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

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проектированию
по курсу «Логика и основы алгоритмизации в инженерных задачах»
на тему «Реализация операции нахождения объединения двух и более множеств»

25.12.25
хорошо

Выполнил: ст. гр. 24ВВВ1
Картушин Р.А
Принял: к.т.н. доцент
Юрова О.В

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 Карпушину Роману Алексеевичу Группа 24ВВВ1

Тема_проекта Реализация операции нахождения объединения 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. Постановка задачи;
2. Теор. часть задания;
3. Описание алгоритма кой. задачи.
4. Пример ручного расчета задачи и вычислений.
5. Описание самой программы;
6. Тесты;
7. Список литературы;
8. Листинг программы;
9. Результаты 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
результаты работы программы на
тестовых данных.

Срок выполнения проекта по разделам

1 Изучение теор. части Курс. проекта
2 Разработка алгоритмов программы
3 Разработка программы
4 Тестирование и завершение разработки
5 программы
6 Оформление пояснительной записки
7
8

Дата выдачи задания "12" сентября 2025

Дата защиты проекта "___" _____

Руководитель Юрова О.В

Задание получил "25" сентября 2025 г.

Студент Карпушин Р.А

Содержание

Реферат

Отчёт содержит, 25 страниц, 18 рисунков, 2 таблицы

МНОЖЕСТВА, ОПЕРАЦИИ НАД МНОЖЕСТВАМИ, ОБЪЕДИНЕНИЕ МНОЖЕСТВ

Цель исследования - разработка программы на языке С, реализующей операцию нахождения объединения двух и более множеств с возможностью создания множеств различными способами и сохранением результатов в файл.

В работе рассмотрены теоретические основы операций над множествами. Разработана программа с консольным интерфейсом, позволяющая создавать множества вручную или случайным образом, находить их объединения, сохранять данные в файл. Реализованный алгоритм корректно находит объединения множеств для различных наборов данных.

Введение

Операция объединения множеств - это одна из фундаментальных операций в теории множеств и дискретной математике. Объединение двух или более множеств представляет собой множество, содержащее все элементы, принадлежащие хотя бы одному из исходных множеств. Данная операция широко применяется в различных областях компьютерных наук, таких как базы данных, анализ данных, теория графов и обработка информации.

Основная идея алгоритма объединения множеств без предварительной сортировки заключается в последовательной проверке всех элементов исходных множеств и добавлении в результирующее множество только тех элементов, которые еще не были включены.

Для написания программы я использовал среду разработки Microsoft Visual Studio 2022. Это современная и удобная программа для создания программ 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 объединения двух и более множеств
3. Обеспечивает сохранение данных через файлы
4. Предлагает интуитивно понятный пользовательский интерфейс

1. Постановка задачи

Требуется разработать программу на языке C, реализующую операцию нахождения объединения двух и более множеств целых чисел. Программа должна предоставлять пользователю удобный интерфейс для работы с множествами и поддерживать различные способы взаимодействия с данными.

Множества могут создаваться двумя способами. Во-первых, программа должна генерировать случайные множества с заданными параметрами: пользователь указывает диапазон значений и количество элементов, после чего система создаёт множество из уникальных случайных чисел в указанном диапазоне. Во-вторых, должна быть реализована возможность ручного ввода элементов множества с клавиатуры, при этом программа должна автоматически устраняет дубликаты, сохраняя только уникальные значения.

Программа должна выполнять несколько основных функций. Ключевой задачей является корректная реализация операции объединения множеств с использованием эффективных алгоритмов проверки уникальности элементов. Управление осуществляется через текстовое меню с понятной навигацией.

Итоговый набор возможностей программы включает: создание множеств (случайным образом и вручную), отображение множеств, операцию объединения с возможностью выбора нескольких исходных множеств и сохранение результатов в файл. Пользователь взаимодействует с программой через консольный интерфейс, получая результаты операций на экране.

2.Теоретическая часть

Множество - это математическая структура, представляющая собой неупорядоченную совокупность различных объектов, называемых элементами множества. В контексте данной работы рассматриваются множества целых чисел, где каждый элемент является целым числом, и в пределах одного множества все элементы уникальны

Объединение множеств - это одна из фундаментальных операций в теории множеств, результатом которой является множество, состоящее из элементов всех исходных множеств.

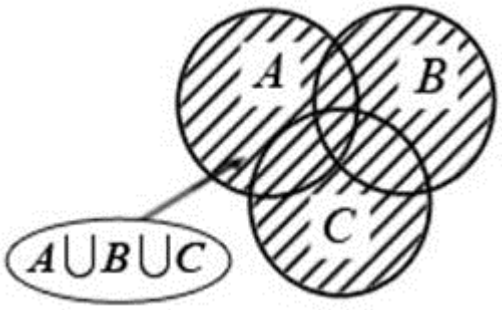

Рисунок 1 – Пример объединения множеств.

Структура данных Set, используемая в программе, представляет собой реализацию математического множества в компьютерной программе. Она состоит из трех основных компонентов:

- Динамического массива elements для хранения элементов множества

- Целого числа size, указывающего текущее количество элементов в множестве

- Целого числа capacity, определяющего текущую емкость выделенной памяти

Алгоритм объединения множеств в данной реализации представляет собой итеративный процесс, который последовательно добавляет элементы из выбранных множеств в результирующее множество, обеспечивая при этом уникальность элементов.

Шаги алгоритма в моей реализации:

1. Инициализация:

Создаем пустое результирующее множество result. Инициализируем индекс для перебора выбранных множеств.

2. Основной цикл по выбранным множествам:

Для каждого выбранного множества currentSet.

3. Вложенный цикл по элементам текущего множества:

Для каждого элемента element в currentSet:

Проверяем, содержится ли element в result, если не содержится - добавляем элемент в result.

4. Завершение:

Возращаем результирующее множество result.

3. Описание программы

3.1 Структура программы

Основной модуль **kurs.c** управляет работой всей программы и предоставляет пользовательский интерфейс в виде консольного меню. Через это меню осуществляется управление всеми функциями.

Func.h - заголовочный файл в котором, в котором объявляются вспомогательные функции.

Func.c – модуль, в котором реализованы вспомогательные функции.

Ключевые функции программы:

createSet() - создание нового пустого множества;

addToSet() - добавление элемента в множество с гарантией уникальности;

freeSet() – освобождение памяти, выделенной под множества;

isInSet() - проверка принадлежности элемента множеству;

unionSelectedSets() - объединение выбранных множеств;

printSet() - вывод множества на экран в математической;

saveToFile() - сохранение всех множеств и результата объединения в текстовый файл.

Такая организация кода обеспечивает чёткое разделение функциональности, что делает программу понятной, легко поддерживаемой и расширяемой.

3.2 Описание алгоритма

ВХОД:

sets[] - массив множеств типа Set;

indices[] - номера множеств, которые нужно объединить;

count - сколько номеров записано в indices[].

ВЫХОД:

result - результирующее множество.

ПЕРЕМЕННЫЕ:

result - результирующее множество;

i - номер выбранного множества (позиция в indices[]);

j - номер элемента внутри текущего множества;

idx - индекс множества из массива sets[];

element - текущий элемент множества.

АЛГОРИТМ

1. ИНИЦИАЛИЗАЦИЯ

1.1 Создать пустое множество result с помощью createSet().

2. ОБХОД ВЫБРАННЫХ МНОЖЕСТВ

2.1 Повторять для каждого выбранного множества:

ПОВТОРЯТЬ i от 0 до count-1:

2.1.1 idx = indices[i]

Берём номер множества, которое надо добавить в объединение.

2.1.2 Обойти все элементы множества sets[idx]:

ПОВТОРЯТЬ j от 0 до sets[idx].size-1:

2.1.2.1 element = sets[idx].elements[j];

Текущий элемент выбранного множества

2.1.2.2 ЕСЛИ isInSet(result, element) == 0, ТО addToSet(&result, element);

ИНАЧЕ (если isInSet == 1), пропустить элемент.

3. ВОЗВРАТ РЕЗУЛЬТАТА

3.1 Вернуть result.

КОНЕЦ АЛГОРИТМА

3.3 Интерфейс программы

После запуска программы пользователь видит следующее главное меню (рисунок 2):

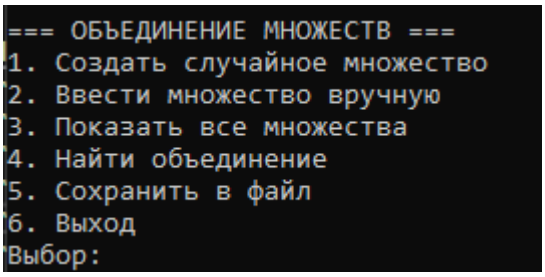

Рисунок 2 – Меню.

Каждый пункт выполняет определённую функцию и возвращает пользователя обратно в меню после завершения операции.

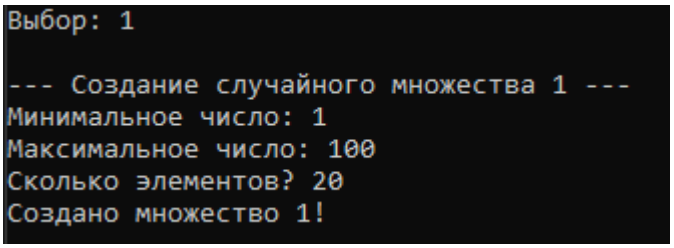

Рисунок 3 – Создание случайного множества.

Программа запрашивает три параметра и создаёт множество с указанными характеристиками (Рисунок 3).

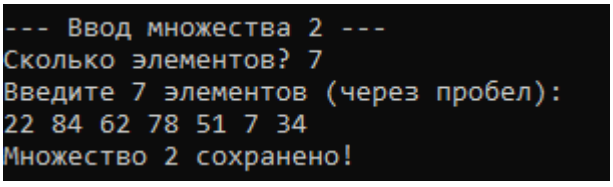

Рисунок 4 – Ручной ввод множества:

Программа запрашивает элементы множества и их количество (Рисунок 4).

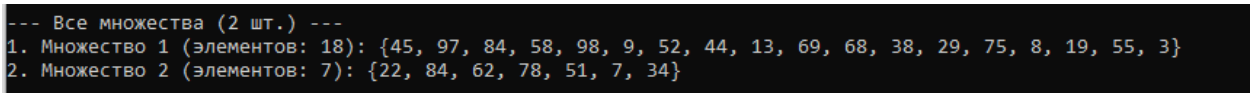

Рисунок 5 – Отображение всех множеств.

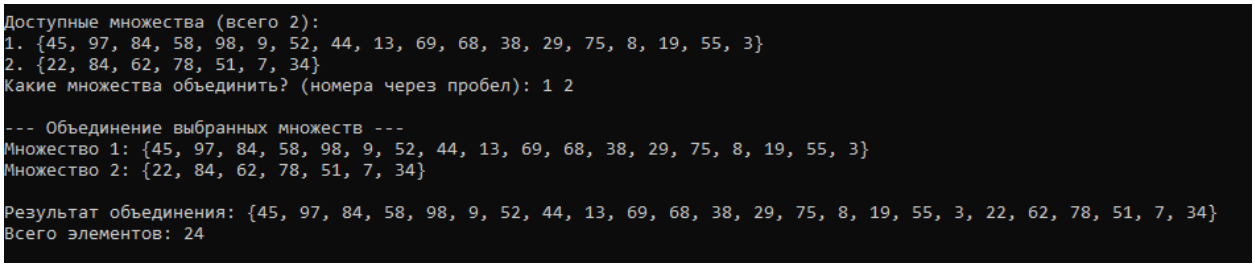

Рисунок 6 – Вывод результата работы алгоритма.

Программа выводит доступные множества и их элементы, запрашивает номера множеств для объединение, после чего выполняет над ними операцию и выводит результат (Рисунок 6).

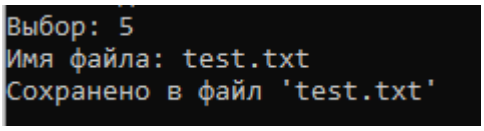

Рисунок 7 – Сохранение в файл.

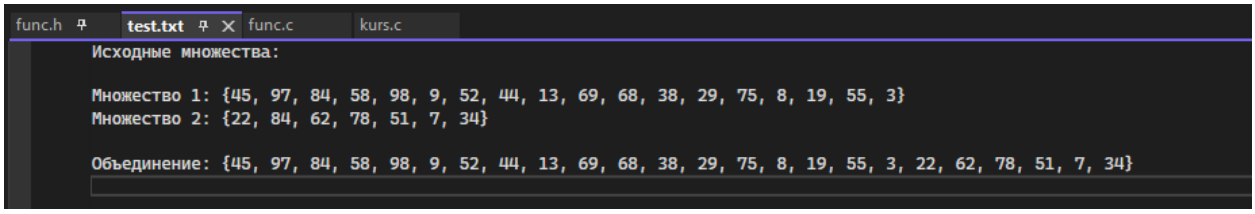

Рисунок 8 – результат в файле.

Для сохранения результатов объединения программа запрашивает имя файла, после чего сохраняет туда исходные множества и результирующее множество (Рисунки 7, 8).

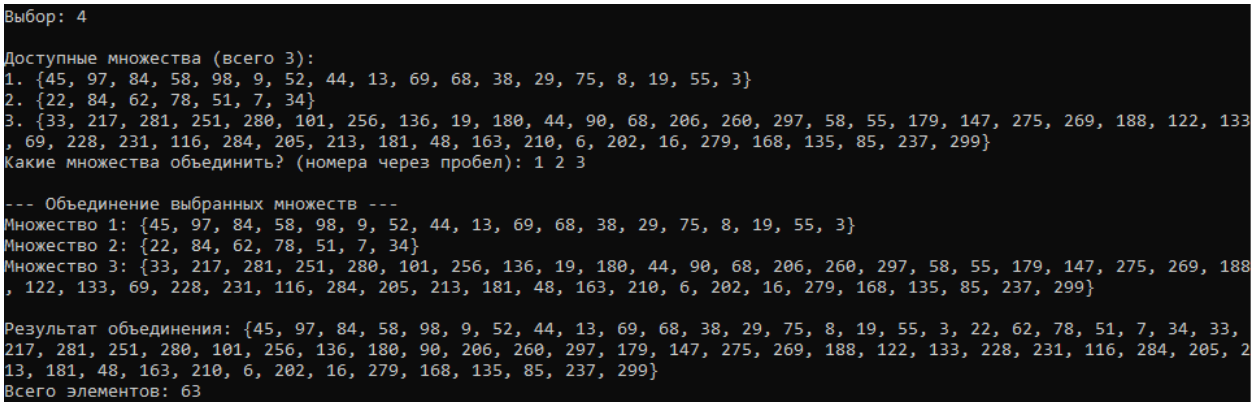

Рисунок 9 – Результат объединения 3-х множеств.

При выборе пункта 6 программа завершается и удаляет множества и результаты объединений.

4.Тестирование

Таблица 1 - план тестирования.

№	Тест	Входные данные	Ожидаемый результат
1	Запуск программы	-	Отображение заголовка и главного меню
2	Генерация случайного множества	Множество из 30 элементов в диапазоне [1;200]	Успешное создание множества с заданными характеристиками
3	Ручной ввод множества	Множество из 10 элементов	Успешное создание множества
4	Вывод всех множеств на экран	Созданные раннее множества	Корректный вывод множеств на экран
5	Нахождение объединения созданных множеств	Предварительно созданные множества	Вывод результирующего множества на экран
6	Сохранение результата в файл	Исходные множества и результат их объединения	Успешное сохранение данных в файл
7	Выход из программы	-	Успешно завершение программы

Полный цикл работы программы представлен на рисунках 10-17.

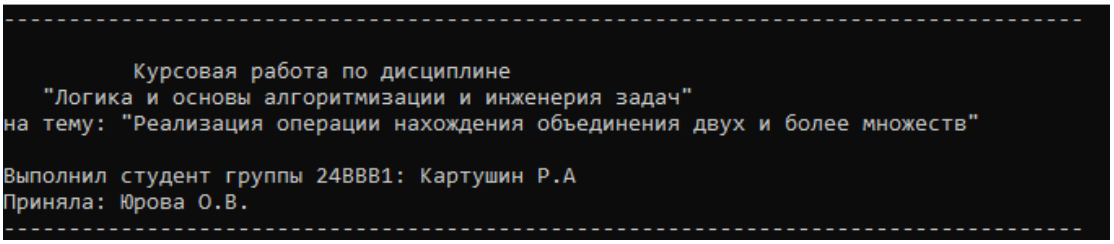

Рисунок 10 – Запуск программы.

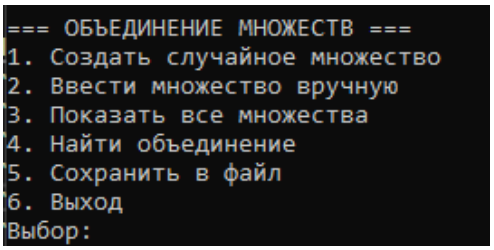

Рисунок 11 – Меню программы.

Результат: Меню отображается корректно.

Выбор пункта 1.

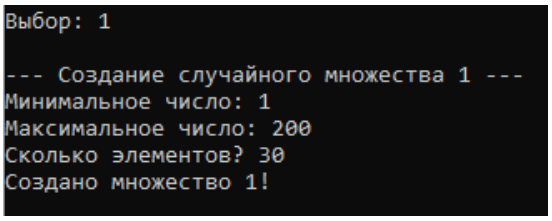

Рисунок 12 – Генерация случайного множества.

Результат: Множество успешно создано.

Выбор пункта 2.

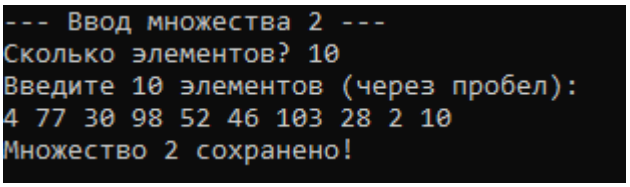

Рисунок 13 – Ручной ввод множества.

Результат: Множество успешно создано.

Выбор пункта 3.

```
Выбор: 3

--- Все множества (2 шт.) ---
1. Множество 1 (элементов: 29): {65, 110, 123, 112, 25, 78, 200, 100, 49, 77, 167, 69, 131, 74, 88, 95, 90, 180, 51, 21,
 178, 169, 66, 31, 148, 173, 73, 139, 113}
2. Множество 2 (элементов: 10): {4, 77, 30, 98, 52, 46, 103, 28, 2, 10}
```

Рисунок 14 – Вывод всех множеств.

Результат: Множества корректно выведены на экран.

Выбор пункта 4.

```
Выбор: 4

Доступные множества (всего 2):
1. {65, 110, 123, 112, 25, 78, 200, 100, 49, 77, 167, 69, 131, 74, 88, 95, 90, 180, 51, 21, 178, 169, 66, 31, 148, 173,
73, 139, 113}
2. {4, 77, 30, 98, 52, 46, 103, 28, 2, 10}
Какие множества объединить? (номера через пробел): 1 2

--- Объединение выбранных множеств ---
Множество 1: {65, 110, 123, 112, 25, 78, 200, 100, 49, 77, 167, 69, 131, 74, 88, 95, 90, 180, 51, 21, 178, 169, 66, 31,
148, 173, 73, 139, 113}
Множество 2: {4, 77, 30, 98, 52, 46, 103, 28, 2, 10}

Результат объединения: {65, 110, 123, 112, 25, 78, 200, 100, 49, 77, 167, 69, 131, 74, 88, 95, 90, 180, 51, 21, 178, 169
, 66, 31, 148, 173, 73, 139, 113, 4, 30, 98, 52, 46, 103, 28, 2, 10}
Всего элементов: 38
```

Рисунок 15 – Объединение множеств.

Результат: Успешное объединение множеств.

Выбор пункта 5.

```
Выбор: 5
Имя файла: test.txt
Сохранено в файл 'test.txt'
```

Рисунок 16 – Сохранение данных в файл.

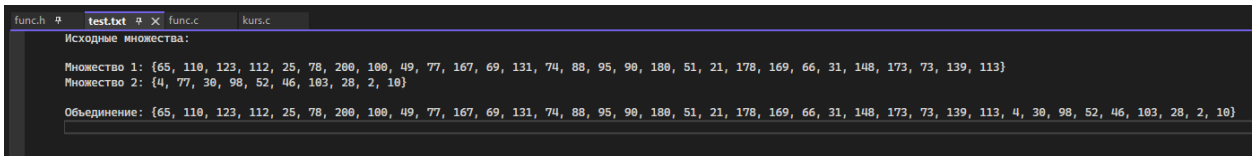

Рисунок 17 – Данные в файле.

Результат: Данные сохранены в файл test.txt.

Выбор пункта 6.

Результат: Успешное завершение программы.

Таблица 2 – результаты поведения программы при тестировании.

№	Тест	Полученный результат
1	Запуск программы	Верно
2	Генерация случайного множества	Верно
3	Ручной ввод множества	Верно
4	Вывод всех множеств на экран	Верно
5	Нахождение объединения созданных множеств	Верно
6	Сохранение результата в файл	Верно
7	Выход из программы	Верно

5. Ручной расчёт

Исходные множества:

1 - {23, 20, 25, 31, 79, 12, 66, 86, 73, 51, 56, 100, 57, 27, 87, 89};

2 - {55, 1, 59, 89, 21, 11, 80, 61, 62, 28, 72, 97, 13, 3, 66, 63, 99, 56, 68, 16, 45, 37};

3 - {81, 84, 100, 87, 34, 57, 12, 11, 82}.

Шаги алгоритма:

Начиная с множества 1, записываем в новоё множество элементы, которые есть хотя бы в одном из исходных.

Получаем результат: {23, 20, 25, 31, 79, 12, 66, 86, 73, 51, 56, 100, 57, 27, 87, 89, 55, 1, 59, 21, 11, 80, 61, 62, 28, 72, 97, 13, 3, 63, 99, 68, 16, 45, 37, 81, 84, 34, 82}.

Проверяем с помощью программы (Рисунок 18):

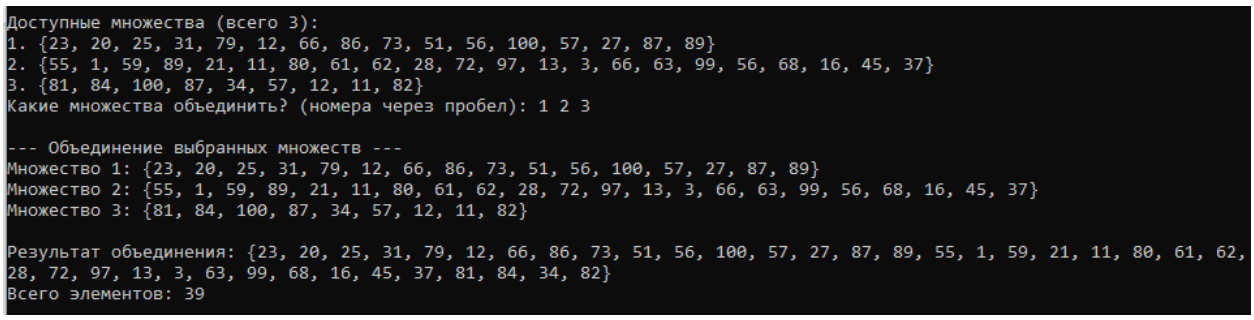

Рисунок 18 – Результат программы.

Таким образом, ручной расчёт подтвердил корректность работы программы. Совпадение результатов демонстрирует, что реализованный алгоритм нахождения объединения множеств функционирует правильно.

Заключение

В ходе выполнения курсовой работы мною была разработана программа на языке C, реализующая алгоритм для нахождения объединения двух и более множеств.

Программа предоставляет пользователю удобный консольный интерфейс, позволяющий генерировать случайные множества, вводить данные вручную, выполнять алгоритм и сохранять результаты. Все поставленные цели достигнуты: изучены теоретические основы нахождения объединения двух и более множеств, проведено тестирование, подтвердившее корректность работы программы.

В процессе работы были приобретены и углублены практические навыки программирования на C, включая работу со стандартной библиотекой шаблонов, структурами данных и файловыми операциями.

Программа имеет небольшой, но достаточный функционал возможностей. В качестве возможных улучшений можно отметить добавление графического интерфейса.

Список использованных источников

1. Френкель А.А., Бар-Хиллел И. "Основания теории множеств" (1966)

2. Кормен, Т. Х., Лейзерсон, Ч. Е., Ривест, Р. Л., Стейн, К. *Введение в алгоритмы*. — М.: Издательство "Дом интеллектуальной собственности", 2011.

3. Куратовский К., Мостовский А. "Теория множеств" (1970)

4. Шамшурин, А. И. *Теория множеств и математическая логика*. — М.: Изд-во МГУ, 2010.

5. Хакимов, С. Г. *Алгоритмы работы с множествами в языке программирования С*. — Программирование и алгоритмы, 2009.

6. Керниган, Б., Ритчи, Д. *Язык программирования С*. — М.: Изд-во "Диалектика", 2008.

Приложение А (Листинг программы)

Kurs.c

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

#define MAX_SETS 10

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

void displayMenu() {
    printf("\n=== ОБЪЕДИНЕНИЕ МНОЖЕСТВ ===\n");
    printf("1. Создать случайное множество\n");
    printf("2. Ввести множество вручную\n");
    printf("3. Показать все множества\n");
    printf("4. Найти объединение \n");
    printf("5. Сохранить в файл\n");
    printf("6. Выход\n");
    printf("Выбор: ");
}

int main() {
    setlocale(LC_ALL, "Russian");
    printf("-------------------------------------------------------------------------------\n\n");
    printf("          Курсовая работа по дисциплине\n");
    printf("   \"Логика и основы алгоритмизации и инженерия задач\"\n");
    printf("на тему: \"Реализация операции нахождения объединения двух и более множеств\"\n\n");
    printf("Выполнил студент группы 24ВВВ1: Картушин Р.А\n");
    printf("Приняла: Юрова О.В.\n");
    printf("-------------------------------------------------------------------------------\n\n");

    Set sets[MAX_SETS];
    Set result;
    int count = 0;
    int choice;

    for (int i = 0; i < MAX_SETS; i++) {
        sets[i] = createSet();
    }
    result = createSet();

    do {
        displayMenu();
        if (scanf("%d", &choice) != 1) {
            printf("Неверный ввод!\n");
            clearInput();
            continue;
        }
        clearInput();

        switch (choice) {
        case 1: {
            if (count >= MAX_SETS) {
                printf("Максимум %d множеств!\n", MAX_SETS);
                break;
            }

            printf("\n--- Создание случайного множества %d ---\n", count + 1);
```

```
            int min, max, elements;

            printf("Минимальное число: ");
            scanf("%d", &min);
            clearInput();

            printf("Максимальное число: ");
            scanf("%d", &max);
            clearInput();

            printf("Сколько элементов? ");
            scanf("%d", &elements);
            clearInput();

            freeSet(&sets[count]);
            sets[count] = createSet();

            srand((unsigned int)time(NULL));
            for (int i = 0; i < elements; i++) {
                int num = min + rand() % (max - min + 1);
                addToSet(&sets[count], num);
            }

            printf("Создано множество %d!\n", count + 1);
            count++;
            break;
        }

        case 2: {
            if (count >= MAX_SETS) {
                printf("Максимум %d множеств!\n", MAX_SETS);
                break;
            }

            printf("\n--- Ввод множества %d ---\n", count + 1);

            freeSet(&sets[count]);
            sets[count] = createSet();

            printf("Сколько элементов? ");
            int n;
            scanf("%d", &n);
            clearInput();

            printf("Введите %d элементов (через пробел):\n", n);
            for (int i = 0; i < n; i++) {
                int num;
                if (scanf("%d", &num) == 1) {
                    addToSet(&sets[count], num);
                }
            }
            clearInput();

            printf("Множество %d сохранено!\n", count + 1);
            count++;
            break;
        }

        case 3: {
            if (count == 0) {
                printf("Нет созданных множеств!\n");
                break;
            }

            printf("\n--- Все множества (%d шт.) ---\n", count);
```

```
    for (int i = 0; i < count; i++) {
        printf("%d. Множество %d (элементов: %d): ",
            i + 1, i + 1, sets[i].size);
        printSet(sets[i]);
    }
    break;
}

case 4: {
    if (count < 2) {
        printf("Нужно минимум 2 множества! Сейчас %d.\n", count);
        break;
    }

    printf("\nДоступные множества (всего %d):\n", count);
    for (int i = 0; i < count; i++) {
        printf("%d. ", i + 1);
        printSet(sets[i]);
    }

    printf("Какие множества объединить? (номера через пробел): ");

    int indices[MAX_SETS];
    int selected = 0;
    char input[100];

    fgets(input, sizeof(input), stdin);
    char* token = strtok(input, " \t\n");

    while (token != NULL && selected < MAX_SETS) {
        int idx = atoi(token) - 1;
        if (idx >= 0 && idx < count) {
            indices[selected++] = idx;
        }
        token = strtok(NULL, " \t\n");
    }

    if (selected < 2) {
        printf("Нужно выбрать минимум 2 множества!\n");
        break;
    }

    printf("\n--- Объединение выбранных множеств ---\n");
    for (int i = 0; i < selected; i++) {
        int idx = indices[i];
        printf("Множество %d: ", idx + 1);
        printSet(sets[idx]);
    }

    freeSet(&result);
    result = unionSelectedSets(sets, indices, selected);

    printf("\nРезультат объединения: ");
    printSet(result);
    printf("Всего элементов: %d\n", result.size);
    break;
}

case 5: {
    if (count == 0) {
        printf("Нет множеств для сохранения!\n");
        break;
    }

    if (result.size == 0) {
        printf("Сначала найдите объединение!\n");
```

```
                break;
            }

            char filename[100];
            printf("Имя файла: ");
            scanf("%99s", filename);
            clearInput();

            saveToFile(sets, count, result, filename);
            break;
        }

        case 6: {
            printf("Выход...\n");
            break;
        }

        default: {
            printf("Неверный выбор! (1-6)\n");
            break;
        }
        }
    } while (choice != 6);


    for (int i = 0; i < MAX_SETS; i++) {
        freeSet(&sets[i]);
    }
    freeSet(&result);

    return 0;
}
```

Приложение А.1

FUNC.H

```
#ifndef FUNC_H
#define FUNC_H

typedef struct {
    int* elements;
    int size;
    int capacity;
} Set;

Set createSet();
void addToSet(Set* set, int element);
int isInSet(Set set, int element);
Set unionSelectedSets(Set sets[], int indices[], int count);
void printSet(Set set);
void saveToFile(Set sets[], int count, Set result, const char* filename);
void freeSet(Set* set);

#endif
```

Приложение А.2

FUNC.C

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "func.h"

Set createSet() {
    Set set;
    set.size = 0;
    set.capacity = 10;
    set.elements = malloc(set.capacity * sizeof(int));
    return set;
}

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

    if (set->size >= set->capacity) {
        set->capacity *= 2;
        set->elements = realloc(set->elements, set->capacity * sizeof(int));
    }

    set->elements[set->size] = element;
    set->size++;
}

int isInSet(Set set, int element) {
    for (int i = 0; i < set.size; i++) {
        if (set.elements[i] == element) return 1;
    }
    return 0;
}

Set unionSelectedSets(Set sets[], int indices[], int count) {
    Set result = createSet();

    for (int i = 0; i < count; i++) {
        int idx = indices[i];
        for (int j = 0; j < sets[idx].size; j++) {
            int element = sets[idx].elements[j];
            if (!isInSet(result, element)) {
                addToSet(&result, element);
            }
        }
    }

    return result;
}

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 saveToFile(Set sets[], int count, Set result, const char* filename) {
    FILE* file = fopen(filename, "w");
    if (!file) {
        printf("Ошибка создания файла!\n");
        return;
    }

    fprintf(file, "Исходные множества:\n\n");
    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]);
            if (j < sets[i].size - 1) fprintf(file, ", ");
        }
        fprintf(file, "}\n");
    }

    fprintf(file, "\nОбъединение: {");
    for (int i = 0; i < result.size; i++) {
        fprintf(file, "%d", result.elements[i]);
        if (i < result.size - 1) fprintf(file, ", ");
    }
    fprintf(file, "}\n");

    fclose(file);
    printf("Сохранено в файл '%s'\n", filename);
}

void freeSet(Set* set) {
    if (set->elements) {
        free(set->elements);
    }
    set->size = 0;
    set->capacity = 0;
}
```