Министерство образования Российской Федерации

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 симметрической разности двух и более множеств”

Выполнил:

Студент группы 23ВВВ1

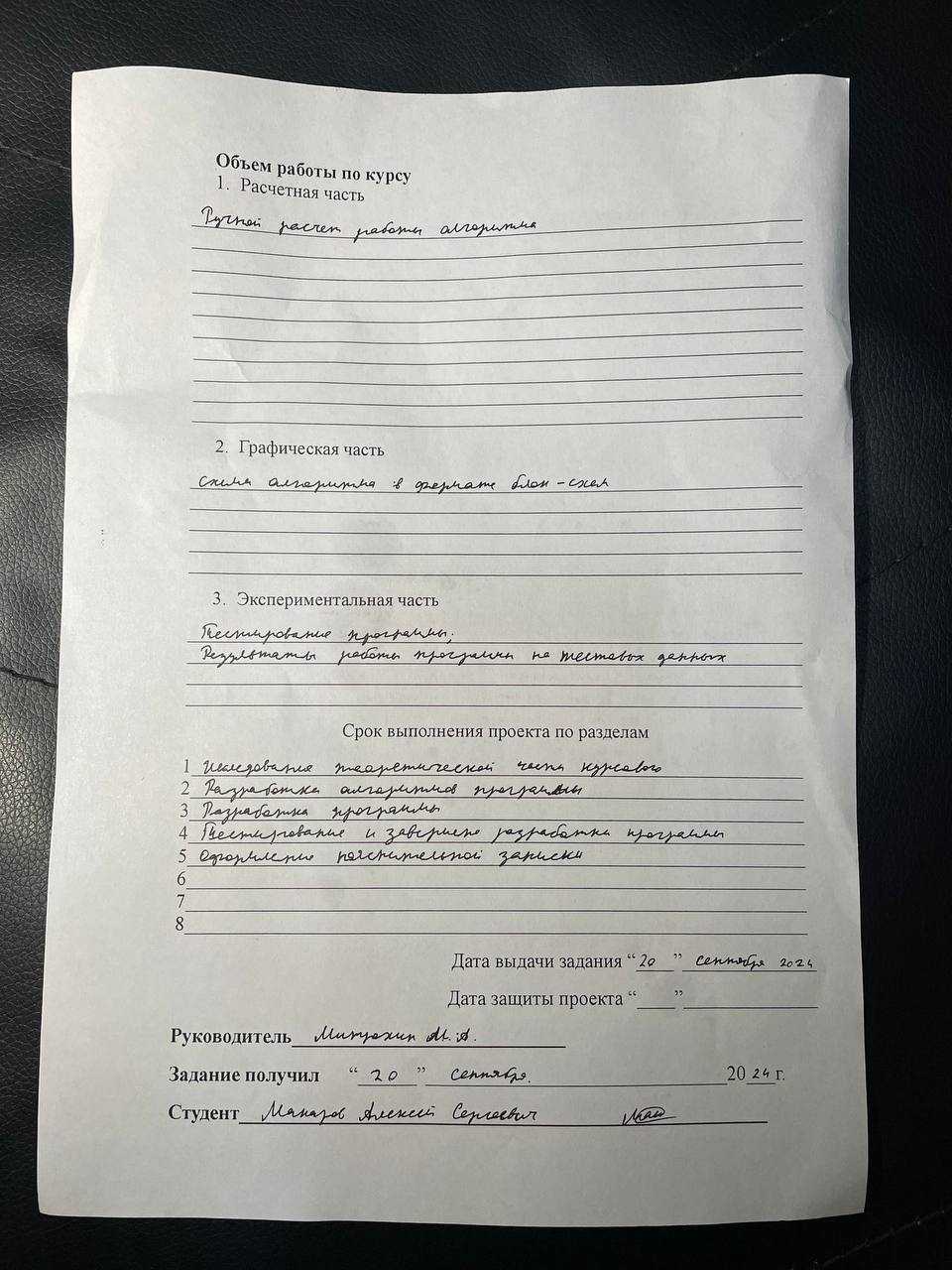
Макаров А.С.

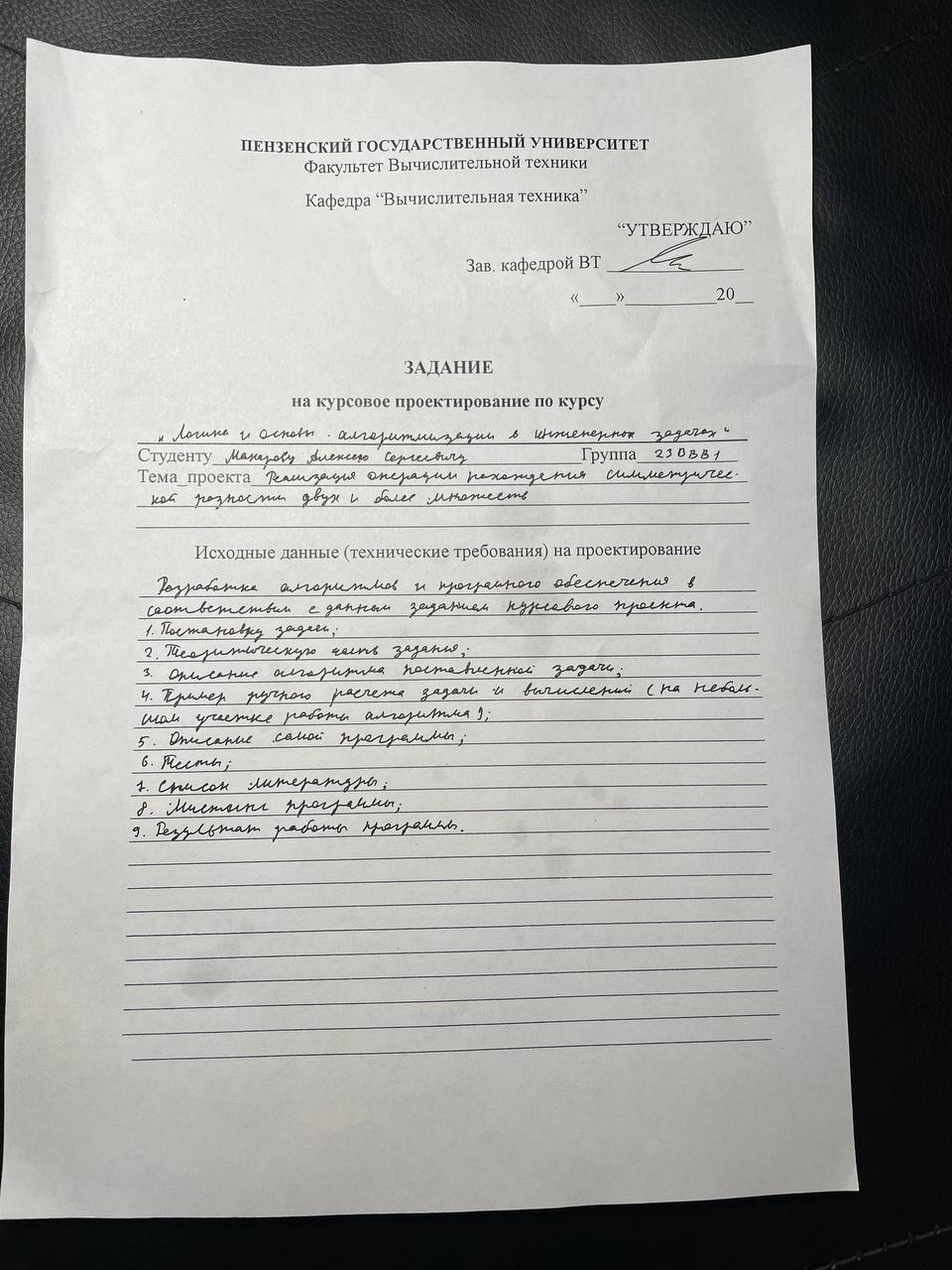
Приняли

К.т.н, доцент Юрова О.В

Д.т.н, профессор Митрохин М.А

Пенза 2024





**Содержание**

[Реферат 5](#_Toc186038563)

[Введение 6](#_Toc186038564)

[1. Постановка задачи 7](#_Toc186038565)

[2. Теоретическая часть задания 8](#_Toc186038566)

[3. Описание алгоритма 10](#_Toc186038567)

[4. Описание программы 13](#_Toc186038568)

[5. Тестирование 22](#_Toc186038569)

[6.Ручной расчет 28](#_Toc186038570)

[Заключение 29](#_Toc186038571)

[Список литературы 30](#_Toc186038572)

[Приложение А. Листинги программы 31](#_Toc186038573)

# Реферат

Цель исследования - разработка программы, реализующую операцию нахождения симметрической разности двух и более множеств.

В работе рассматривается способ нахождения симметрической разности двух и более множеств с использованием языка программирования C.

Симметрическая разность — это операция в теории множеств, которая позволяет определить элементы, принадлежащие только одному из множеств, но не обоим одновременно.

Эта операция имеет важное значение в различных областях, таких как информатика, статистика и логика.

Теория множеств — это раздел математики, изучающий свойства и отношения между множествами.

Множество представляет собой коллекцию уникальных объектов, называемых элементами.

Операции над множествами, такие как объединение, пересечение и разность, позволяют исследовать их взаимосвязи.

Симметрическая разность, в частности, используется для выявления уникальных значений в наборах данных и анализа их взаимосвязей.

# Введение

Симметрическая разность — это операция в теории множеств, позволяющая определить элементы, которые принадлежат только одному из двух или более множеств, но не принадлежат одновременно всем множествам. Эта операция имеет важное значение в различных областях, таких как информатика, статистика и логика, поскольку она помогает выявлять уникальные значения и анализировать взаимосвязи между наборами данных.

В данной работе реализован алгоритм нахождения симметрической разности для двух и более множеств с использованием языка программирования C. Программа позволяет пользователю вводить множества, проверять уникальность элементов и вычислять симметрическую разность, что делает ее полезным инструментом для анализа данных.

Алгоритм работает за O(n^2) времени в зависимости от количества множеств и их размеров, и использует O(n) памяти для хранения результатов. Реализация включает функции для ввода множеств, вычисления симметрической разности и отображения результатов, что обеспечивает удобный интерфейс для пользователя.

Данная работа демонстрирует практическое применение теории множеств и алгоритмических подходов для решения задач, связанных с обработкой и анализом информации. Реализация операции нахождения симметрической разности позволяет упростить анализ данных и является актуальной в условиях, когда объемы информации постоянно растут.

# 1. Постановка задачи

Требуется разработать программу, которая реализует операцию нахождения симметрической разности двух и более множеств.

Программа должна позволять пользователю вводить множества вручную, обеспечивая при этом проверку уникальности элементов.

После ввода множества, пользователь должен иметь возможность вычислить симметрическую разность между всеми введенными множествами.

# 2. Теоретическая часть задания

**Симметрическая разность** двух [множеств](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE) — теоретико множественная операция, результатом которой является новое множество, включающее все элементы исходных множеств, не принадлежащие одновременно обоим исходным множествам.

Другими словами, если есть два множества AA и BB, их симметрическая разность есть объединение элементов AA, не входящих в BB, с элементами BB, не входящими в AA.

На письме для обозначения симметрической разности множеств AA и BB используется обозначение A△B, реже используется обозначение A−˙BA - B или A+BA + B.

Основная идея алгоритма нахождения симметрической разности заключается в том, чтобы пройти по всем элементам введенных множеств и определить, какие из них являются уникальными для каждого множества.

Если элемент присутствует только в одном из множеств, он добавляется в результат.

Основные шаги алгоритма включают:

Инициализация: Создание массива для хранения результата и массива для отслеживания добавленных элементов. В программе выделяется память для хранения до 1000 элементов результат и используется массив для отметки добавленных элементов.

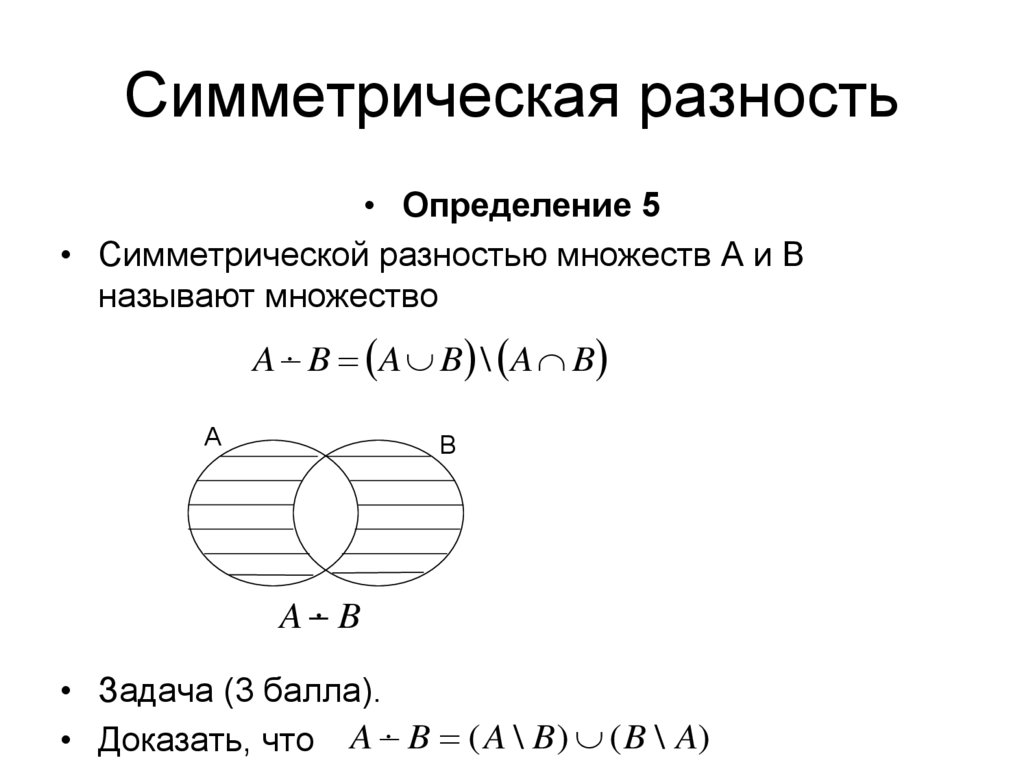
Цикл по множествам: Для каждого множества программа проходит по его элементам и проверяет, присутствует ли элемент в других множествах. Если элемент не найдет в других множествах и еще не был добавлен в результат, он добавляется в массив результата.

Проверка уникальности: Программа использует вложенные циклы для проверки наличия элемента в других множества. Если элемент найдет в другом множестве, он не добавляется в результат.

Вывод результата: После обработки всех множеств программа выводит элементы, которые составляют симметрическую разносить на экран.

Программа использует динамическое выделение памяти для хранения множеств и результатов. Важно правильно управлять памятью, освобождая выделенные ресурсы после завершения работы программы, чтобы избежать утечек программы.

Симметрическая разность является важной операцией в теории множеств, и ее реализация в виде программы позволяет эффективно решать задачи, связанные с анализом данных.



**Рисунок 1- Диаграмма операции**

# 3. Описание алгоритма

Для написания данной программы был использован язык программирование С.

Проект был создан в виде консольного приложения в программе VS 2019.

НачалоФункция:

Входные данные: sets-Массив указателей на массивы целых чисел, sizes-Массив целых чисел, numSets-Целое число.

Инициализировать массив result размером 1000 для хранения результата

Инициализировать переменную resultSize равной 0

Инициализировать массив added размером 1000 для отслеживания добавленных элементов

Для i от 0 до numSets - 1:

Для j от 0 до sizes[i] - 1:

Установить found равным 0

Для k от 0 до numSets - 1:

Если k не равно i:

Для l от 0 до sizes[k] - 1:

Если sets[i][j] равно sets[k][l]:

Установить found равным 1

Прервать внутренний цикл

Конец Для

Конец Если

Если found равно 1:

Прервать внешний цикл

Конец Для

Если found равно 0 и added[sets[i][j]] равно 0:

Добавить sets[i][j] в result[resultSize]

Увеличить resultSize на 1

Установить added[sets[i][j]] равным 1

Вывести "Симметрическая разность: "

Для i от 0 до resultSize - 1:

Вывести result[i]

Конец Для

Освободить память для result

Освободить память для added

КонецФункции

НачалоФункция input\_set(set, size):

Входные данные: set- Указатель на массив целых чисел, seize- Указатель на целое число.

Вывести "Введите количество элементов множества: "

Считать size

Выделить память для set размером size

Инициализировать переменную uniqueCount равной 0

Вывести "Введите элементы множества:"

Пока uniqueCount меньше size:

Считать element

Инициализировать exists равным 0

Для i от 0 до uniqueCount - 1:

Если set[i] равно element:

Установить exists равным 1

Прервать цикл

Конец Для

Если exists равно 0:

Установить set[uniqueCount] равным element

Увеличить uniqueCount на 1

Иначе:

Вывести "Элемент element уже существует в множестве. Пожалуйста, введите другой элемент."

Установить size равным uniqueCount

Конец Функции

Работа программы начинается с отображения информации о проекте и запроса выбора действия в меню. Пользователь может выбрать один из трех пунктов: ввод множества, нахождение симметрической разности или выход из программы.

Если пользователь выберет пункт 1, программа предложит ввести элементы множества. Сначала будет запрошено количество элементов, после чего пользователь сможет вводить сами элементы. Программа проверяет уникальность вводимых значений, и если элемент уже существует в множестве, пользователю будет предложено ввести другойэлемент. После успешного ввода программа обновляет массив, хранящий элементы множества, и его размер.

Если пользователь выберет пункт 2, программа проверит, введено ли хотя бы два множества. Если да, то она выполнит операцию нахождения симметрической разности. Для этого программа пройдет по всем элементам введенных множеств и определит, какие из них являются уникальными для каждого множества. Результат будет выведен на экран в виде списка уникальных элементов, которые присутствуют только в одном из множеств, но не в других.

Пункт 3 завершает работу программы. После завершения всех операций программа освобождает выделенную память для массивов, хранящих множества и их размеры, чтобы избежать утечек памяти.

Таким образом, программа предоставляет пользователю удобный интерфейс для работы с множествами, позволяя вводить данные, вычислять симметрическую разность и управлять памятью. Программа будет готова к новому запуску до тех пор, пока пользователь не выберет пункт "Выход" в меню.

# 4. Описание программы

Программа реализует операцию нахождения симметрической разности двух и более множеств.

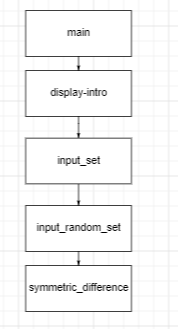
Пользователь может вводить множества, при этом программа проверяет уникальность элементов и позволяет вводить только неповторяющиеся значения.

После ввода множества пользователь может выбрать опцию для вычисления симметрической разности, и программа выведет элементы, которые присутствуют только в одном из множеств, но не в других.

Программа имеет удобный текстовый интерфейс с меню, позволяющим легко навигировать между функциями. В конце работы программа освобождает выделенную память, предотвращая утечки ресурсов.

Были использованы функции Symmetric\_difference(), input\_set(),input\_set(),display\_intro(),main().

Общее представление программы на диаграмме модулей:



**Рисунок 2 – Диаграмма модулей**

**Функция symmetric\_difference**

void symmetric\_difference(int\*\* sets, int\* sizes, int numSets) {

int\* result = (int\*)malloc(1000 \* sizeof(int));

int resultSize = 0;

int\* added = (int\*)calloc(1000, sizeof(int));

for (int i = 0; i < numSets; i++) {

for (int j = 0; j < sizes[i]; j++) {

int found = 0;

for (int k = 0; k < numSets; k++) {

if (k != i) {

for (int l = 0; l < sizes[k]; l++) {

if (sets[i][j] == sets[k][l]) {

found = 1;

break;

}

}

}

if (found) break;

}

if (!found && !added[sets[i][j]]) {

result[resultSize++] = sets[i][j];

added[sets[i][j]] = 1;

}

}

}

printf("Симметрическая разность: ");

for (int i = 0; i < resultSize; i++) {

printf("%d ", result[i]);

}

printf("\n");

free(result);

free(added);

}

Назначение:

Функция предназначена для вычисления симметрической разности между несколькими множествами, представленными в виде массива указателей на массивы целых чисел.

Логика работы:

1. Инициализация

2. Основной цикл по множествам

3. Проверка уникальности элемента

4. Добавление уникального элемента в результат

5. Вывод результата

6. Освобождение памяти и проверка.

**Функция input\_set**

void input\_set(int\*\* set, int\* size) {

printf("Введите количество элементов множества: ");

scanf("%d", size);

\*set = (int\*)malloc((\*size) \* sizeof(int));

int uniqueCount = 0;

printf("Введите элементы множества:\n");

while (uniqueCount < \*size) {

int element;

scanf("%d", &element);

int exists = 0;

for (int i = 0; i < uniqueCount; i++) {

if ((\*set)[i] == element) {

exists = 1;

break;

}

}

if (!exists) {

(\*set)[uniqueCount++] = element;

}

else {

printf("Элемент %d уже существует в множестве. Пожалуйста, введите другой элемент.\n", element);

}

}

\*size = uniqueCount;

printf("Вы ввели множество: { ");

for (int i = 0; i < \*size; i++) {

printf("%d ", (\*set)[i]);

}

printf("}\n");

}

Назначение:

Функция предназначена для ввода множества целых чисел от пользователя.

Логика работы:

1. Запрос количества элементов

2. Выделение памяти

3. Ввод элементов множества

4. Проверка уникальности

5. Добавление уникального элемента

6. Обновление размера множества

**Функция input\_random\_set**

void input\_random\_set(int\*\* set, int\* size) {

printf("Хотите ли вы задать количество элементов вручную (1) или использовать случайное количество (2)? ");

int choice;

scanf("%d", &choice);

if (choice == 1) {

printf("Введите количество элементов множества: ");

scanf("%d", size);

}

else {

\*size = rand() % 10 + 1;

printf("Случайное количество элементов: %d\n", \*size);

}

\*set = (int\*)malloc((\*size) \* sizeof(int));

int uniqueCount = 0;

while (uniqueCount < \*size) {

int element = rand() % 100;

int exists = 0;

for (int i = 0; i < uniqueCount; i++) {

if ((\*set)[i] == element) {

exists = 1;

break;

}

}

if (!exists) {

(\*set)[uniqueCount++] = element;

}

}

printf("Сгенерированное множество: { ");

for (int i = 0; i < \*size; i++) {

printf("%d ", (\*set)[i]);

}

printf("}\n");

}

Назначение:

Функция input\_random\_set предназначена для создания множества с элементами, которые могут быть заданы пользователем вручную или сгенерированы случайным образом.

Логика работы:

1. Запрос выбора способа задания количества элементов
2. Задание количества элементов
3. Выделение памяти для множества
4. Заполнение множества уникальными случайными значениями
5. Вывод сгенерированного множества

**Функция display\_intro**

printf("===============================================================================================\n");

printf(" Реализация операции нахождения симметрической разности двух и более множеств \n");

printf(" Выполнил: Макаров Алексей Сергеевич \n");

printf(" Учебная группа: 23ВВВ1 \n");

printf("===============================================================================================\n");

printf("\nНажмите Enter для продолжения...\n");

getchar();

Назначение:

Функция предназначена для отображения информации о программе и ее авторе.

Логика работы:

1. Формирование заголовка

2. Вывод информации о программе

3. Завершение заголовка

4. Ожидание ввода от пользователя

**Логика работы main**

setlocale(LC\_ALL, "Rus");

display\_intro();

char y = '0';

int\*\* sets = NULL;

int\* sizes = NULL;

int numSets = 0;

int choice;

do {

system("cls");

printf("\n==============================\n");

printf(" МЕНЮ \n");

printf("==============================\n");

printf("1. Ввести множество\n");

printf("2. Найти симметрическую разность\n");

printf("3. Выход\n");

printf("==============================\n");

printf("Выберите опцию: ");

scanf("%d", &choice);

switch (choice) {

case 1: {

system("cls");

sets = (int\*\*)realloc(sets, (numSets + 1) \* sizeof(int\*));

sizes = (int\*)realloc(sizes, (numSets + 1) \* sizeof(int));

input\_set(&sets[numSets], &sizes[numSets]);

numSets++;

printf("\nНажмите Enter для перехода в меню...\n");

getchar();

getchar();

break;

}

case 2:

if (numSets > 0) {

system("cls");

for (int i = 0; i < numSets; i++) {

printf("В %d множестве находятся: ", i + 1);

for (int j = 0; j < sizes[i]; j++) printf("%d ", sets[i][j]);

printf("\n");

}

if (numSets < 2) {

printf("Пожалуйста, введите хотя бы два множества для нахождения симметрической разности.\n");

}

else {

symmetric\_difference(sets, sizes, numSets);

}

printf("\nНажмите Enter для перехода в меню...\n");

getchar();

getchar();

}

else {

printf("Сначала введите хотя бы одно множество.\n");

printf("\nНажмите Enter для перехода в меню...\n");

getchar();

getchar();

}

break;

case 3:

printf("Выход из программы.\n");

break;

default:

printf("Неверный выбор. Пожалуйста, попробуйте снова.\n");

printf("\nНажмите Enter для перехода в меню...\n");

getchar();

getchar();

}

} while (choice != 3);

for (int i = 0; i < numSets; i++) {

free(sets[i]);

}

free(sets);

free(sizes);

return 0;

}

1. Инициализация и объявление переменных:

В начале функции main происходит объявление необходимых переменных:

Устанавливается локализация для корректного отображения русских символов с помощью setlocale(LC\_ALL, "Rus");.

Инициализируется генератор случайных чисел с помощью srand(time(NULL));, что позволяет получать разные случайные значения при каждом запуске программы.

Массив указателей int\* sets[2], который будет хранить указатели на два множества целых чисел, и инициализируется значением NULL.

Массив целочисленных переменных sizes[2], который будет хранить размеры каждого из двух множеств, и инициализируется значениями 0.

Целочисленная переменная numSets, которая хранит количество введенных множеств и инициализируется значением 0.

Целочисленная переменная choice, которая используется для ввода выбора пользователя в меню программы.

2. Ввод пользователем множества:

Программа запрашивает у пользователя ввод количества элементов для каждого множества.

Вводимые значения проверяются на соответствие рабочему диапазону, чтобы избежать ошибок при выделении памяти.

Введенные значения обрабатываются с помощью функции input\_set, которая позволяет пользователю вводить элементы множества, проверяя их уникальность.

Также предусмотрена возможность создания множества с случайными значениями с помощью функции input\_random\_set.

3. Обработка данных алгоритмом:

После ввода множества пользователь может выбрать опцию для нахождения симметрической разности.

Программа проверяет, были ли введены оба множества, и вызывает функцию symmetric\_difference, которая обрабатывает введенные множества и находит уникальные элементы, присутствующие только в одном из множеств.

Результаты обработки выводятся на экран, показывая симметрическую разность.

4. Завершение работы программы:

После завершения всех операций программа освобождает выделенную память для массивов, чтобы избежать утечек памяти.

Пользователь может завершить работу программы, выбрав соответствующий пункт в меню, после чего программа выводит сообщение о завершении.

Это описание охватывает основные этапы работы программы, аналогично вашему примеру, и подчеркивает логику и структуру кода в функции main.

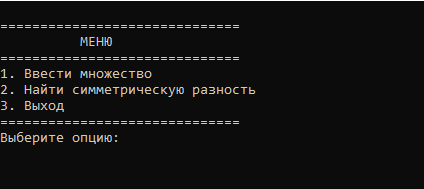
# 5. Тестирование

Среда разработки Microsoft Visual Studio 2022 предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Программа будет протестирована на разном наборе данных, проверки функциональности вывода сообщений вывода данных в консоль, а также на проверку ошибок ввода и вывода ошибок.

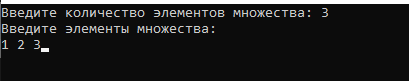
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Описание теста | Состояние программ | Действия | Ожидаемый результат | Полученный результат |
| Запуск программы и работа с меню | Ожидание ввода пользователя. | Ввод пользователем данных с клавиатуры. | Вывод работоспособного меню в консоль | Верно |
| Выбор в меню первого пункта. Создание первого множества | Ожидание пока пользователь введет количество элементов в множестве и введет эти элементы | Пользователь вводит количество элементов в множестве и заполняет его различными элементами | Добавление в список нового множества в котором будет введенное количество элементов и они будут соответствовать введенным пользователем. | Верно |
| Выбор в меню первого пункта. Создание первого множества | Ожидание пока пользователь введет количество элементов в множестве и введет эти элементы | Пользователь вводит количество элементов в множестве и заполняет его различными элементами | Добавление в список второго множества в котором будет введенное количество элементов и они будут соответствовать введенным пользователем. | Верно |
| Выбор в меню второго пункта “Нахождения симметрической разности” | Вывод внесенных ранее множеств с внесенных в них элементов | Просмотри пользователем значений в множествах | Вывод разности множеств. Нахождение верной разности | Верно |
| Проверка на заполнения множества случайными значениями | Программа предлагает выбрать как указать количество элементов(в ручную или случайным образом) | Пользователь выбирает указать количество в ручную. | Заполнение множества случайными числами и их количество будет совпадать заданному количеству в множестве | Верно |
| Проверка на то рассчитана ли программа на работу с большим количеством множеств больше двух | Ожидание пока пользователь введет количество элементов в множестве и введет эти элементы | Пользователь вводит количество элементов в множестве и заполняет его различными элементами | Добавление в список нового множества в котором будет введенное количество элементов и они будут соответствовать введенным пользователем. | Верно |

1. Запуск программы и работа с меню.
   1. Вывод меню с программы:



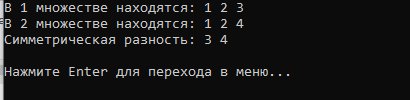
**Рисунок 3 – Вывод меню**

* 1. Выбор пункта “1”в меню программы.



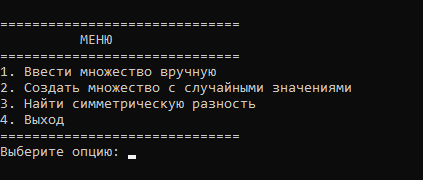
**Рисунок 4 – Выбор пункта “1” в меню**

* 1. Выбор пункта “2” в меню программы.



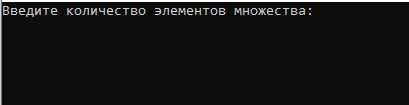
**Рисунок 5 – Выбор пункта “2” в меню**

* 1. Выбор пункта “3” в меню программы.



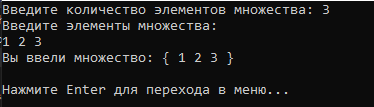
**Рисунок 6 – Выбор пункта “3” в меню**

1. Создание первого множества.
   1. Ожидание ввода количества элементов в множестве.



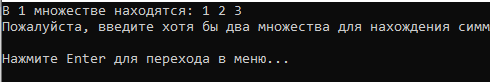
**Рисунок 7 – Ожидание ввода**

* 1. Ввод количества элементов и заполнение множества.



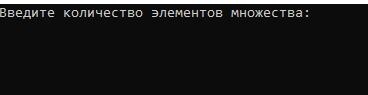
**Рисунок 8 – Заполнение множества**

* 1. Проверка на правильность введенных значений.



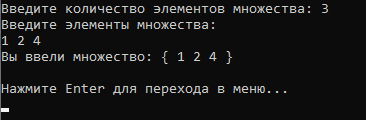
**Рисунок 9 – Проверка введенных элементов**

1. Ввод второго множества.
   1. Ожидание ввода количества элементов во втором множестве.



**Рисунок 10 – Ввод количества элементов**

* 1. Ввод количества элементов и заполнение второго множества.



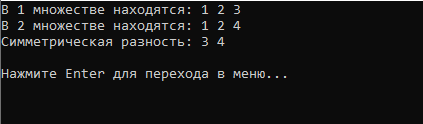
**Рисунок 11 – заполнение второго множества**

* 1. Проверка на добавление и правильность второго множества.



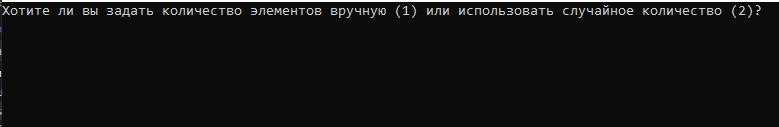
**Рисунок 12 – Проверка на добавление второго множества**

1. Выбор второго пункта и нахождения разности между элементами.



**Рисунок 13 – Нахождение симметрической разности**

1. Проверка на заполнение множества случайными элементами.
   1. Создание множества.



**Рисунок 14 – Выбор варианта**

* 1. Выбор варианта “1”.



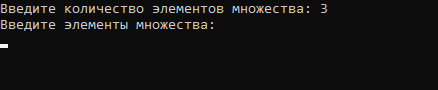
**Рисунок 15 – заполняем множество**

* 1. Смотрим на результат.



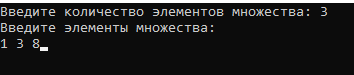
**Рисунок 16 – Ошибка ввода повторяющихся значений**

1. Проверка на добавления большего количества множеств.
   1. Добавление множества в уже добавленный список множеств.



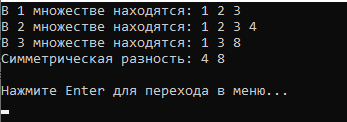
**Рисунок 17 – Добавление третьего множества**

* 1. Заполнение добавленного множества.



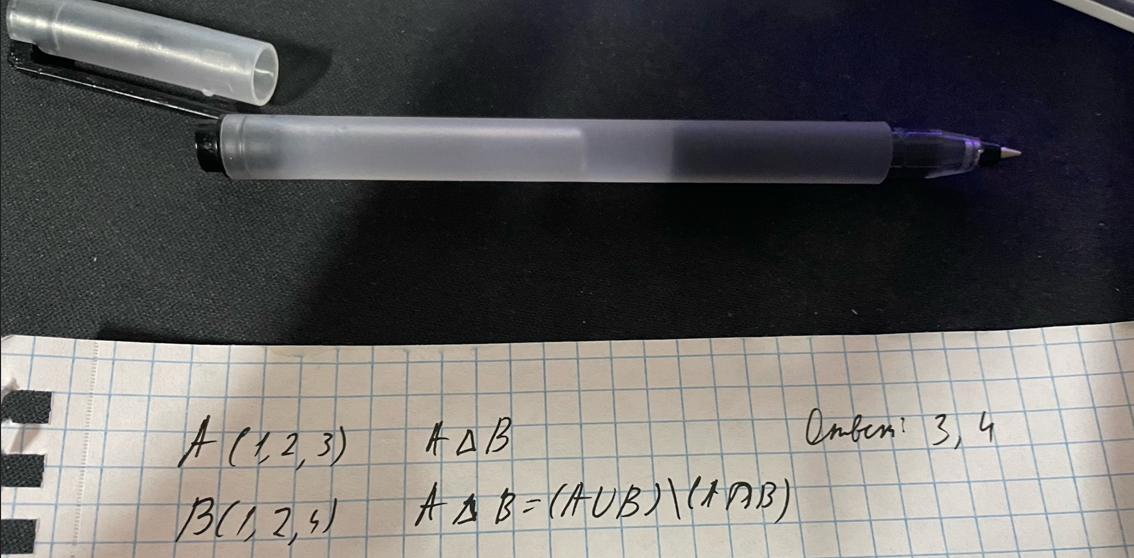
**Рисунок 18 – заполнение третьего множества**

* 1. Проверка на добавление нового множества.



**Рисунок 19 – Проверка на добавление нового множества**

# 6.Ручной расчет



**Рисунок 20 – Ручное решение**

Элементы, которые есть в ( A ), но нет в ( B ):

( 3 ) (в ( A ), но не в ( B ))

Элементы, которые есть в ( B ), но нет в ( A ):

( 4 ) (в ( B ), но не в ( A ))

Результат

Таким образом, симметрическая разность ( A ) и ( B ) будет: [ A \Delta B = {3, 4} ]

Итог

Симметрическая разность множеств ( A ) и ( B ) равна ( {3, 4} ).

# Заключение

В данной курсовой работе была разработана программа на языке C, реализующая операции с множествами, в частности, нахождение симметрической разности двух и более множеств. Программа позволяет пользователю вводить множества, проверять уникальность вводимых элементов и выводить результаты вычислений в среде Microsoft Visual Studio 2022.

В результате выполнения данной курсовой работы была достигнута поставленная цель — разработка программы для работы с множествами, которая может быть полезна как в учебных целях, так и в практических задачах. Программа демонстрирует основные принципы работы с динамическими структурами данных, а также алгоритмы, связанные с множествами, что является важным аспектом в изучении программирования на языке C.

В дальнейшем можно рассмотреть возможность расширения функционала программы, добавив дополнительные операции с множествами, такие как объединение, пересечение и разность, а также улучшить пользовательский интерфейс для повышения удобства использования.

# Список литературы

1. <https://dep_mpmnk.pnzgu.ru/files/dep_mpmnk.pnzgu.ru/posobiya/mnozhestva.pdf>
2. Книга [Язык С (Авторы Керниган Б.В. Ричи Д.М)](https://vk.com/doc531076711_520477325?hash=19kOs5T8ee1GtTxrkyDRqsdGQyO51Jbckt4ZPMU2v98&dl=FjJDoohvFN6QsHnPsBNtvJs6pFjXCbUJX8Y3mJKeR48&api=1&no_preview=1" \t "_blank)
3. [Изучаем программирование на C Дэвид Гриффитс, Дон Гриффитс](https://vk.com/doc531076711_520477325?hash=19kOs5T8ee1GtTxrkyDRqsdGQyO51Jbckt4ZPMU2v98&dl=FjJDoohvFN6QsHnPsBNtvJs6pFjXCbUJX8Y3mJKeR48&api=1&no_preview=1" \t "_blank)

# Приложение А. Листинги программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <windows.h>

#include <time.h>

void symmetric\_difference(int\* setA, int sizeA, int\* setB, int sizeB) {

int\* result = (int\*)malloc((sizeA + sizeB) \* sizeof(int));

int resultSize = 0;

for (int i = 0; i < sizeA; i++) {

int found = 0;

for (int j = 0; j < sizeB; j++) {

if (setA[i] == setB[j]) {

found = 1;

break;

}

}

if (!found) {

result[resultSize++] = setA[i];

}

}

for (int i = 0; i < sizeB; i++) {

int found = 0;

for (int j = 0; j < sizeA; j++) {

if (setB[i] == setA[j]) {

found = 1;

break;

}

}

if (!found) {

result[resultSize++] = setB[i];

}

}

printf("Симметрическая разность: ");

for (int i = 0; i < resultSize; i++) {

printf("%d ", result[i]);

}

printf("\n");

free(result);

}

void input\_set(int\*\* set, int\* size) {

printf("Введите количество элементов множества: ");

scanf("%d", size);

\*set = (int\*)malloc((\*size) \* sizeof(int));

int uniqueCount = 0;

printf("Введите элементы множества:\n");

while (uniqueCount < \*size) {

int element;

scanf("%d", &element);

int exists = 0;

for (int i = 0; i < uniqueCount; i++) {

if ((\*set)[i] == element) {

exists = 1;

break;

}

}

if (!exists) {

(\*set)[uniqueCount++] = element;

}

else {

printf("Элемент %d уже существует в множестве. Пожалуйста, введите другой элемент.\n", element);

}

}

\*size = uniqueCount;

printf("Вы ввели множество: { ");

for (int i = 0; i < \*size; i++) {

printf("%d ", (\*set)[i]);

}

printf("}\n");

}

void input\_random\_set(int\*\* set, int\* size) {

printf("Хотите ли вы задать количество элементов вручную (1) или использовать случайное количество (2)? ");

int choice;

scanf("%d", &choice);

if (choice == 1) {

printf("Введите количество элементов множества: ");

scanf("%d", size);

}

else {

\*size = rand() % 10 + 1;

printf("Случайное количество элементов: %d\n", \*size);

}

\*set = (int\*)malloc((\*size) \* sizeof(int));

int uniqueCount = 0;

while (uniqueCount < \*size) {

int element = rand() % 100;

int exists = 0;

for (int i = 0; i < uniqueCount; i++) {

if ((\*set)[i] == element) {

exists = 1;

break;

}

}

if (!exists) {

(\*set)[uniqueCount++] = element;

}

}

printf("Сгенерированное множество: { ");

for (int i = 0; i < \*size; i++) {

printf("%d ", (\*set)[i]);

}

printf("}\n");

}

void display\_intro() {

printf("===============================================================================================\n");

printf(" Реализация операции нахождения симметрической разности двух и более множеств \n");

printf(" Выполнил: Макаров Алексей Сергеевич \n");

printf(" Учебная группа: 23ВВВ1 \n");

printf("===============================================================================================\n");

printf("\nНажмите Enter для продолжения...\n");

getchar();

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

display\_intro();

int\* sets[2] = { NULL, NULL };

int sizes[2] = { 0, 0 };

int numSets = 0;

int choice;

do {

system("cls");

printf("\n==============================\n");

printf(" МЕНЮ \n");

printf("==============================\n");

printf("1. Ввести множество вручную\n");

printf("2. Создать множество с случайными значениями\n");

printf("3. Найти симметрическую разность\n");

printf("4. Выход\n");

printf("==============================\n");

printf("Выберите опцию: ");

scanf("%d", &choice);

switch (choice) {

case 1: {

if (numSets < 2) {

system("cls");

input\_set(&sets[numSets], &sizes[numSets]);

numSets++;

}

else {

printf("Вы уже ввели два множества.\n");

}

printf("\nНажмите Enter для перехода в меню...\n");

getchar();

getchar();

break;

}

case 2: {

if (numSets < 2) {

system("cls");

input\_random\_set(&sets[numSets], &sizes[numSets]);

numSets++;

}

else {

printf("Вы уже ввели два множества.\n");

}

printf("\nНажмите Enter для перехода в меню...\n");

getchar();

getchar();

break;

}

case 3:

if (numSets == 2) {

system("cls");

printf("Первое множество: ");

for (int j = 0; j < sizes[0]; j++) printf("%d ", sets[0][j]);

printf("\nВторое множество: ");

for (int j = 0; j < sizes[1]; j++) printf("%d ", sets[1][j]);

printf("\n");

symmetric\_difference(sets[0], sizes[0], sets[1], sizes[1]);

printf("\nНажмите Enter для перехода в меню...\n");

getchar();

getchar();

}

else {

printf("Пожалуйста, введите два множества для нахождения симметрической разности.\n");

printf("\nНажмите Enter для перехода в меню...\n");

getchar();

getchar();

}

break;

case 4:

printf("Выход из программы.\n");

break;

default:

printf("Неверный выбор. Пожалуйста, попробуйте снова.\n");

printf("\nНажмите Enter для перехода в меню...\n");

getchar();

getchar();

}

} while (choice != 4);

for (int i = 0; i < numSets; i++) {

free(sets[i]);

}

return 0;

}