Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

**Пояснительная записка к курсовой работе**
по курсу «Логика и основы алгоритмизации в инженерных задачах»
на тему «Реализация операции нахождения пересечения двух и более множеств»

25.12.25

Выполнил:
Студент группы 24ВВВ1
Выдрин Д.А.

Принял:
к.т.н., доцент Юрова О. В.

Пенза 2025

Пенза 2025

**ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ _____

«____»_____20__

**ЗАДАНИЕ**

**на курсовое проектирование по курсу**

"Логика и основы алгоритмизации в инженерных задачах"

Студенту Вадрину Дмитрию Алексеевичу Группа 24ВВВ1

Тема проекта Реализация нахождения пересечений двух и более множеств

Исходные данные (технические требования) на проектирование

Разработка алгоритмов и программного обеспечения в соответствии с данным заданием курсового проекта. Пояснительная записка должна содержать:

1. Постановку задачи
2. Теоретическую часть задания
3. Описание алгоритма поставленной задачи
4. Пример ручного расчёта задачи и вычислений
5. Описание самой программы
6. Тесты
7. Список литературы
8. Листинг программы
9. Результат работы программы.

**Объем работы по курсу**

1. Расчетная часть

Ручной расчет работы алгоритма

2. Графическая часть

Схема алгоритма в формате блок-схем

3. Экспериментальная часть

Тестирование программы
Результата работы программы по тестовым данным

Срок выполнения проекта по разделам

1 Исследование теоретической части курсовой
2 Разработка алгоритма программы
3 Разработка программы
4 Тестирование и завершение разработки программы
5 Оформление пояснительной записки
6
7
8

Дата выдачи задания "12" сентября 2025 г.

Дата защиты проекта "___" _____

**Руководитель** Юдова О.В.

**Задание получил** "25" октября 2025 г.

**Студент** Входин Дмитрий Алексеевич

## Содержание

# Реферат

Отчёт содержит, 28 страниц, 17 рисунков ,2 таблицы

МНОЖЕСТВО, ПЕРЕСЕЧЕНИЕ МНОЖЕСТВ, СТРУКТУРА ДАННЫХ, ФАЙЛОВЫЕ ОПЕРАЦИИ

Цель исследования - разработка программы на языке C++, реализующей нахождение пересечения двух и более множеств с поддержкой различных способов заполнения данных, сохранения и загрузки результатов.

В работе рассмотрены теоретические основы операций над множествами. Разработана программа с консольным интерфейсом, позволяющая создавать множества вручную или случайным образом, находить их пересечения, сохранять и загружать данные из файлов. Реализованный алгоритм корректно находит пересечение множеств для различных наборов данных.

# Введение

Операции над множествами — фундаментальная тема в математике и информатике. Пересечение множеств — одна из основных операций теории множеств, которая находит элементы, принадлежащие одновременно всем рассматриваемым множествам. Эта операция имеет широкое практическое применение в различных областях:

- анализ данных и статистика;
- информационный поиск и фильтрация;
- работа с базами данных;
- машинное обучение и обработка естественного языка;
- решение логических задач.

Для написания программы использовалась среда разработки Microsoft Visual Studio 2022 года. Это современная и удобная программа для создания приложений на C++, которая предоставляет все необходимые инструменты для написания, отладки и тестирования кода.

Целью данной курсовой работы является создание программы, которая:

1. Позволяет работать с множествами целых чисел
2. Реализует алгоритм нахождения пересечения двух и более множеств
3. Обеспечивает сохранение и загрузку данных через файлы
4. Предлагает интуитивно понятный пользовательский интерфейс

## 1. Постановка задачи

Требуется разработать программу на языке C++, реализующую нахождение пересечения двух и более множеств целых чисел.

Множества для обработки могут быть созданы несколькими способами. Во-первых, их можно ввести вручную через клавиатуру. Во-вторых, программа позволяет заполнить множества случайными числами с указанием диапазона значений. В-третьих, предусмотрена возможность загрузки данных из заранее подготовленного текстового файла. Для корректной работы необходимо обеспечить уникальность элементов внутри каждого множества.

Программа должна выполнять несколько основных задач. Самое главное — правильно находить пересечение множеств, то есть элементы, которые присутствуют во всех рассматриваемых множествах. Управление осуществляется через простое текстовое меню. На экран можно вывести все созданные множества, а также результат работы алгоритма — элементы пересечения. Все данные и результаты можно сохранять в файлы для последующего использования.

Итоговый набор функций программы включает: создание множеств различными способами, работу с файлами (загрузку и сохранение), отображение множеств, нахождение пересечения двух или нескольких множеств. Пользователь работает с программой через клавиатуру, а все результаты видит в консоли или в сохранённых текстовых файлах.

Задания выполняются в соответствии с вариантом №8.

## 2.Теоретическая часть

Множество — это фундаментальное понятие математики, представляющее собой набор различных объектов, рассматриваемых как единое целое. В контексте данной работы рассматриваются конечные множества целых чисел.

Пересечение множеств — это операция над множествами, результатом которой является новое множество, содержащее элементы, которые принадлежат одновременно всем исходным множествам. Обозначается символом ∩.

Основные свойства пересечения множеств:

1. Коммутативность: $A \cap B = B \cap A$
2. Ассоциативность: $(A \cap B) \cap C = A \cap (B \cap C)$
3. Идемпотентность: $A \cap A = A$
4. Поглощение: $A \cap (A \cup B) = A$
5. Дистрибутивность пересечения относительно объединения: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

В программе множества представлены с помощью структуры данных Set, которая содержит массив элементов и размер множества. Для обеспечения уникальности элементов используется проверка на наличие элемента перед его добавлением.

Алгоритм нахождения пересечения двух множеств:

1. Инициализация пустого результирующего множества
2. Для каждого элемента первого множества:

– Проверка наличия этого элемента во втором множестве

– Если элемент присутствует в обоих множествах, добавление его в результат

3. Возврат результирующего множества

Для нахождения пересечения нескольких множеств используется последовательное применение алгоритма:

1. Найти пересечение первых двух множеств
2. Результат пересечь с третьим множеством
3. Продолжать до обработки всех множеств

Особенности реализации:

- Использование функции hasElement() для проверки наличия элемента
- Функция addElement() обеспечивает уникальность элементов
- Поддержка различных способов ввода данных
- Работа с файлами для сохранения состояния

# 3. Описание программы

## 3.1 Структура программы

Основной модуль программы содержит функции для работы с множествами и предоставляет пользовательский интерфейс в виде консольного меню. Через это меню осуществляется управление всеми функциями программы.

Структура данных Set определена в основном файле программы. Она представляет собой структуру, содержащую массив целых чисел для хранения элементов и переменную для отслеживания текущего размера множества. Максимальный размер множества ограничен константой MAX_ELEMENTS.

Ключевые функции программы:

1. hasElement() — проверяет наличие элемента в множестве
2. addElement() — добавляет элемент в множество с проверкой уникальности
3. printSet() — выводит множество на экран
4. inputSet() — заполняет множество вручную
5. randomSet() — заполняет множество случайными числами
6. intersectSets() — находит пересечение двух множеств
7. saveToFile() — сохраняет данные в файл
8. loadFromFile() — загружает данные из файла

Такая организация кода обеспечивает чёткое разделение функциональности, что делает программу понятной, легко поддерживаемой и расширяемой. Визуальное представление структуры программы приведено в Приложении Б (Диаграмма классов).

### 3.2 Описание алгоритма

АЛГОРИТМ: Нахождение пересечения двух множеств

ВХОД:

a - первое множество (объект типа Set)

b - второе множество (объект типа Set)

ВЫХОД:

result - множество-пересечение (объект типа Set)

ПЕРЕМЕННЫЕ:

result - результирующее множество

i - счётчик для итерации по элементам

АЛГОРИТМ:

1. ИНИЦИАЛИЗАЦИЯ:
   1.1 Создать пустое множество result
   (result.size = 0)
2. ОСНОВНОЙ ЦИКЛ:
   ДЛЯ i = 0 ДО a.size - 1 ВЫПОЛНЯТЬ:
   2.1 element = a.elements[i] // текущий элемент первого множества
   2.2 ЕСЛИ hasElement(b, element) = true:
   // элемент присутствует во втором множестве
   2.2.1 addElement(result, element)
3. ВОЗВРАТ result

КОНЕЦ АЛГОРИТМА

АЛГОРИТМ: Нахождение пересечения нескольких множеств

ВХОД:

sets[] - массив множеств

count - количество множеств

ВЫХОД:

result - пересечение всех множеств

АЛГОРИТМ:

1. ПРОВЕРКА:

   ЕСЛИ count = 0:

   ВОЗВРАТ пустое множество

ЕСЛИ count = 1:

ВОЗВРАТ sets[0]

2. ИНИЦИАЛИЗАЦИЯ:

   2.1 result = intersectSets(sets[0], sets[1])

3. ОСНОВНОЙЦИКЛ:

   ДЛЯ i = 2 ДО count - 1 ВЫПОЛНЯТЬ:

   3.1 result = intersectSets(result, sets[i])

4. ВОЗВРАТ result

КОНЕЦ АЛГОРИТМА

Особенности реализации:

- Функция hasElement() использует линейный поиск для проверки наличия элемента
- Функция addElement() предотвращает добавление дубликатов
- Алгоритм работает за O(n*m) времени для двух множеств, где n и m — размеры множеств
- Память используется эффективно — только для хранения результата

### 3.3 Интерфейс программы

После запуска программы пользователь видит следующее главное меню (Рисунок 1):

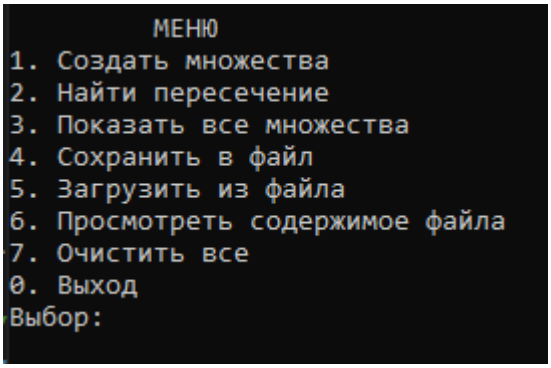


Рисунок 1 – Меню

Каждый пункт выполняет определённую функцию и возвращает пользователя обратно в меню после завершения операции (Рисунок 2,3,4).

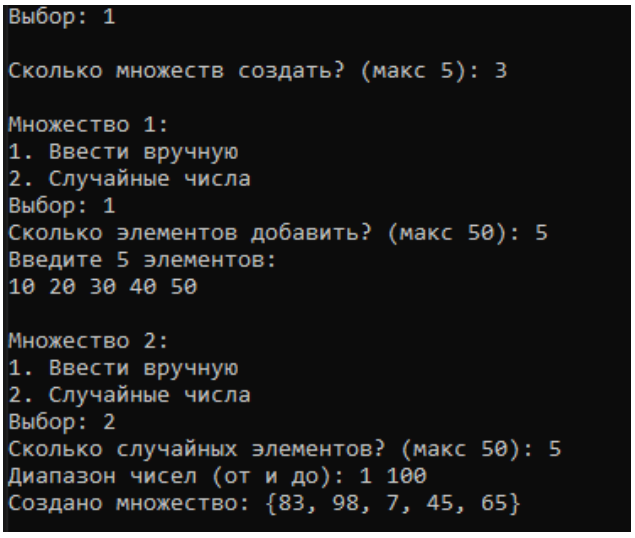


Рисунок 2 – Создание множеств различными способами

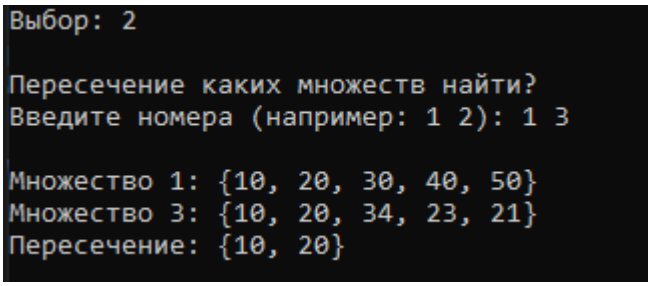


Рисунок 3 – Нахождение пересечения двух множеств

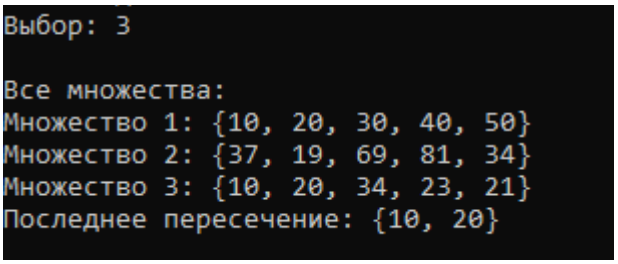


Рисунок 4 – Отображение всех множеств

Программа показывает текущие множества, а также последние пересечения.

Пользователь может сохранить множества и их пересечения в отдельный файл(Рисунок 5,6,7):

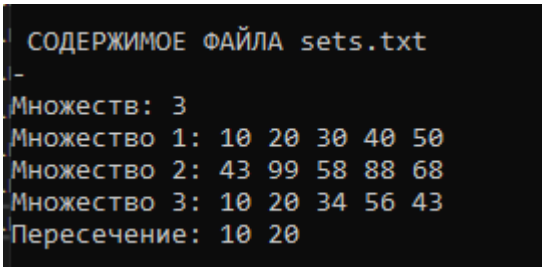


Рисунок 5 – Сохранение данных в файл

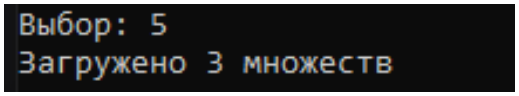


Рисунок 6 –Загрузка данных из файла

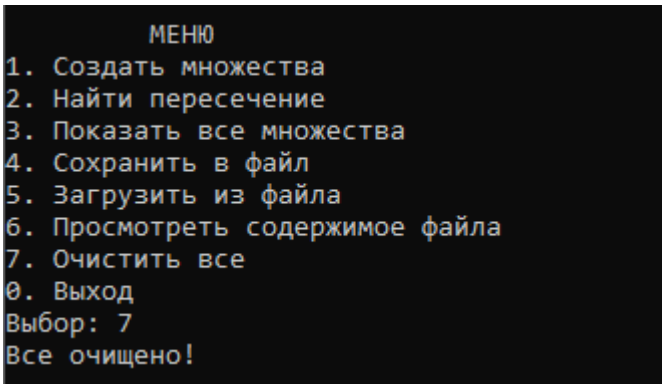


Рисунок 7 – Очистка данных

После выбора этого пункта все созданные множества и результаты очищаются, что позволяет начать работу с чистого листа.

## 4. Тестирование

Таблица 1 - план тестирования.

| № | Тест | Входные данные | Ожидаемый результат |
|---|---|---|---|
| 1 | Запуск программы | - | Отображение заголовка и главного меню |
| 2 | Создание множеств вручную | 2 множества по 3 элемента | Корректное создание множеств |
| 3 | Создание множеств случайными числами | 1 множество, 5 элементов, диапазон 1-10 | Создание множества с уникальными числами |
| 4 | Нахождение пересечения двух множеств | {1,2,3,4,5} и {3,4,5,6,7} | {3,4,5} |
| 5 | Нахождение пересечения без общих элементов | {1,2,3} и {4,5,6} | Пустое множество {} |
| 6 | Нахождение пересечения трёх множеств | {1,2,3}, {2,3,4}, {3,4,5} | {3} |
| 7 | Сохранение данных в файл | 2 множества, имя файла test.txt | Файл создан с корректными данными |

Продолжение таблицы 1.

| № | Тест | Входные данные | Ожидаемый результат |
|---|---|---|---|
| 8 | Загрузка данных из файла | Файл test.txt с сохранёнными данными | Успешная загрузка множеств |
| 9 | Очистка всех данных | - | Все множества удалены |
| 10 | Некорректный ввод | Буквы вместо чисел | Сообщение об ошибке и повторный запрос |
| 11 | Загрузка несуществующего файла | Имя несуществующего файла | Сообщение об ошибке |
| 12 | Поиск пересечения без создания множеств | Выбор пункта 2 без данных | Сообщение о необходимости создать множества |

Полный цикл работы программы представлен на рисунках 8,9,10,11,12,13,14.

```
Курсовая работа по дисциплине
"Логика и основы алгоритмизации в инженерия задач"
на тему "Реализация нахождения пересечения двух и более множеств"

Выполнил студент группы 248ВВ1: Выдрин Дмитрий
Приняла: Юрова О.В.
```

Рисунок 8 – Запуск программы

```
        МЕНЮ
1. Создать множества
2. Найти пересечение
3. Показать все множества
4. Сохранить в файл
5. Загрузить из файла
6. Просмотреть содержимое файла
7. Очистить все
0. Выход
Выбор:
```

Рисунок 9 – Меню программы

Результат: Меню отображается корректно.

1. Выбор пункта 1.

```
Сколько множеств создать? (макс 5): 2

Множество 1:
1. Ввести вручную
2. Случайные числа
Выбор: 1
Сколько элементов добавить? (макс 50): 5
Введите 5 элементов:
10 20 30 40 50

Множество 2:
1. Ввести вручную
2. Случайные числа
Выбор: 2
Сколько случайных элементов? (макс 50): 10
Диапазон чисел (от и до): 1 50
Создано множество: {21, 42, 6, 31, 43, 18, 22, 33, 40, 38}
```

Рисунок 10 – Создание двух множеств разными способами

Результат: Множества успешно созданы.

2. Выбор пункта 3.

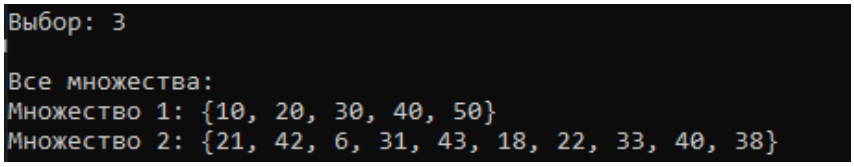


Рисунок 11 – Отображение созданных множеств

Результат: Множества отображаются корректно.

3. Выбор пункта 2.

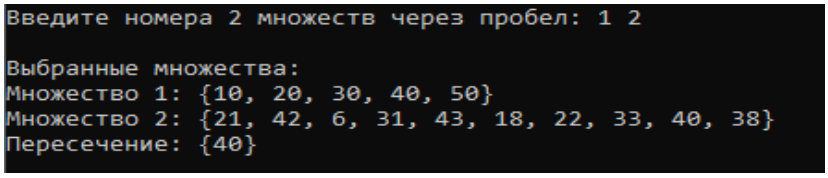


Рисунок 12 – Результат пересечения множеств

Результат: Алгоритм правильно нашел общий элемент.

4. Выбор пункта 4.

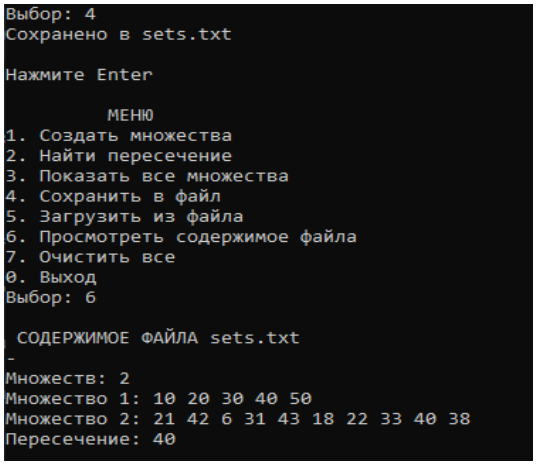


Рисунок 13 – Сохранение данных

Результат: Данные сохранены корректно.

5. Загрузка из файла.

Очистка данных (пункт 7), затем загрузка:

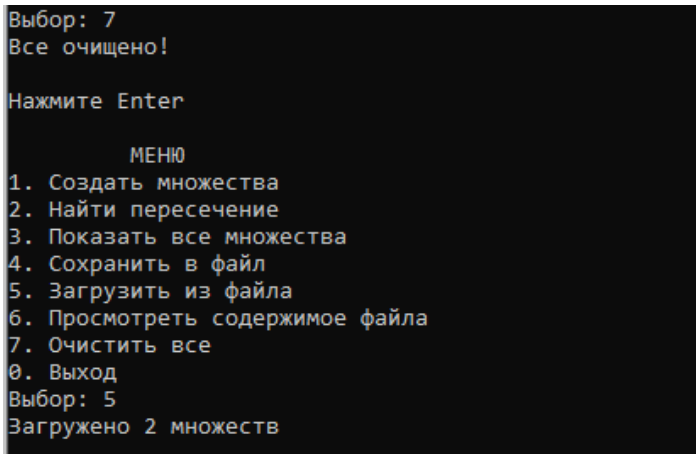


Рисунок 14 – Загрузка данных из файла

Результат: Данные успешно загружены.

6. Обработка ошибок представлены на рисунках 15,16,17.

1. Некорректный ввод данных:

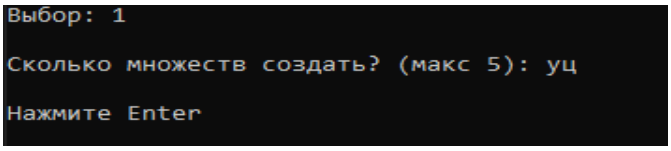


Рисунок 15 – Обработка некорректного ввода

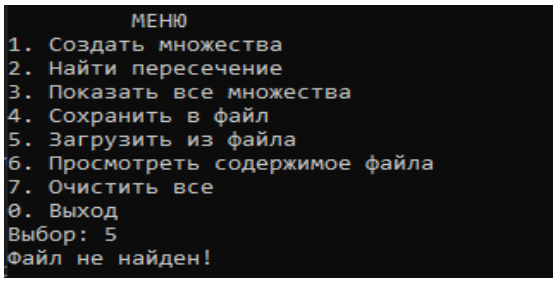


Рисунок 16 – Обработка ошибки загрузки файла

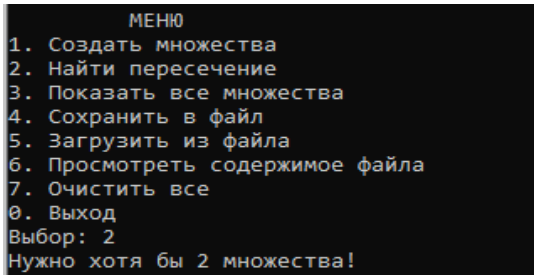


Рисунок 17 – Обработка попытки работы без данных

Результат: Защита от выполнения без данных работает.

Таблица 2 – результаты поведения программы при тестировании.

| № | Тест | Полученный результат |
|---|---|---|
| 1 | Запуск программы | Верно |
| 2 | Создание множеств вручную | Верно |
| 3 | Создание множеств случайными числами | Верно |
| 4 | Нахождение пересечения двух множеств | Верно |
| 5 | Нахождение пересечения без общих элементов | Верно |
| 6 | Нахождение пересечения трёх множеств | Верно |
| 7 | Сохранение данных в файл | Верно |
| 8 | Загрузка данных из файла | Верно |
| 9 | Очистка всех данных | Верно |
| 10 | Некорректный ввод | Верно |
| 11 | Загрузка несуществующего файла | Верно |
| 12 | Поиск пересечения без создания множеств | Верно |

## 5. Ручной расчёт

Дано:

Множество A = {1, 2, 3, 4, 5}

Множество B = {2, 3, 5, 7, 11}

Множество C = {3, 5, 8, 13, 21}

Ручной расчёт:

Находим A ∩ B:

Элементы A: {1, 2, 3, 4, 5}

Проверяем каждый элемент на наличие в B:

1 ∈ B? Нет → не добавляем

2 ∈ B? Да → добавляем 2

3 ∈ B? Да → добавляем 3

4 ∈ B? Нет → не добавляем

5 ∈ B? Да → добавляем 5

Промежуточный результат: A ∩ B = {2, 3, 5}

Находим (A ∩ B) ∩ C:

Элементы A ∩ B: {2, 3, 5}

Проверяем каждый элемент на наличие в C:

2 ∈ C? Нет → не добавляем

3 ∈ C? Да → добавляем 3

5 ∈ C? Да → добавляем 5

Итоговый результат: A ∩ B ∩ C = {3, 5}

Пример 2:

Дано:

Множество A = {2, 4, 6, 8, 10, 12}

Множество B = {3, 6, 9, 12, 15}

Ручной расчёт:

Шаг 1:Берем первый элемент множества A: 2

Проверяем: 2 ∈ B? Нет (2 нет в множестве B)

Шаг 2: Берем второй элемент множества A: 4

Проверяем: 4 ∈ B? Нет (4 нет в множестве B)

Шаг 3:Берем третий элемент множества A: 6

Проверяем: 6 ∈ B? Да (6 есть в множестве B: {3, 6, 9, 12, 15})

Шаг 4:Берем четвертый элемент множества A: 8

Проверяем: 8 ∈ B? Нет (8 нет в множестве B)

Шаг 5:Берем пятый элемент множества A: 10

Проверяем: 10 ∈ B? Нет (10 нет в множестве B)

Шаг 6:Берем шестой элемент множества A: 12

Проверяем: 12 ∈ B? Да (12 есть в множестве B: {3, 6, 9, 12, 15})

Итог:A ∩ B = {6, 12}

# Заключение

В ходе выполнения курсовой работы мною была разработана программа на языке C++, реализующая нахождение пересечения двух и более множеств целых чисел.

Программа предоставляет пользователю удобный консольный интерфейс, позволяющий создавать множества различными способами (ручной ввод, случайная генерация), находить их пересечения, сохранять и загружать данные из файлов. Все поставленные цели достигнуты: изучены теоретические основы операций над множествами, разработано программное обеспечение с чёткой структурой, проведено тестирование, подтвердившее корректность работы программы.

В процессе работы были приобретены и углублены практические навыки программирования на C++, включая работу со структурами данных, файловыми операциями и обработкой пользовательского ввода.

Программа имеет достаточный функционал для решения учебных и практических задач, связанных с операциями над множествами. В качестве возможных улучшений можно отметить добавление других операций над множествами (объединение, разность, симметрическая разность) и реализацию графического интерфейса.

## Список использованных источников

1. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: Построение и анализ - М.: МЦНМО, 2001 - 960 с.

2. Подбельский В.В. «Язык С++ » 5-е издание 2001г.

3. Уилсон Р. Введение в теорию графов. Пер. с анг. 1977 208 с.

4. Харви Дейтел, Пол Дейтел. Как программировать на C/C++. 2009 г.

# Приложение А (Листинг программы)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <locale.h>

#define MAX_ELEMENTS 50
#define MAX_SETS 5
#define FILENAME "sets.txt"

typedef struct {
    int elements[MAX_ELEMENTS];
    int size;
} Set;

void clearBuffer() {
while (getchar() != '\n');
}

int hasElement(Set* set, int element) {
    for (int i = 0; i < set->size; i++) {
        if (set->elements[i] == element) {
            return 1;
        }
    }
    return 0;
}

void addElement(Set* set, int element) {
    if (set->size < MAX_ELEMENTS && !hasElement(set, element)) {
        set->elements[set->size] = element;
        set->size++;
    }
}

void printSet(Set* set) {
    printf("{");
    for (int i = 0; i < set->size; i++) {
        printf("%d", set->elements[i]);
        if (i < set->size - 1) printf(", ");
    }
    printf("}\n");
}

void inputSet(Set* set) {
    set->size = 0;
    int count;

    printf("Сколько элементов добавить? (макс %d): ", MAX_ELEMENTS);
    scanf("%d", &count);
    clearBuffer();

    if (count > MAX_ELEMENTS) count = MAX_ELEMENTS;

printf("Введите %d элементов:\n", count);
for (int i = 0; i < count; i++) {
        int element;
        scanf("%d", &element);
        addElement(set, element);
    }
    clearBuffer();
}
```

```
void randomSet(Set* set) {
    set->size = 0;
    int count;

    printf("Сколько случайных элементов? (макс %d): ", MAX_ELEMENTS);
    scanf("%d", &count);
    clearBuffer();

    if (count > MAX_ELEMENTS) count = MAX_ELEMENTS;

printf("Диапазон чисел (от и до): ");
int min, max;
    scanf("%d %d", &min, &max);
    clearBuffer();

    for (int i = 0; i < count; i++) {
        int num = min + rand() % (max - min + 1);
        addElement(set, num);
    }

    printf("Создано множество: ");
    printSet(set);
}
Set intersectMultipleSets(Set sets[], int indices[], int count) {
    Set result;
    result.size = 0;

    if (count < 2) {
        if (count == 1) {
            return sets[indices[0]];
        }
        return result;
    }

    Set temp;
    temp.size = 0;

    for (int i = 0; i < sets[indices[0]].size; i++) {
        addElement(&temp, sets[indices[0]].elements[i]);
    }

    for (int i = 1; i < count; i++) {
        Set newTemp;
        newTemp.size = 0;

        for (int j = 0; j < temp.size; j++) {
            if (hasElement(&sets[indices[i]], temp.elements[j])) {
                addElement(&newTemp, temp.elements[j]);
            }
        }

        temp = newTemp;
    }

    return temp;
}

void saveToFile(Set sets[], int count, Set result) {
    FILE* file = fopen(FILENAME, "w");
if (!file) {
        printf("Ошибка сохранения!\n");
return;
    }
```

```
    fprintf(file, "Множеств: %d\n", count);
    for (int i = 0; i < count; i++) {
        fprintf(file, "Множество %d: ", i + 1);
        for (int j = 0; j < sets[i].size; j++) {
            fprintf(file, "%d ", sets[i].elements[j]);
        }
        fprintf(file, "\n");
    }

    fprintf(file, "Пересечение: ");
    for (int i = 0; i < result.size; i++) {
        fprintf(file, "%d ", result.elements[i]);
    }

    fclose(file);
    printf("Сохраненов %s\n", FILENAME);
}

int loadFromFile(Set sets[], int* count, Set* result) {
    FILE* file = fopen(FILENAME, "r");
    if (!file) return 0;

    fscanf(file, "Множеств: %d\n", count);
    for (int i = 0; i < *count; i++) {
        sets[i].size = 0;
        fscanf(file, "Множество %*d: ");

        int element;
        while (fscanf(file, "%d", &element) == 1) {
            char c = fgetc(file);
            sets[i].elements[sets[i].size] = element;
            sets[i].size++;
            if (c == '\n') break;
        }
    }

    result->size = 0;
    fscanf(file, "Пересечение: ");
    int element;
    while (fscanf(file, "%d", &element) == 1) {
        result->elements[result->size] = element;
        result->size++;
    }

    fclose(file);
    return 1;
}

void viewFile() {
    FILE* file = fopen(FILENAME, "r");
    if (!file) {
        printf("Файл %s не найден!\n", FILENAME);
        return;
    }

    printf("\n СОДЕРЖИМОЕ ФАЙЛА %s \n", FILENAME);
    printf("-\n");

    char line[256];
    while (fgets(line, sizeof(line), file)) {
        printf("%s", line);
    }
```

```
    printf("\n");
    fclose(file);
}

void printTitlePage() {
    printf("\n");
printf("Курсовая работа по дисциплине\n");
    printf("\"Логика и основы алгоритмизации в инженерия задач\"\n");
    printf("на тему \"Реализация нахождения пересечения двух и более множеств\"\n");
    printf("\n");
    printf("Выполнил студент группы 248ВВ1: Выдрин Дмитрий\n");
    printf("Приняла: Юрова О.В.\n");
printf("\n");
}

int main() {
    Set sets[MAX_SETS];
    Set result;
    int setCount = 0;

    setlocale(LC_ALL, "RUSSIAN");

    srand(time(NULL));

    printTitlePage();

    printf("Нажмите Enter");
    clearBuffer();

    while (1) {
        printf("\n         МЕНЮ \n");
printf("1. Создать множества\n");
        printf("2. Найти пересечение\n");
        printf("3. Показать все множества\n");
        printf("4. Сохранить в файл\n");
        printf("5. Загрузить из файла\n");
        printf("6. Просмотреть содержимое файла\n");
        printf("7. Очистить все\n");
        printf("0. Выход\n");
        printf("Выбор: ");

        int choice;
        scanf("%d", &choice);
        clearBuffer();

        switch (choice) {
        case 0:
            printf("Выход\n");
            return 0;

case 1: {
            printf("\nСколько множеств создать? (макс %d): ", MAX_SETS);
            scanf("%d", &setCount);
            clearBuffer();

            if (setCount > MAX_SETS) setCount = MAX_SETS;

            for (int i = 0; i < setCount; i++) {
                printf("\nМножество %d:\n", i + 1);
                printf("1. Ввести вручную\n");
                printf("2. Случайные числа\n");
printf("Выбор: ");

                int method;
```

```
                scanf("%d", &method);
                clearBuffer();

                sets[i].size = 0;

                if (method == 1) {
                    inputSet(&sets[i]);
                }
                else {
                    randomSet(&sets[i]);
                }
            }
            break;
        }

        case 2:
            if (setCount < 2) {
                printf("Нужно хотя бы 2 множества!\n");
break;
            }

            printf("\nПересечение скольких множеств найти? (от 2 до %d): ", setCount);
            int intersectCount;
            scanf("%d", &intersectCount);
            clearBuffer();

            if (intersectCount < 2 || intersectCount > setCount) {
printf("Неверное количество!\n");
                break;
            }

            int indices[MAX_SETS];
            printf("Введите номера %d множеств через пробел: ", intersectCount);
for (int i = 0; i < intersectCount; i++) {
                scanf("%d", &indices[i]);
                indices[i]--;

                if (indices[i] < 0 || indices[i] >= setCount) {
printf("Неверный номер множества!\n");
clearBuffer();
                    intersectCount = -1;
                    break;
                }
            }
            clearBuffer();

            if (intersectCount == -1) break;

            result = intersectMultipleSets(sets, indices, intersectCount);

            printf("\nВыбранные множества:\n");
            for (int i = 0; i < intersectCount; i++) {
                printf("Множество %d: ", indices[i] + 1);
                printSet(&sets[indices[i]]);
            }
            printf("Пересечение: ");
            printSet(&result);
            break;

        case 3:
            if (setCount == 0) {
printf("Множеств нет!\n");
            }
            else {
```

```
printf("\nВсемножества:\n");
                for (int i = 0; i < setCount; i++) {
                    printf("Множество %d: ", i + 1);
                    printSet(&sets[i]);
}
                if (result.size > 0) {
                    printf("Последнее пересечение: ");
printSet(&result);
                }
            }
            break;

        case 4:
            if (setCount > 0) {
                saveToFile(sets, setCount, result);
            }
            else {
printf("Нет данных для сохранения!\n");
}
            break;

        case 5:
            if (loadFromFile(sets, &setCount, &result)) {
printf("Загружено %d множеств\n", setCount);
            }
            else {
                printf("Файл не найден!\n");
}
            break;

        case 6:
            viewFile();
            break;

        case 7:
            setCount = 0;
            result.size = 0;
            printf("Все очищено!\n");
            break;

        default:
            printf("Неверный выбор!\n");
        }

        printf("\nНажмите Enter");
clearBuffer();
    }

    return 0;
}
```