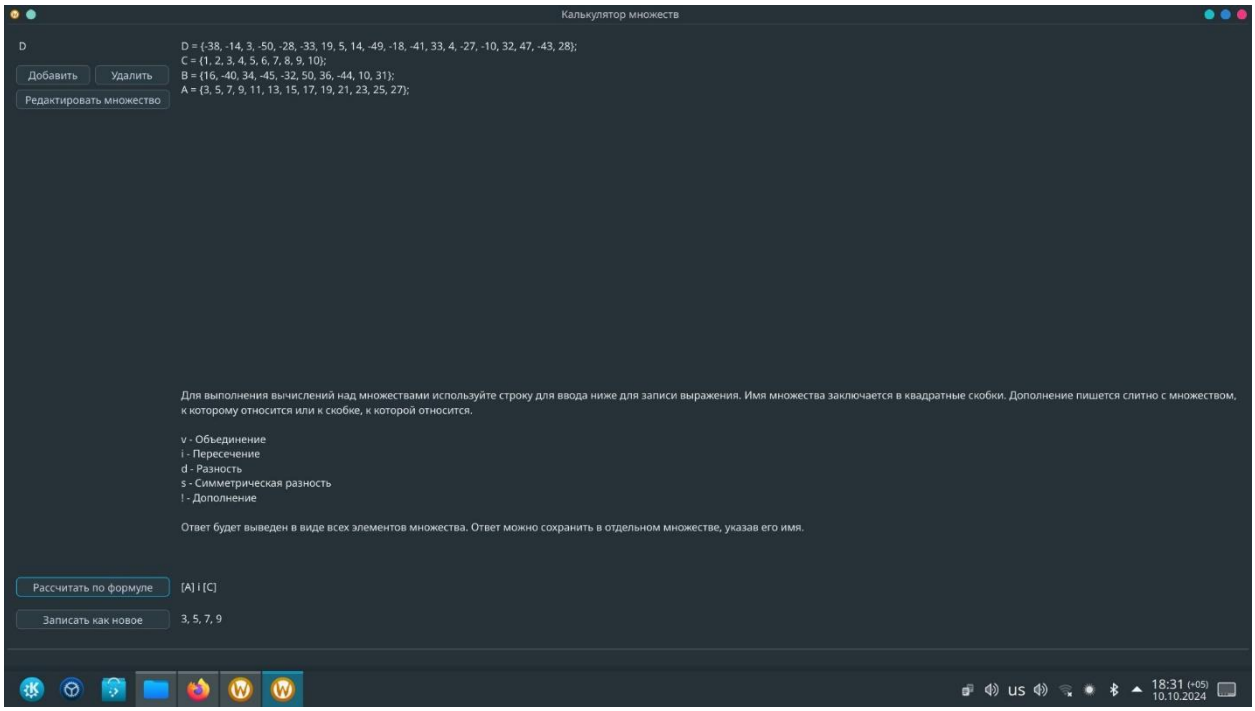

Рисунок 17 – Пересечение

Операция разности показана на рисунке 18.

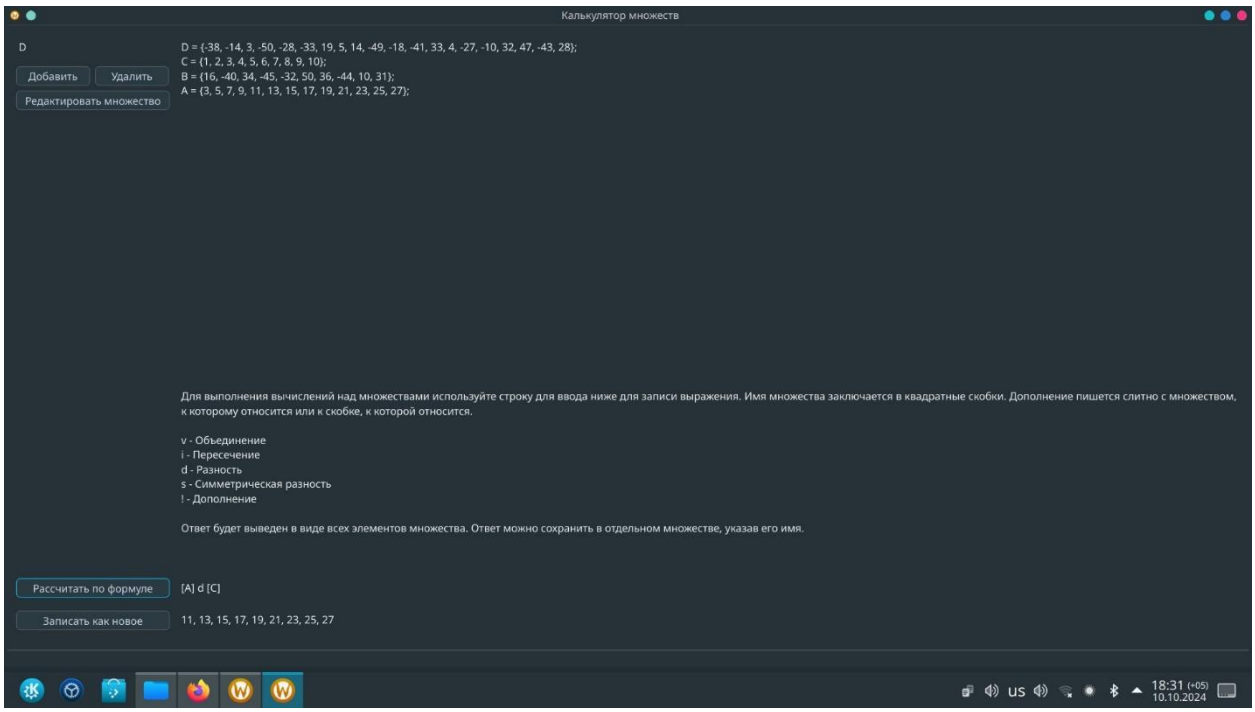

Рисунок 18 – Разность

Операция симметрической разности показана на рисунке 19.

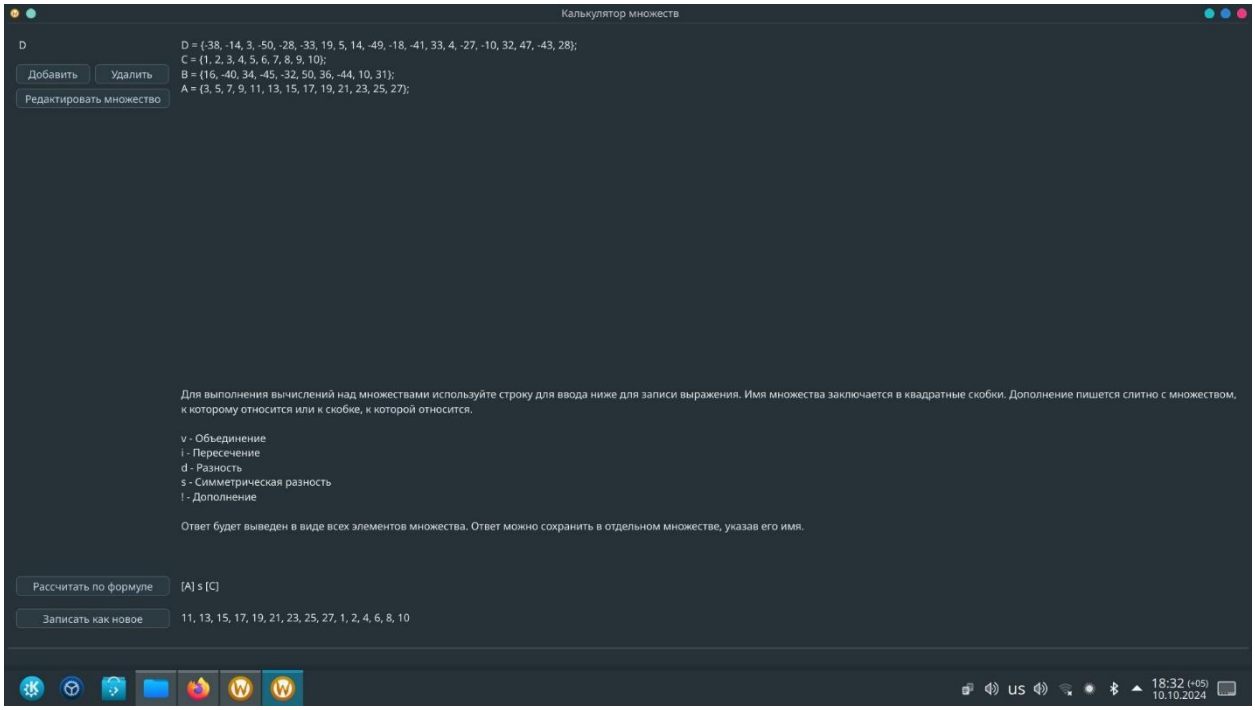

Рисунок 19 – Симметрическая разность

Операция дополнения показана на рисунках 20 и 21.

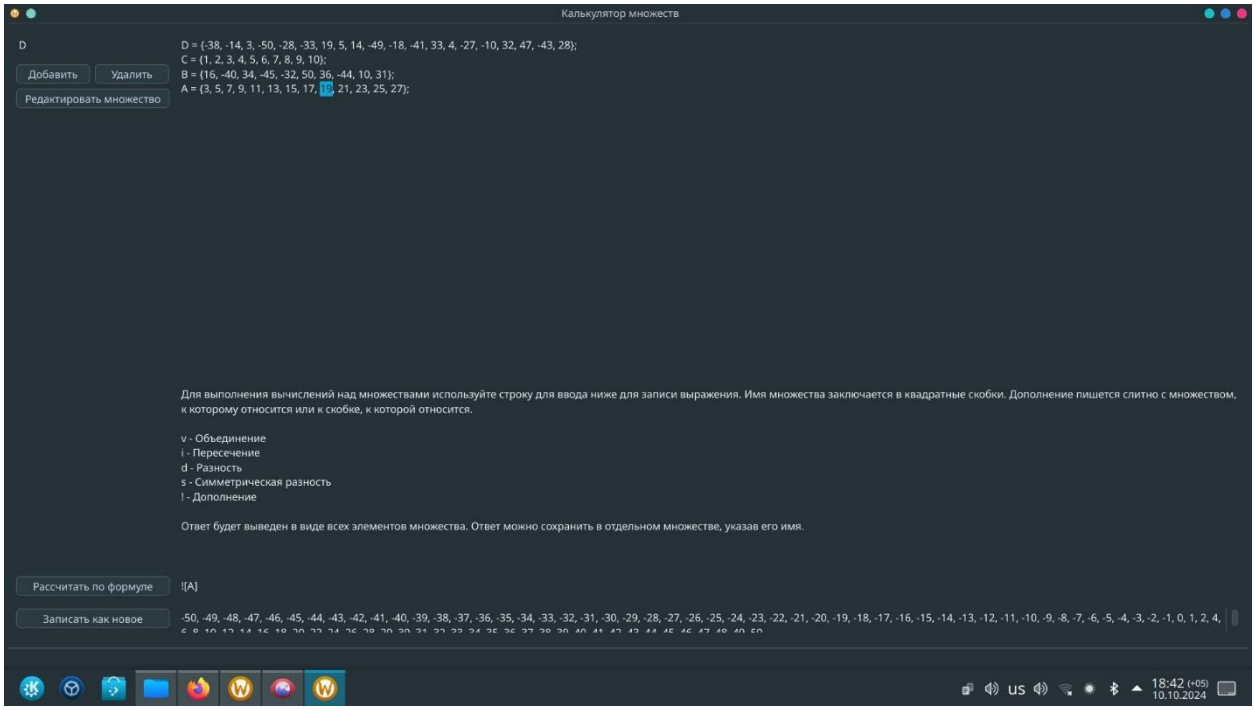

Рисунок 20 – Дополнение 1

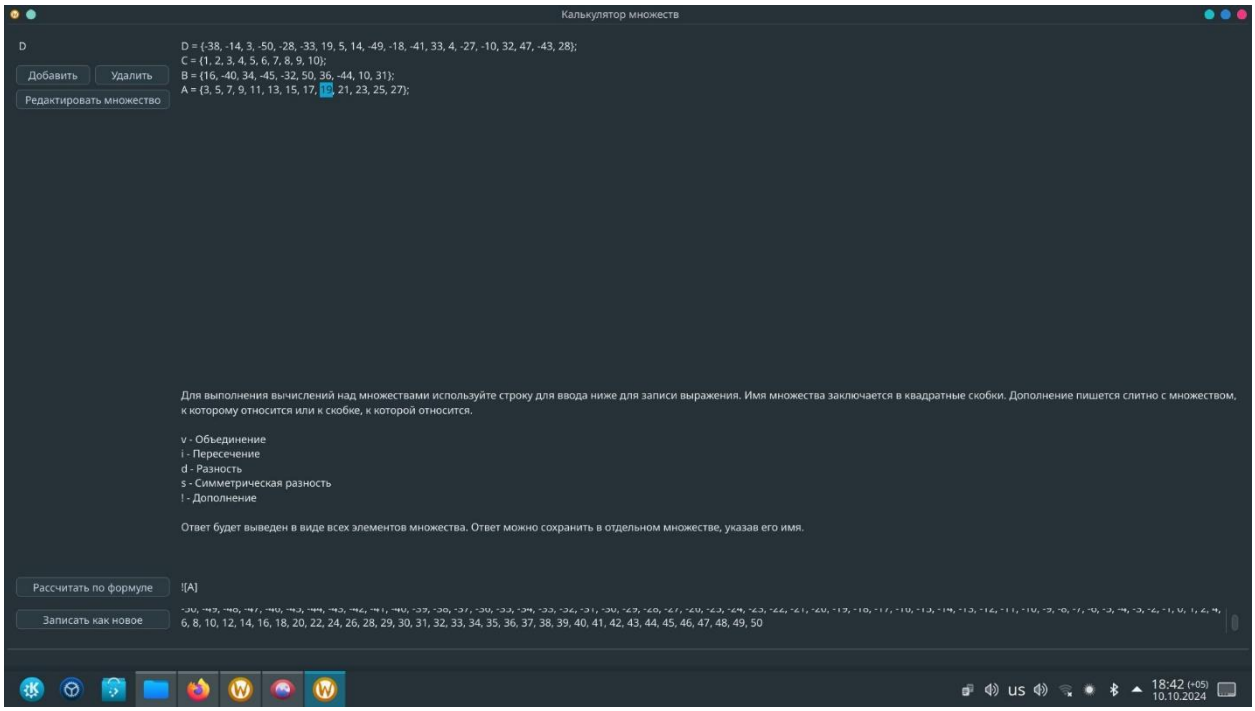

Рисунок 21 – Дополнение 2

Пользователь может ввести целое выражения для вычисления (в выражении дополнение может относится к целой скобке) (рис 22).

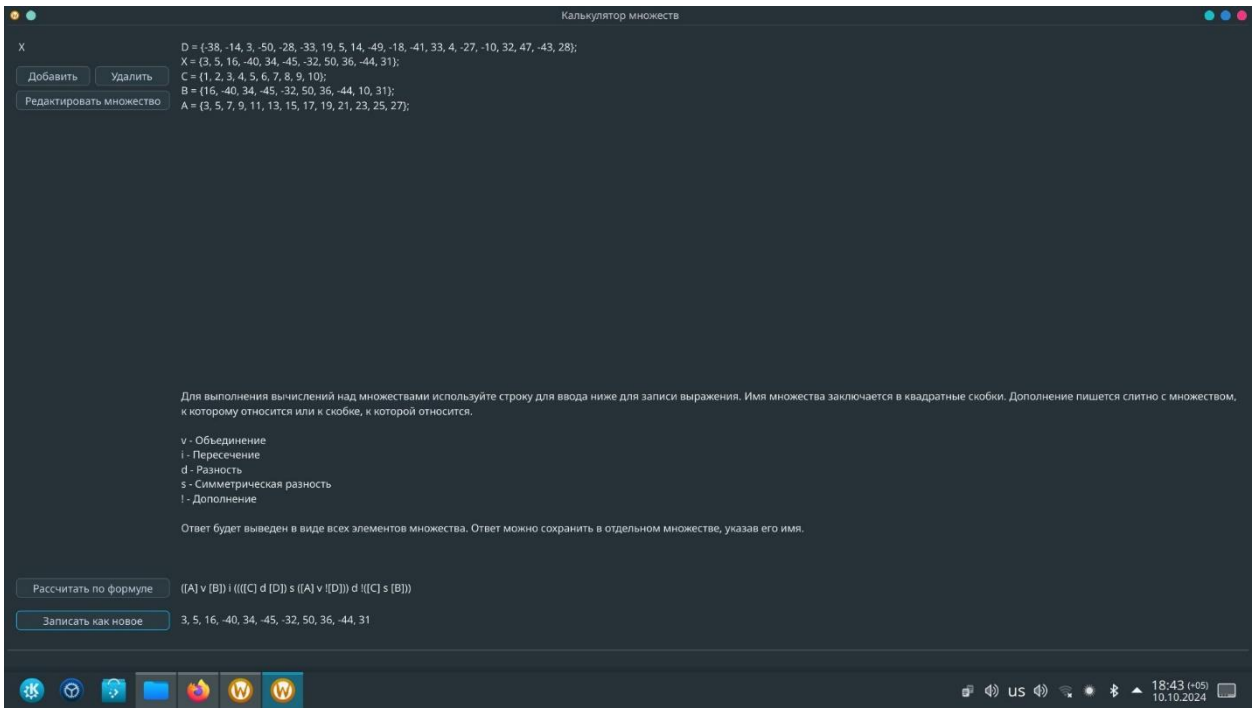

Рисунок 22 – Пример вычисления выражения

Процесс сохранения результата выражения в виде нового множества «X» показана на рисунке 23.

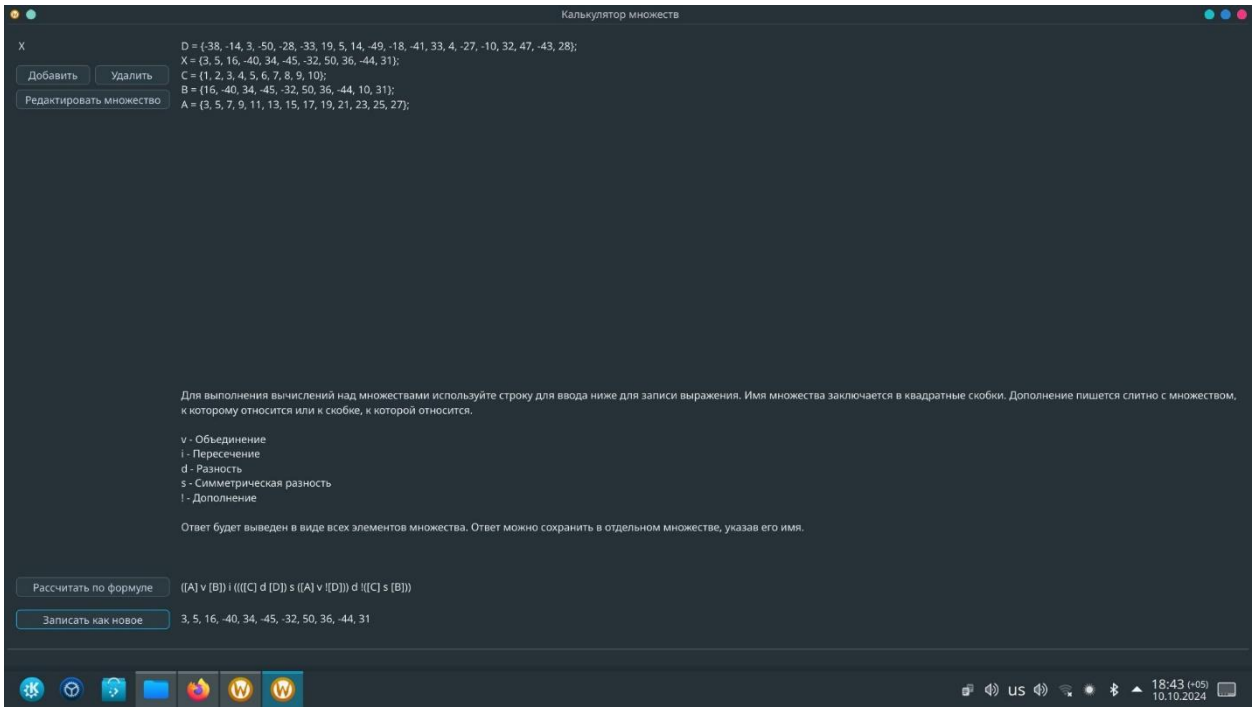

Рисунок 23 – Сохранение вычисления

Вывод по лабораторной работе

В ходе выполнения лабораторной работы была разработана программа-калькулятор множеств, реализованная на языке C++ с использованием фреймворка Qt для создания графического интерфейса. В процессе выполнения были закреплены знания по теории множеств [1] и получено практическое применение полученных знаний через разработку программного обеспечения.

Список использованных источников

1. Теория множеств: основы и базовые операции над множествами // Хекслет URL: https://ru.hexlet.io/blog/posts/teoriya-mnozhestv-osnovy-i-bazovye-operatsii-nad-mnozhestvami (дата обращения: 08.10.2024).