

Министерство образования и науки РФ  
Пермский национальный исследовательский политехнический университет  
Электротехнический факультет  
Кафедра информационных технологий и автоматизированных систем

Дискретная математика и математическая логика  
Лабораторная работа № 1

Тема: «Разработка калькулятора множеств»

Выполнил: студент группы  
ИВТ -23-1Б  
Долганова Диана Евгеньевна  
Проверил: ст. пр.  
Рустамханова Г. И.

## **Оглавление**

<b>Цель работы.....</b>	<b>2</b>
<b>Задачи работы.....</b>	<b>3</b>
<b>Этапы выполнения.....</b>	<b>4</b>
<b>Тестирование программы.....</b>	<b>14</b>
<b>Заключение.....</b>	<b>18</b>
<b>Список используемой литературы.....</b>	<b>19</b>

## **Цель работы**

Разработать калькулятор множеств.

## Задачи работы

Разработать калькулятор множеств, который будет иметь следующие характеристики:

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. Возможность задать множество следующими способами:
  - 3.1 случайная наполненность множества
  - 3.2 ручной ввод
  - 3.3 совокупность условий. Меню условий: условие соблюдения знаков, кратность какому-либо числу (включая четность и нечетность), диапазон от минимального к максимальному, предусмотреть на количество допустимых значений.
4. Меню отдельных операций над множествами (объединение, разность, симметрическая разность, дополнение до универсума).
5. Составление формулы множества

## Этапы выполнения

Разработка калькулятора была выполнена на языке C++ в программе Microsoft Visual Studio 2022.

1. Задаем границы универсума от -50 до 50 и его размер (рис. 1).

```
#include <iostream>
#include <vector>
#include <sstream>
#include <cstdlib>
#include <ctime>

using namespace std;

// Задаем границы универсума
const int UNIVERSE_MIN = -50; // Минимальное значение универсума
const int UNIVERSE_MAX = 50; // Максимальное значение универсума
const int UNIVERSE_SIZE = UNIVERSE_MAX - UNIVERSE_MIN + 1; // Размер универсума
```

Рисунок 1 - Создание границ универсума

2. Объявляем класс SetCalculator, предназначенный для работы с множествами, и создаем его конструктор. Перечисляем методы класса, которые будут использоваться для добавления множеств, их отображения и выполнения различных операций над ними. Создаем переменные условий (рис. 2).

```
// Конструктор класса
SetCalculator() {
    sets.reserve(10); // Резервируем место для множеств (до 10)
    minAllowedValue = UNIVERSE_MIN; // Устанавливаем минимальное допустимое значение
    maxAllowedValue = UNIVERSE_MAX; // Устанавливаем максимальное допустимое значение
    allowNegative = true; // Разрешаем отрицательные числа
    allowEven = true; // Разрешаем четные числа
    allowOdd = true; // Разрешаем нечетные числа
}

// Методы класса
void addRandomSet(); // Добавление случайного множества
void addManualSet(); // Добавление множества вручную
size_t displaySets(); // Отображение всех множеств
void unionSets(int a, int b); // Объединение множеств
void differenceSets(int a, int b); // Разность множеств
void intersectionSets(int a, int b); // Пересечение множеств
void symmetricDifference(int a, int b); // Симметрическая разность
void complementSet(int index); // Дополнение множества
void calculateExpression(const string& expression); // Расчет выражения
void setConditions(); // Установка условий

private:
    vector<IntSet> sets;

    // Переменные для условий
    int minAllowedValue; // Минимально допустимое значение
    int maxAllowedValue; // Максимально допустимое значение
    bool allowNegative; // Флаг для разрешения отрицательных чисел
    bool allowEven; // Флаг для разрешения четных чисел
    bool allowOdd; // Флаг для разрешения нечетных чисел

    void addToSet(IntSet& set, int value);
    IntSet parseExpression(const string& expression);
    bool checkConditions(int value);
};
```

Рисунок 2 - Объявление класса SetCalculator

### 3. Метод добавления элементов в множество

Метод проверяет, соответствует ли значение заданным условиям, и если да, устанавливает позицию в множестве (в векторе) как true, что указывает на наличие этого значения (рис. 3)

```
// Метод добавления элемента в множество
void SetCalculator::addToSet(IntSet& set, int value) {
    if (checkConditions(value)) { // Проверяем условия
        set[value - UNIVERSE_MIN] = true; // Добавляем значение в множество
    }
    else {
        cout << "Значение " << value << " не соответствует условиям." << endl;
    }
}
```

Рисунок 3 - Метод добавления элементов в множество

### 4. Метод проверки условий

Метод проверяет, соответствует ли значение установленным условиям (в рамках универсума, отрицательные, четные/нечетные). Возвращает true, если все проверки пройдены (рис. 4)

```
// Метод проверки, соответствует ли значение условиям
bool SetCalculator::checkConditions(int value) {
    if (value < minAllowedValue || value > maxAllowedValue) return false; // Проверка по границам
    if (!allowNegative && value < 0) return false; // Проверка на отрицательные значения
    if (!allowEven && value % 2 == 0) return false; // Проверка на четные значения
    if (!allowOdd && value % 2 != 0) return false; // Проверка на нечетные значения
    return true;
}
```

Рисунок 4 - Метод проверки условий

### 5. Метод добавления случайного множества

Создает новое множество, заполняет его случайными числами из универсума и добавляет в контейнер sets (рис. 5)

```
// Метод добавления случайного множества
void SetCalculator::addRandomSet() {
    IntSet newSet(UNIVERSE_SIZE, false); // Создаем новое множество заполненное ложными значениями
    int size = rand() % (UNIVERSE_SIZE + 1); // Генерируем случайный размер множества
    while (size-- > 0) {
        int value = rand() % UNIVERSE_SIZE + UNIVERSE_MIN; // Генерируем случайное значение из универсума
        addToSet(newSet, value); // Добавляем значение в множество
    }
    sets.push_back(newSet); // Сохраняем множество
}
```

Рисунок 5 - Метод добавления случайного множества

## 6. Метод ручного добавления множества

Метод позволяет пользователю вручную вводить числа, которые затем добавляются в множество. Ввод заканчивается при вводе символа 'q' (рис. 6)

```
// Метод для ручного добавления множества
void SetCalculator::addManualSet() {
    IntSet newSet(UNIVERSE_SIZE, false); // Создаем новое множество
    string input;
    cout << "Введите числа для множества (q для окончания ввода):" << endl;
    while (true) {
        cin >> input;
        if (input == "q") break;
        try {
            int number = stoi(input); // Пробуем преобразовать ввод в число
            addToSet(newSet, number); // Добавляем число в множество
        }
        catch (invalid_argument&) {
            cout << "Некорректный ввод, попробуйте снова." << endl;
        }
    }
    sets.push_back(newSet); // Сохраняем множество
}
```

Рисунок 6 - Метод ручного добавления множества

## 7. Метод отображения всех множеств

Метод проходит по всем множествам, выводя их содержимое на экран (рис. 7).

```
// Метод отображения всех множеств
size_t SetCalculator::displaySets() {
    for (size_t i = 0; i < sets.size(); ++i) { // Проходим по всем множествам
        cout << "Множество " << (i + 1) << ": { ";
        for (int j = 0; j < UNIVERSE_SIZE; ++j) {
            if (sets[i][j]) { // Если элемент существует в множестве
                cout << (j + UNIVERSE_MIN) << " "; // Выводим элемент
            }
        }
        cout << "}" << endl;
    }
    return sets.size(); // Возвращаем количество множеств
}
```

Рисунок 6 - Метод отображения всех множеств

## 8. Методы для выполнения операций над множествами

Методы для выполнения операций имеют схожую структуру. Они создают новое множество resultSet, выполняют необходимые операции с элементами и выводят результат (рис. 7 - 11).

```

// Метод объединения двух множеств
void SetCalculator::unionSets(int a, int b) {
    IntSet resultSet(UNIVERSE_SIZE, false); // Создаем результирующее множество
    for (int i = 0; i < UNIVERSE_SIZE; ++i) {
        resultSet[i] = sets[a][i] || sets[b][i]; // Объединяем множества
    }

    cout << "Объединение множеств: { ";
    for (int j = 0; j < UNIVERSE_SIZE; ++j) {
        if (resultSet[j]) { // Если элемент существует в результирующем множестве
            cout << (j + UNIVERSE_MIN) << " "; // Выводим элемент
        }
    }

    cout << "}" << endl;
}

```

Рисунок 7 - Метод объединения двух множеств

```

// Метод вычисления разности двух множеств
void SetCalculator::differenceSets(int a, int b) {
    IntSet resultSet(UNIVERSE_SIZE, false); // Создаем результирующее множество
    for (int i = 0; i < UNIVERSE_SIZE; ++i) {
        resultSet[i] = sets[a][i] && !sets[b][i]; // Вычисляем разность
    }

    cout << "Разность множеств: { ";
    for (int j = 0; j < UNIVERSE_SIZE; ++j) {
        if (resultSet[j]) {
            cout << (j + UNIVERSE_MIN) << " "; // Выводим элемент
        }
    }

    cout << "}" << endl;
}

```

Рисунок 8 - Метод вычисления разности двух множеств

```

// Метод вычисления пересечения двух множеств
void SetCalculator::intersectionSets(int a, int b) {
    IntSet resultSet(UNIVERSE_SIZE, false); // Создаем результирующее множество
    for (int i = 0; i < UNIVERSE_SIZE; ++i) {
        resultSet[i] = sets[a][i] && sets[b][i]; // Вычисляем пересечение
    }

    cout << "Пересечение множеств: { ";
    for (int j = 0; j < UNIVERSE_SIZE; ++j) {
        if (resultSet[j]) {
            cout << (j + UNIVERSE_MIN) << " "; // Выводим элемент
        }
    }

    cout << "}" << endl;
}

```

Рисунок 9 - Метод вычисления пересечения двух множеств

```

// Метод вычисления симметрической разности двух множеств
void SetCalculator::symmetricDifference(int a, int b) {
    IntSet resultSet(UNIVERSE_SIZE, false); // Создаем результирующее множество
    for (int i = 0; i < UNIVERSE_SIZE; ++i) {
        resultSet[i] = sets[a][i] != sets[b][i]; // Вычисляем симметрическую разность
    }

    cout << "Симметрическая разность: { ";
    for (int j = 0; j < UNIVERSE_SIZE; ++j) {
        if (resultSet[j]) {
            cout << (j + UNIVERSE_MIN) << " "; // Выводим элемент
        }
    }

    cout << "}" << endl;
}

```

Рисунок 10 - Метод вычисления симметрической разности двух множеств



```

// Метод вычисления дополнения множества
void SetCalculator::complementSet(int index) {
    IntSet resultSet(UNIVERSE_SIZE, false); // Создаем результирующее множество
    for (int i = 0; i < UNIVERSE_SIZE; ++i) {
        resultSet[i] = !sets[index][i]; // Вычисляем дополнение
    }
    cout << "Дополнение до универсума: { ";
    for (int j = 0; j < UNIVERSE_SIZE; ++j) {
        if (resultSet[j]) {
            cout << (j + UNIVERSE_MIN) << " "; // Выводим элемент
        }
    }
    cout << "}" << endl;
}

```

Рисунок 11 - Метод вычисления дополнения множества

## 9. Метод парсинга выражения

Метод `parseExpression` обрабатывает строковое выражение для выполнения операций над множествами. Он инициализирует пустое множество, считывает токены, выполняет операции (объединение, разность, пересечение, симметрическая разность, дополнение) и возвращает итоговые элементы (рис. 12 - 17).

```

// Метод парсинга выражения
SetCalculator::IntSet SetCalculator::parseExpression(const string& expression) {
    IntSet resultSet(UNIVERSE_SIZE, false); // Создаем результирующее множество
    stringstream ss(expression); // Создаем поток для обработки строки
    string token; // Переменная для хранения токенов

    while (ss >> token) { // Считываем токены из строки
        if (token == "1" || token == "2" || token == "3") { // Если токен - индекс множества
            int setIndex = stoi(token) - 1; // Преобразуем токен в индекс
            if (setIndex >= 0 && setIndex < sets.size()) { // Проверяем действительность индекса
                for (int i = 0; i < UNIVERSE_SIZE; ++i) {
                    resultSet[i] = resultSet[i] || sets[setIndex][i]; // Объединяем множество
                }
            }
        }
    }
}

```

Рисунок 12 - Инициализация и считывание токенов

```

else if (token == "U" || token == "u") { // Если токен - объединение
    ss >> token; // Считываем следующий токен
    if (token == "1" || token == "2" || token == "3") {
        int setIndex = stoi(token) - 1; // Преобразуем токен в индекс
        if (setIndex >= 0 && setIndex < sets.size()) {
            for (int i = 0; i < UNIVERSE_SIZE; ++i) {
                resultSet[i] = resultSet[i] || sets[setIndex][i]; // Объединяем множество
            }
        }
    }
}
}

```

Рисунок 13 - Объединение

```

else if (token == "\\") { // Если токен - разность
    IntSet tempSet = resultSet; // Сохраняем текущее множество
    ss >> token; // Считываем следующий токен
    if (token == "1" || token == "2" || token == "3") {
        int setIndex = stoi(token) - 1; // Преобразуем токен в индекс
        if (setIndex >= 0 && setIndex < sets.size()) {
            for (int i = 0; i < UNIVERSE_SIZE; ++i) {
                resultSet[i] = tempSet[i] && !sets[setIndex][i]; // Вычисляем разность
            }
        }
    }
}
}

```

Рисунок 14 - Разность

```

else if (token == "5") { // Если токен - симметрическая разность
    IntSet tempSet = resultSet; // Сохраняем текущее множество
    ss >> token; // Считываем следующий токен
    if (token == "1" || token == "2" || token == "3") {
        int setIndex = stoi(token) - 1; // Преобразуем токен в индекс
        if (setIndex >= 0 && setIndex < sets.size()) {
            for (int i = 0; i < UNIVERSE_SIZE; ++i) {
                resultSet[i] = (tempSet[i] && !sets[setIndex][i]) || (!tempSet[i] && sets[setIndex][i]); // Симметрическая разность
            }
        }
    }
}
}

```

Рисунок 15 - Симметрическая разность

```

else if (token == "C" || token == "c") { // Если токен - дополнение
    ss >> token; // Считываем следующий токен
    if (token == "1" || token == "2" || token == "3") {
        int setIndex = stoi(token) - 1; // Преобразуем токен в индекс
        if (setIndex >= 0 && setIndex < sets.size()) {
            IntSet complementSet(UNIVERSE_SIZE, false); // Создаем множество дополнения
            for (int i = 0; i < UNIVERSE_SIZE; ++i) {
                complementSet[i] = !sets[setIndex][i]; // Вычисляем дополнение
            }
            for (int i = 0; i < UNIVERSE_SIZE; ++i) {
                resultSet[i] = resultSet[i] || complementSet[i]; // Объединяем с дополнением
            }
        }
    }
}
}

```

Рисунок 16 - Дополнение

```

else if (token == "n") { // Если токен - пересечение
    IntSet tempSet = resultSet; // Сохраняем текущее множество
    ss >> token; // Считываем следующий токен
    if (token == "1" || token == "2" || token == "3") {
        int setIndex = stoi(token) - 1; // Преобразуем токен в индекс
        if (setIndex >= 0 && setIndex < sets.size()) {
            for (int i = 0; i < UNIVERSE_SIZE; ++i) {
                resultSet[i] = tempSet[i] && sets[setIndex][i]; // Вычисляем пересечение
            }
        }
    }
}
}
return resultSet; // Возвращаем результирующее множество
}

```

Рисунок 17 - Пересечение

10. Метод для финальных расчетов и вывода результата

Метод выводит результат вычисления выражения (рис. 18).

```

// Метод вычисления выражения и вывод результата
void SetCalculator::calculateExpression(const string& expression) {
    IntSet result = parseExpression(expression); // Парсим выражение
    cout << "Результат: { ";
    for (int j = 0; j < UNIVERSE_SIZE; ++j) {
        if (result[j]) { // Если элемент существует в результирующем множестве
            cout << (j + UNIVERSE_MIN) << " "; // Выводим элемент
        }
    }
    cout << "}" << endl; // Закрываем множество
}

```

Рисунок 18 - Метод для финальных расчетов и вывода результата

#### 11. Метод для установки условий

Метод устанавливает параметры для ввода элементов во множество (рис. 19).

```

// Метод установки условий для ввода элементов
void SetCalculator::setConditions() {
    cout << "Установите условия для ввода элементов:\n"; // Выводим сообщение
    cout << "1. Разрешить отрицательные числа? (1 - Да, 0 - Нет): ";
    cin >> allowNegative;
    cout << "2. Разрешить четные числа? (1 - Да, 0 - Нет): ";
    cin >> allowEven;
    cout << "3. Разрешить нечетные числа? (1 - Да, 0 - Нет): ";
    cin >> allowOdd;
    cout << "4. Укажите минимальное значение: ";
    cin >> minAllowedValue;
    cout << "5. Укажите максимальное значение: ";
    cin >> maxAllowedValue;
}

```

Рисунок 19 - Метод для установки условий

#### 12. Главная функция

Основная функция запускает цикл, который позволяет пользователю взаимодействовать с программой, выбирая различные действия (рис. 20 - 21).

```

int main() {
    setlocale(LC_ALL, "RU");
    srand(static_cast<unsigned>(time(0)));
    SetCalculator calc;
    int choice;

    while (true) { // Основной цикл программы
        cout << "\nМеню:\n" // Вывод меню
            << "1. Установить условия ввода\n"
            << "2. Добавить случайное множество\n"
            << "3. Добавить ручную множество\n"
            << "4. Отобразить множества\n"
            << "5. Объединение\n"
            << "6. Разность\n"
            << "7. Симметрическая разность\n"
            << "8. Пересечение\n"
            << "9. Дополнение\n"
            << "10. Расчет выражения\n"
            << "11. Выход\n"
            << "Ваш выбор: ";
        cin >> choice; // Читаем выбор пользователя

        switch (choice) { // Обрабатываем выбор
            case 1:
                calc.setConditions(); // Устанавливаем условия
                break;
            case 2:
                calc.addRandomSet(); // Добавляем случайное множество
                break;
            case 3:
                calc.addManualSet(); // Добавляем ручную множество
                break;
            case 4:
                calc.displaySets(); // Отображаем все множества
                break;
            case 5: {
                int a, b;
                cout << "Введите индексы множеств для объединения (1-" << calc.displaySets() << "): ";
                cin >> a >> b;
                if (a > 0 && b > 0 && a <= calc.displaySets() && b <= calc.displaySets()) {
                    calc.unionSets(a - 1, b - 1); // Выполняем объединение
                }
                else {
                    cout << "Некорректные индексы." << endl;
                }
                break;
            }
        }
    }
}

```

Рисунок 20 - Главная функция

```

case 6: {
    int a, b;
    cout << "Введите индексы множеств для разности (1-" << calc.displaySets() << "): ";
    cin >> a >> b;
    if (a > 0 && b > 0 && a <= calc.displaySets() && b <= calc.displaySets()) {
        calc.differenceSets(a - 1, b - 1); // Выполняем разность
    }
    else {
        cout << "Некорректные индексы." << endl;
    }
    break;
}

case 7: {
    int a, b;
    cout << "Введите индексы множеств для симметрической разности (1-" << calc.displaySets() << "): ";
    cin >> a >> b;
    if (a > 0 && b > 0 && a <= calc.displaySets() && b <= calc.displaySets()) {
        calc.symmetricDifference(a - 1, b - 1); // Выполняем симметрическую разность
    }
    else {
        cout << "Некорректные индексы." << endl;
    }
    break;
}

case 8: {
    int a, b;
    cout << "Введите индексы множеств для пересечения (1-" << calc.displaySets() << "): ";
    cin >> a >> b;
    if (a > 0 && b > 0 && a <= calc.displaySets() && b <= calc.displaySets()) {
        calc.intersectionSets(a - 1, b - 1); // Выполняем пересечение
    }
    else {
        cout << "Некорректные индексы." << endl;
    }
    break;
}

case 9: {
    int index;
    cout << "Введите индекс множества для дополнения (1-" << calc.displaySets() << "): ";
    cin >> index;
    if (index > 0 && index <= calc.displaySets()) {
        calc.complementSet(index - 1); // Выполняем дополнение
    }
    else {
        cout << "Некорректный индекс." << endl;
    }
    break;
}

case 10: {
    string expression;
    cout << "Введите выражение (\ - разность, S - симметрическая разность, U - объединение, n - пересечение, C - дополнение (пр. C 1)): ";
    cin.ignore(); // Игнорируем предыдущий ввод
    getline(cin, expression); // Считываем всю строку как выражение
    calc.calculateExpression(expression); // Вычисляем выражение
    break;
}

case 11:
    return 0;
default:
    cout << "Некорректный выбор" << endl;
}
}

```

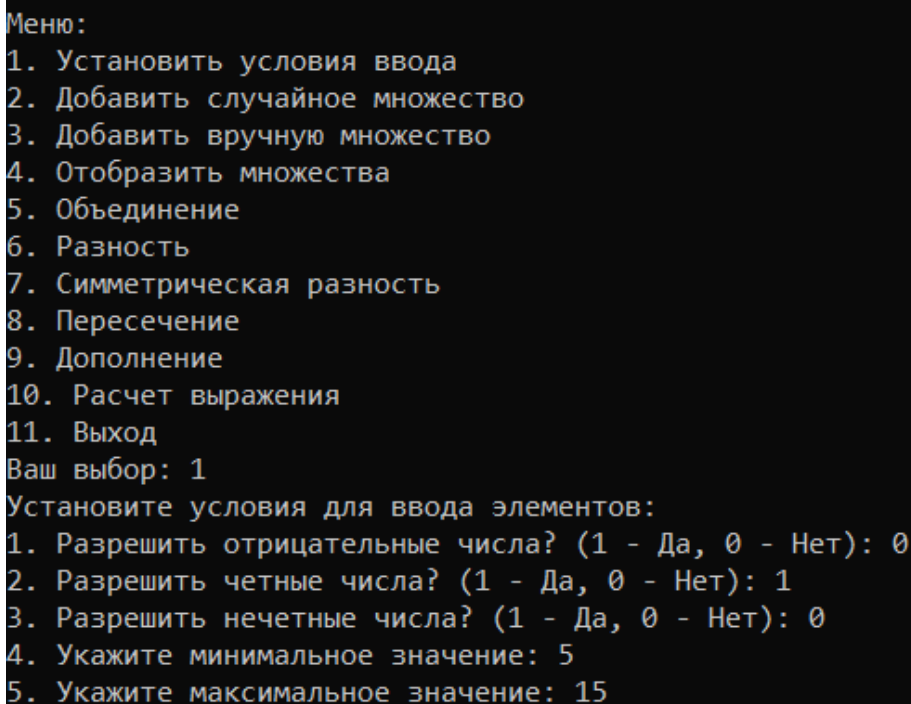
Рисунок 21 - Главная функция (продолжение)

## Тестирование программы

Для проверки корректности работы программы создадим 3 множества и выполним различные операции над ними.

### 1. Создание множества по условиям

Выберем условия и создадим случайное множество. Программа отберет необходимые значения и выведет конечное множество (рис. 22-24).



```
Меню:
1. Установить условия ввода
2. Добавить случайное множество
3. Добавить вручную множество
4. Отобразить множества
5. Объединение
6. Разность
7. Симметрическая разность
8. Пересечение
9. Дополнение
10. Расчет выражения
11. Выход
Ваш выбор: 1
Установите условия для ввода элементов:
1. Разрешить отрицательные числа? (1 - Да, 0 - Нет): 0
2. Разрешить четные числа? (1 - Да, 0 - Нет): 1
3. Разрешить нечетные числа? (1 - Да, 0 - Нет): 0
4. Укажите минимальное значение: 5
5. Укажите максимальное значение: 15
```

Рисунок 22 - Установка условий

Значение 39 не соответствует условиям.  
Значение 5 не соответствует условиям.  
Значение 9 не соответствует условиям.  
Значение -42 не соответствует условиям.  
Значение -31 не соответствует условиям.  
Значение 19 не соответствует условиям.  
Значение 7 не соответствует условиям.  
Значение -10 не соответствует условиям.  
Значение -46 не соответствует условиям.  
Значение 18 не соответствует условиям.  
Значение 18 не соответствует условиям.  
Значение -48 не соответствует условиям.  
Значение 13 не соответствует условиям.  
Значение 13 не соответствует условиям.  
Значение 50 не соответствует условиям.  
Значение -23 не соответствует условиям.  
Значение 16 не соответствует условиям.  
Значение 11 не соответствует условиям.  
Значение -31 не соответствует условиям.  
Значение -35 не соответствует условиям.  
Значение 17 не соответствует условиям.  
Значение -28 не соответствует условиям.  
Значение 27 не соответствует условиям.  
Значение 45 не соответствует условиям.  
Значение 36 не соответствует условиям.  
Значение -40 не соответствует условиям.  
Значение -25 не соответствует условиям.  
Значение 3 не соответствует условиям.  
Значение 39 не соответствует условиям.  
Значение 17 не соответствует условиям.  
Значение -25 не соответствует условиям.  
Значение -20 не соответствует условиям.  
Значение -48 не соответствует условиям.  
Значение -49 не соответствует условиям.  
Значение -40 не соответствует условиям.  
Значение -24 не соответствует условиям.  
Значение 3 не соответствует условиям.  
Значение 16 не соответствует условиям.  
Значение 24 не соответствует условиям.  
Значение 19 не соответствует условиям.  
Значение -8 не соответствует условиям.  
Значение -26 не соответствует условиям.  
Значение -50 не соответствует условиям.

Рисунок 23 - Выборка значений

Меню:  
1. Установить условия ввода  
2. Добавить случайное множество  
3. Добавить ручную множество  
4. Отобразить множества  
5. Объединение  
6. Разность  
7. Симметрическая разность  
8. Пересечение  
9. Дополнение  
10. Расчет выражения  
11. Выход  
Ваш выбор: 4  
Множество 1: { 6 12 }

Рисунок 24 - Отображение множества

## 2. Ручной ввод

Введем вручную элементы множества (рис. 25).

```
Ваш выбор: 3
Введите числа для множества (q для окончания ввода):
12 13 14 15 16 17 18 19 20 q

Меню:
1. Установить условия ввода
2. Добавить случайное множество
3. Добавить ручную множество
4. Отобразить множества
5. Объединение
6. Разность
7. Симметрическая разность
8. Пересечение
9. Дополнение
10. Расчет выражения
11. Выход
Ваш выбор: 4
Множество 1: { 6 12 }
Множество 2: { 12 13 14 15 16 17 18 19 20 }
```

Рисунок 25 - Ручной ввод множества

## 3. Случайное множество

Создадим случайное множество, содержащее значения из диапазона от -50 до 50 (рис. 26).

```
Меню:
1. Установить условия ввода
2. Добавить случайное множество
3. Добавить ручную множество
4. Отобразить множества
5. Объединение
6. Разность
7. Симметрическая разность
8. Пересечение
9. Дополнение
10. Расчет выражения
11. Выход
Ваш выбор: 4
Множество 1: { 6 12 }
Множество 2: { 12 13 14 15 16 17 18 19 20 }
Множество 3: { -43 -39 -38 -37 -35 -34 -33 -32 -31 -29 -27 -25 -24 -20 -17 -16 -15 -14 -12 -9 -8 -7 -5 -3 -1 0 1 2 3 5 6
7 11 12 13 14 15 16 19 20 22 27 28 29 30 31 32 33 34 35 36 39 42 44 45 46 49 50 }
```

Рисунок 26 - Создание случайного множества

## 4. Операция объединения

Выполним операцию объединения над множествами 1 и 2 (рис. 27).

```
Введите индексы множеств для объединения (1-3): 1 2
Множество 1: { 6 12 }
Множество 2: { 12 13 14 15 16 17 18 19 20 }
Множество 3: { -43 -39 -38 -37 -35 -34 -33 -32 -31 -29 -27 -25 -24 -20 -17 -16 -15 -14 -12 -9 -8 -7 -5 -3 -1 0 1 2 3 5 6
7 11 12 13 14 15 16 19 20 22 27 28 29 30 31 32 33 34 35 36 39 42 44 45 46 49 50 }
Множество 1: { 6 12 }
Множество 2: { 12 13 14 15 16 17 18 19 20 }
Множество 3: { -43 -39 -38 -37 -35 -34 -33 -32 -31 -29 -27 -25 -24 -20 -17 -16 -15 -14 -12 -9 -8 -7 -5 -3 -1 0 1 2 3 5 6
7 11 12 13 14 15 16 19 20 22 27 28 29 30 31 32 33 34 35 36 39 42 44 45 46 49 50 }
Объединение множеств: { 6 12 13 14 15 16 17 18 19 20 }
```

Рисунок 27 - Объединение двух множеств



## 5. Операция дополнения

Дополним множество 3 до универсума (рис. 28).

```
Введите индекс множества для дополнения (1-3): 3
Множество 1: { 6 12 }
Множество 2: { 12 13 14 15 16 17 18 19 20 }
Множество 3: { -43 -39 -38 -37 -35 -34 -33 -32 -31 -29 -27 -25 -24 -20 -17 -16 -15 -14 -12 -9 -8 -7 -5 -3 -1 0 1 2 3 5 6
7 11 12 13 14 15 16 19 20 22 27 28 29 30 31 32 33 34 35 36 39 42 44 45 46 49 50 }
Дополнение до универсума: { -50 -49 -48 -47 -46 -45 -44 -42 -41 -40 -36 -30 -28 -26 -23 -22 -21 -19 -18 -13 -11 -10 -6 -
4 -2 4 8 9 10 17 18 21 23 24 25 26 37 38 40 41 43 47 48 }
```

Рисунок 28 - Дополнение до универсума

## 6. Операция разности

Выполним разность множеств 2 и 1 (рис. 29).

```
Множество 1: { 6 12 }
Множество 2: { 12 13 14 15 16 17 18 19 20 }
Множество 3: { -43 -39 -38 -37 -35 -34 -33 -32 -31 -29 -27 -25 -24 -20 -17 -16 -15 -14 -12 -9 -8 -7 -5 -3 -1 0 1 2 3 5 6
7 11 12 13 14 15 16 19 20 22 27 28 29 30 31 32 33 34 35 36 39 42 44 45 46 49 50 }
Разность множеств: { 13 14 15 16 17 18 19 20 }
```

Рисунок 30 - Разность множеств

## 7. Операция симметрической разности

Выполним симметрическую разность множеств 1 и 2 (рис. 30).

```
Множество 1: { 6 12 }
Множество 2: { 12 13 14 15 16 17 18 19 20 }
Множество 3: { -43 -39 -38 -37 -35 -34 -33 -32 -31 -29 -27 -25 -24 -20 -17 -16 -15 -14 -12 -9 -8 -7 -5 -3 -1 0 1 2 3 5 6
7 11 12 13 14 15 16 19 20 22 27 28 29 30 31 32 33 34 35 36 39 42 44 45 46 49 50 }
Симметрическая разность: { 6 13 14 15 16 17 18 19 20 }
```

Рисунок 30 - Симметрическая разность множеств

## 8. Операция пересечения

Выполним операцию пересечения для множеств 1 и 2 (рис. 31).

```
Множество 1: { 6 12 }
Множество 2: { 12 13 14 15 16 17 18 19 20 }
Множество 3: { -43 -39 -38 -37 -35 -34 -33 -32 -31 -29 -27 -25 -24 -20 -17 -16 -15 -14 -12 -9 -8 -7 -5 -3 -1 0 1 2 3 5 6
7 11 12 13 14 15 16 19 20 22 27 28 29 30 31 32 33 34 35 36 39 42 44 45 46 49 50 }
Пересечение множеств: { 12 }
```

Рисунок 31 - Пересечение множеств

## 9. Формула

Введем формулу  $1 \cap 2 \cup (-3)$  (Объединение пересечения множеств 1 и 2 с дополнением множества 3) (рис. 32).

```
Введите выражение ( - разность, S - симметрическая разность, U - объединение, п - пересечение, С - дополнение (пр. С 1))
: 1 п 2 U С 3
Результат: { -43 -39 -38 -37 -35 -34 -33 -32 -31 -29 -27 -25 -24 -20 -17 -16 -15 -14 -12 -9 -8 -7 -5 -3 -1 0 1 2 3 5 6
11 12 13 14 15 16 19 20 22 27 28 29 30 31 32 33 34 35 36 39 42 44 45 46 49 50 }
```

Рисунок 32 - Расчет формулы

## **Заключение**

Был разработан калькулятор множеств, позволяющий создавать множества тремя разными способами, выполнять операции над множествами и рассчитывать выражения, состоящие из множеств. Программа была протестирована на корректность.

## Список используемой литературы

1. Теория множеств: основы и базовые операции над множествами URL: <https://ru.hexlet.io/blog/posts/teoriya-mnozhestv-osnovy-i-bazovye-operatsii-nad-mnozhestvami> ( дата обращения: 02.10.24)
2. Множества. Операции над множествами. Отображение множеств. Мощность множества URL: <http://www.mathprofi.ru/mnozhestva.html> (дата обращения: 03.10.24)