Министерство науки и высшего 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 пересечения двух и более множеств»

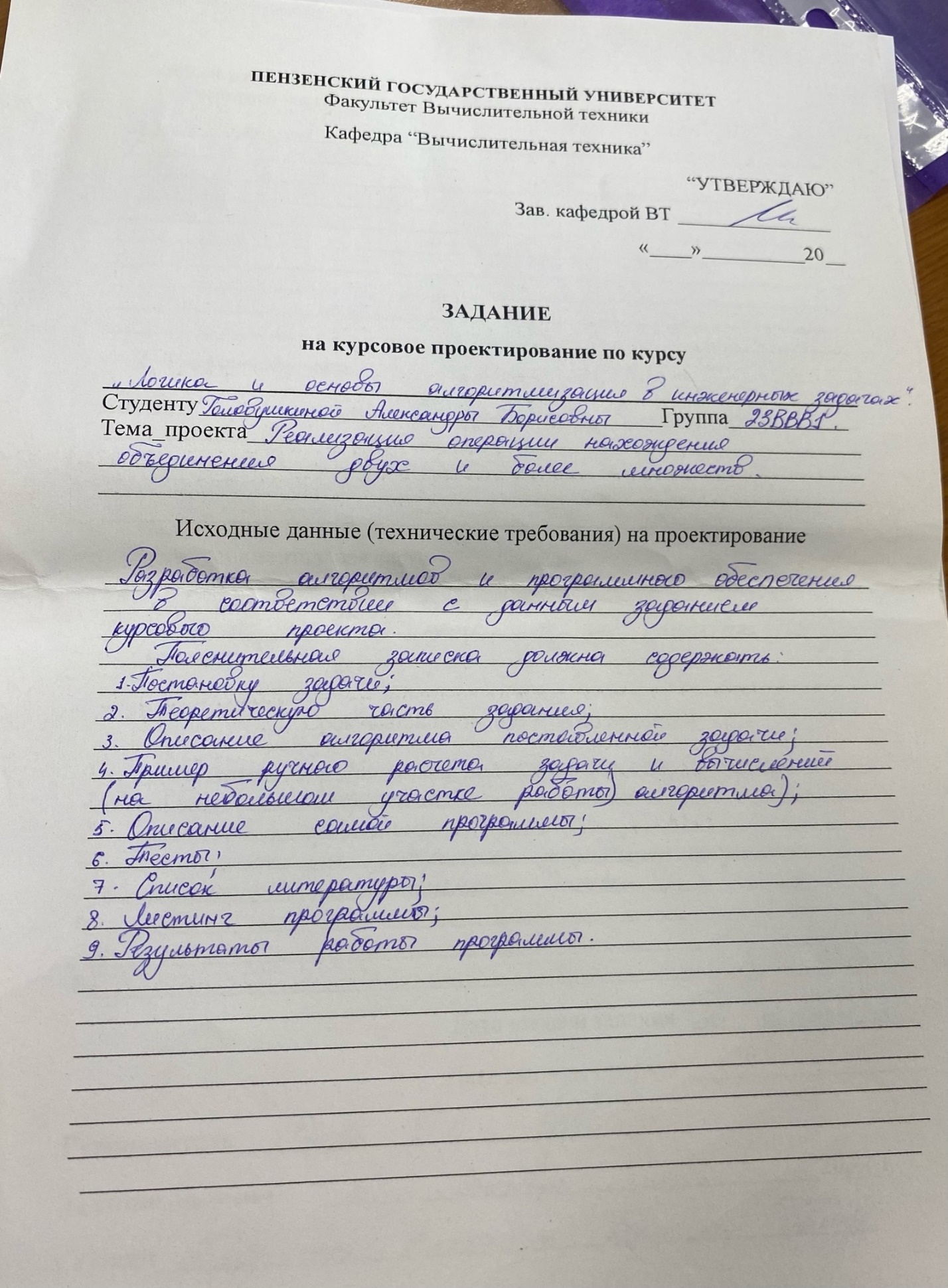
Выполнила:

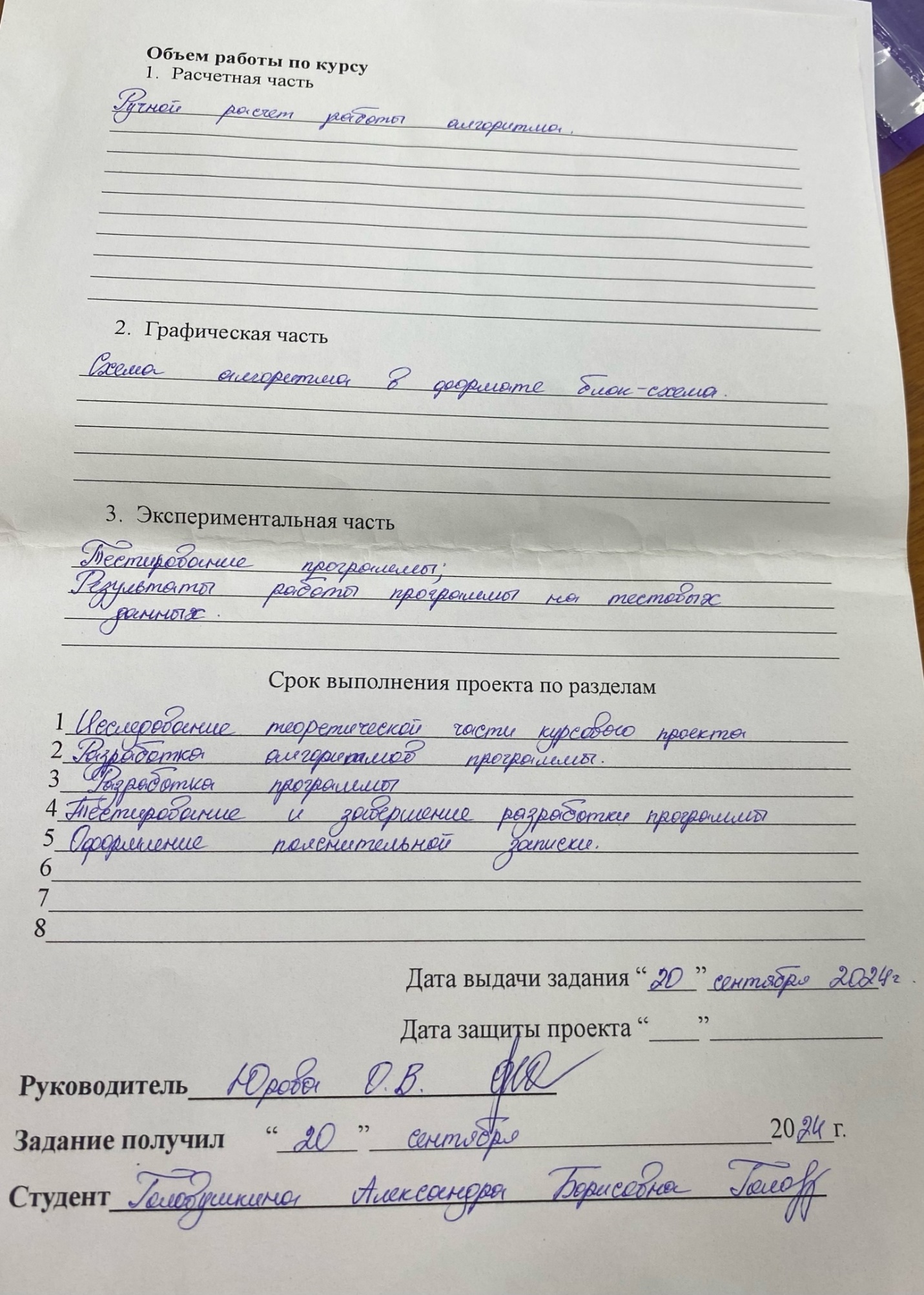
Ст. гр. 23ВВВ1 Головушкина А. Б.

Приняла:

Юрова О. В.

Пенза 2024





**Содержание**

[Реферат 5](#_Toc185201400)

[Введение 6](#_Toc185201401)

[1. Постановка задачи 7](#_Toc185201402)

[2. Теоретическая часть задания 8](#_Toc185201403)

[3. Описание алгоритма программы 9](#_Toc185201404)

[4. Описание программы 11](#_Toc185201405)

[5. Тестирование 15](#_Toc185201406)

[6. Ручная проверка 19](#_Toc185201407)

[Заключение 19](#_Toc185201408)

[Список литературы 20](#_Toc185201409)

[Приложение А. Листинг программы. 21](#_Toc185201410)

Реферат

Отчет 18 стр, 23 рисунка, 1 таблица, 1 приложение

Множества, ПЕРесечение множеств.

Цель работы – разработка программы реализации операции нахождения пересечения двух и более множеств.

В данной курсовой работе рассмотрена задача нахождения пересечения двух и более множеств. Реализована программа на языке C, которая позволяет пользователю вводить несколько массивов (множества) и находить их пересечение. Также предусмотрена возможность автоматического и ручного заполнения массивов. В работе описан алгоритм, использованный для нахождения пересечения, а также проведено тестирование и ручная проверка работы программы.

Введение

Множества — это важная структура данных, которая широко используется в математике, логике и программировании. Одной из базовых операций над множествами является нахождение пересечения двух или более множеств. Пересечение множеств включает все элементы, которые принадлежат одновременно всем рассматриваемым множествам.

Целью данной работы является реализация операции нахождения пересечения для двух и более множеств с использованием языка программирования C. Работа включает в себя описание теоретических аспектов задачи, разработку алгоритма, создание программы и тестирование.

В качестве среды разработки был выбран Microsoft visual studio 2022, язык программирования – C. Целью данной курсовой работы является разработка программы на языке C. Именно с его помощью реализуется алгоритм реализации операции нахождения пересечения двух и более множеств.

1. Постановка задачи

Требуется разработать программу, которая найдёт пересечение двух и более множеств.

Заданы 2 или более одномерных массива(множества). Требуется найти их пересечение, то есть одинаковые элементы во всех множествах, вывести результат в консоль.

Интерфейс ­– консоль.

Устройства ввода – клавиатура и мышь.

1. Теоретическая часть задания

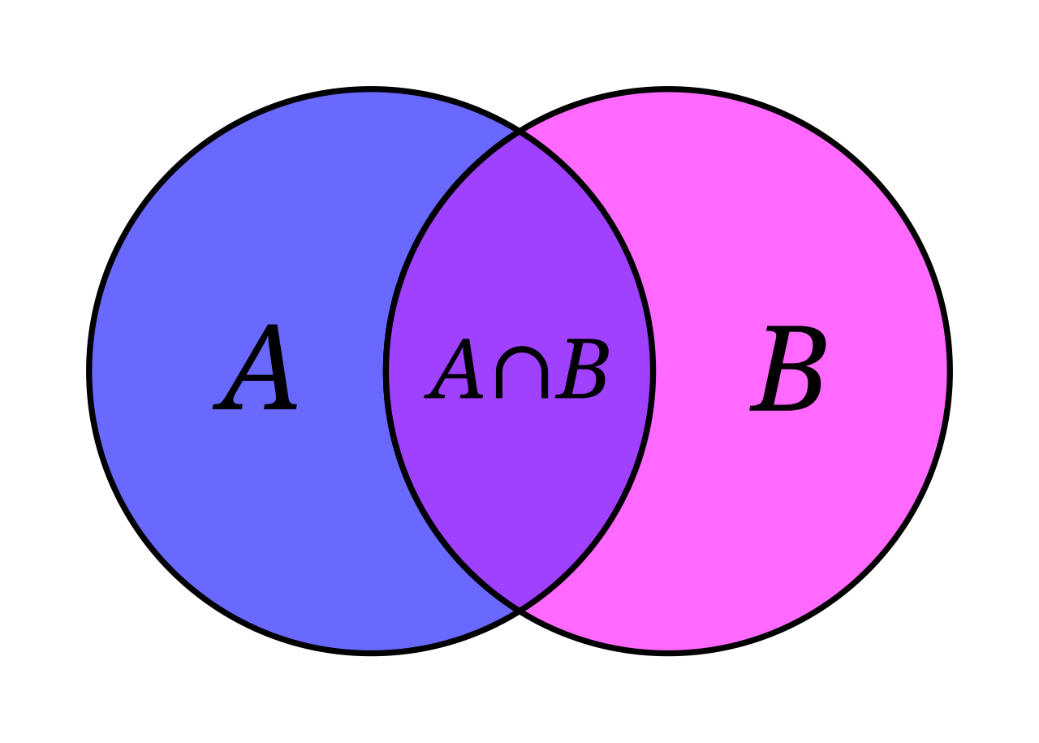
Пересечение множеств в [теории множеств](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2) — это [множество](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE), которому принадлежат те и только те [элементы](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B0), которые одновременно принадлежат всем данным множествам.

Рисунок 1 – Пример пересечения двух множеств

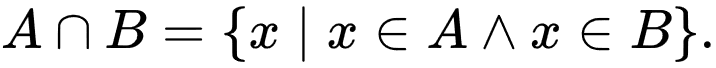
Пусть даны множества А и В. Тогда их пересечением называется множество:

Рисунок 2 –пересечение двух множеств

Пересечением нескольких множеств называется множество, состоящее из элементов, которые есть во всех изначальных множествах

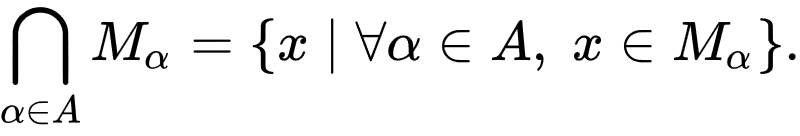


Рисунок 3 –пересечение нескольких множеств

1. Описание алгоритма программы

Для программной реализации понадобиться объявить методы findIntersection и menu.

Работа программы:

* Пользователь выбирает количество множеств
* Пользователь выбирает как заполнить множества: вручную или рандомно.
* Пользователь вводить размеры массивов
* Выполняется алгоритм нахождения пересечения
* Результат выводится на экран

Переменные:

int choice – переменная для выбора пользователя в главном меню

int numArrays – количество массивов(множеств)

int\*\* arrays - динамический массив, представляющий собой массив указателей на массивы чисел (множества). Каждое множество будет храниться в отдельном динамически выделяемом массиве.

int\* sizes - массив, хранящий размеры каждого множества. Этот массив хранит количество элементов для каждого множества.

int fillChoice - переменная для выбора способа заполнения массивов (ручное или автоматическое)

Ниже псевдокодом представлены методы findIntersection(int\*\* arrays, int\* sizes, int numArrays), menu() и main()

**findIntersection(int\*\* arrays, int\* sizes, int numArrays):**

1)создаем пустой массив пересечения размером равным размеру первого множества пересечение\_размер = 0

2)для каждого элемента первого множества (множество[0]):

3)флаг\_общий = истина

4)для каждого множества с индексом от 1 до количество\_массивов - 1:

5)флаг\_найдено = ложь

6)для каждого элемента текущего множества:

7)Если элемент текущего множества равен элементу первого множества:

8)флаг\_найдено = истина

9)выход из цикла

10)если флаг\_найдено = ложь:

11)флаг\_общий = ложь

12)выход из цикла

13)если флаг\_общий = истина:

14)добавляем элемент в массив пересечения

15)увеличиваем размер пересечения на 1

16)вывод "Пересечение: ", пересечение

17)для каждого множества:

18)вывод множество

19)освобождаем память для массива пересечения

20)Конец функции

**Menu():**

1)установить локализацию для русского языка

2)пока выбор != 4:

3)вывести меню

4)считать выбор пользователя

5)если выбор = 1:

6)вывести "Введите количество массивов"

7)считать количество массивов

8)выделить память для массивов и размеров множества

9)если выбор = 2:

10)вывести "Выберите способ заполнения массивов"

11)вывести "1. Ручное заполнение"

12)вывести "2. Автоматическое заполнение"

13)читать выбор способа заполнения

14)для каждого множества от 0 до количество массивов - 1:

15)вывести "Введите размер множества"

16)считать размер множества

17)выделить память для массива

18)если выбор способа заполнения = 1:

19)вывести "Введите элементы множества"

20)для каждого элемента массива:

21)считать элемент

22)иначе если выбор способа заполнения = 2:

23)вывести "Автоматически заполняем множество"

24)для каждого элемента массива:

25)Заполнить случайным числом от 0 до 1

26)если выбор = 3:

27)вызвать функцию findIntersection()

28)если выбор = 4:

29)вывести "Выход из программы"

30)для каждого множества:

31)освободить память

32)освободить память для множества и размеров

33)если выбор не верен:

34)вывести "Неверный выбор, попробуйте снова"

35)очистить экран

36)Конец функции

**Main():**

1)Установить кодировку языка для ввода\вывода

2)Вызвать функцию menu()

3)Конец функции

1. Описание программы

Для написания данной программы использован язык программирования Си. Язык программирования Си – универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

Проект был создан в виде консольного приложения в терминале Windows (консольное приложение C++).

Данная программа является многомодульной поскольку состоит из нескольких методов: findIntersection, menu и int main.

findIntersection – функция для нахождения пересечения нескольких множеств и вывода результата на экран

menu – функция реализует основное меню программы, предоставляя выбор определенных действия для пользователя.

main – точка входа в программу, устанавливающая кодировку ввода\вывода и вызывающая меню.

Работа программы начинается с вывода меню, в котором пользователь выбирает действие, вводит количество массивов, их размерность, вариант заполнения. Затем вычисляется пересечения этих множеств, а результат выводится на экран.

Далее показан начальный вид консоли при запуске программы:

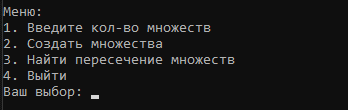


Рисунок 4 – Вывод меню

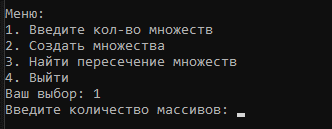


Рисунок 5 – Выбор 1 действия

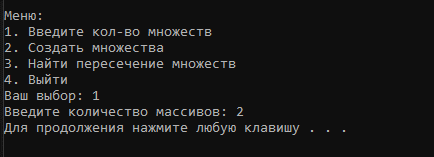


Рисунок 6 – Ввод количества массивов

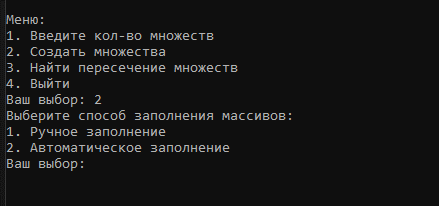


Рисунок 7 – выбор 2 действия

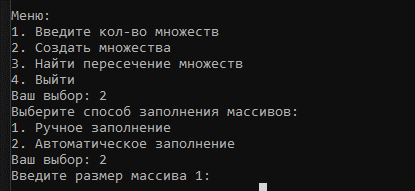


Рисунок 8 – выбор автоматического заполнения массивов

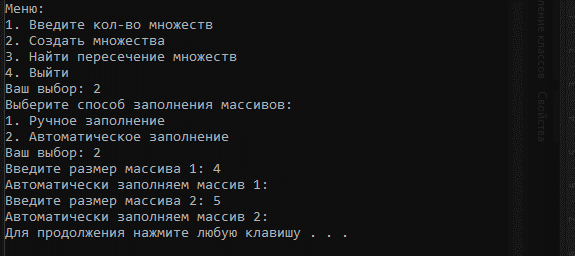


Рисунок 9 – ввод размеров 1 и 2 массива

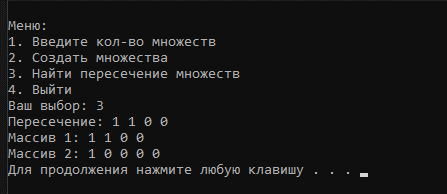


Рисунок 10 – выбор 3 действия: вывод результатов на экран

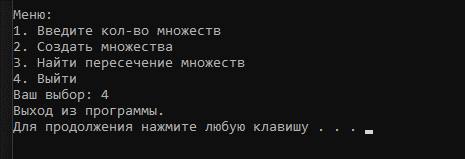


Рисунок 11 – Завершение работы программы из меню

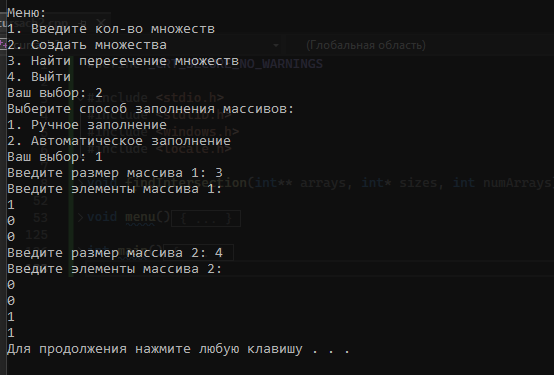


Рисунок 12 – ручной ввод массивов

1. Тестирование

Среда разработки Microsoft visual studio 2022 представляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, алгоритмом пересечения массивов и выводом результатов.

Ниже продемонстрирован результат тестирования функционала программы при вводе количества множеств, выборе способа заполнения массивов, вычислении пересечений и выводе результатов в консоль.

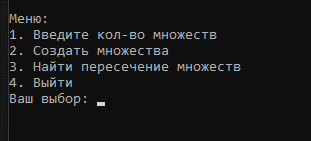


Рисунок 13 – тестирование: запуск программы

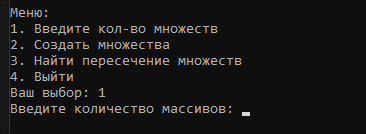


Рисунок 14 – тестирование: выбор ввода количества множеств

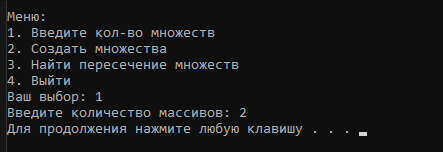


Рисунок 15 – тестирование: ввод количества вершин левой части графа

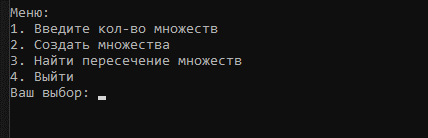


Рисунок 16 – тестирование: нажатие любой клавиши

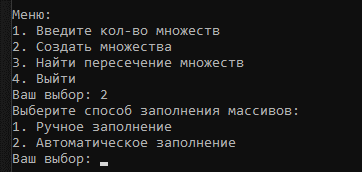


Рисунок 17 – тестирование: ввод создания множеств

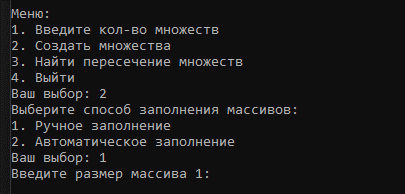


Рисунок 18 – тестирование: выбор ручного заполнения массива

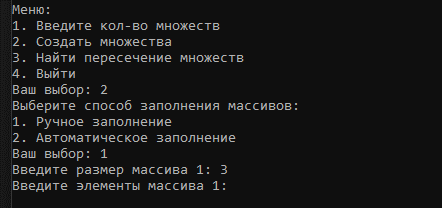


Рисунок 19 – тестирование: ввод размера массива

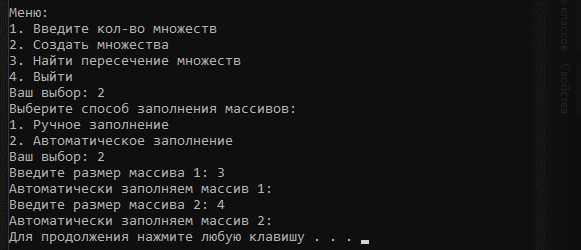


Рисунок 20 – тестирование: автоматическое заполнение массивов

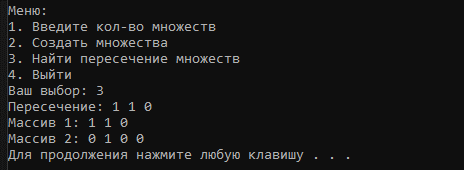


Рисунок 21 – тестирование: выбор найти пересечение множеств

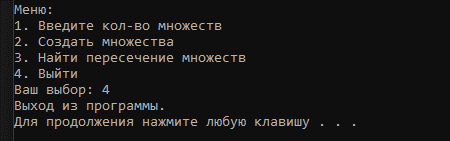


Рисунок 22 – тестирование: выход из программы из меню

Таблица 1 – описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод меню | Верно |
| Выбор ввода количества множества | Ожидание ввода количества множеств | Верно |
| Ввод количества множеств | Ожидание нажатия любой клавиши | Верно |
| Нажатие любой клавиши | Вывод меню | Верно |
| Выбор создания множеств | Ожидание выборы варианта заполнения массивов | Верно |
| Выбор ручного заполнения массива | Ожидание ввода размера массива | Верно |
| Ввод размера массива | Ожидание заполнения массива вручную | Верно |
| Автоматическое заполнения массива | Ожидание ввода размера массива | Верно |
| Выбор найти пересечение множеств | Вывод на экран изначальных массивов и массива пересечений | Верно |
| Выход из программы из меню | Завершение работы программы | Верно |

В результате тестирования было выявлено, что программа успешно проверяет данные на соответствие необходимым требованиям.

1. Ручная проверка



Множество 1: {1, 2, 3}

Множество 2: {2, 3, 4}

Ожидаемое пересечение: {2, 3}

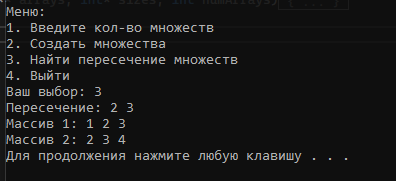


Рисунок 23 – Сравнение ручной проверки с работой пограммы

Таким образом, можно сделать вывод, что программа работает верно.

Заключение

Таким образом, в процессе выполнения данной курсовой работы была разработана программа, реализующая операцию нахождения пересечения двух и более множеств, в Microsoft visual studio 2022.

При выполнении были получены навыки разработки программ, закреплены методы работы с динамической памятью и одномерными массивами. Улучшены навыки владения языком программирования C.

В будущем можно добавить дополнительные функциональные возможности, такие как поддержку множества различных типов данных или оптимизацию алгоритма поиска пересечения для больших наборов данных.

Список литературы

1. Вирт, С. *Алгоритмы и структуры данных*. — М.: Мир, 2001.
2. Кормен, Т. Х., Лейзерсон, Ч. E., Ривест, Р. Л., Стейн, К. *Введение в алгоритмы*. — М.: Издательство "Дом интеллектуальной собственности", 2011.
3. Шамшурин, А. И. *Теория множеств и математическая логика*. — М.: Изд-во МГУ, 2010.
4. Хакимов, С. Г. *Алгоритмы работы с множествами в языке программирования C*. — Программирование и алгоритмы, 2009.
5. Керниган, Б., Ритчи, Д. *Язык программирования C*. — М.: Изд-во "Диалектика", 2008.
6. Бенашвили, И. В. *Программирование на языке C: Учебник для вузов*. — СПб.: Питер, 2011.

Приложение А. Листинг программы.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <locale.h>

void findIntersection(int\*\* arrays, int\* sizes, int numArrays) {

int\* intersection = (int\*)malloc(sizeof(int) \* sizes[0]);

int intersectionSize = 0;

// Поиск пересечения

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

int isCommon = 1;

for (int j = 1; j < numArrays; j++) {

int found = 0;

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

if (arrays[0][i] == arrays[j][k]) {

found = 1;

break;

}

}

if (!found) {

isCommon = 0;

break;

}

}

if (isCommon) {

intersection[intersectionSize++] = arrays[0][i];

}

}

printf("Пересечение: ");

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

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

}

printf("\n");

// Вывод изначальных массивов

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

printf("Массив %d: ", i + 1);

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

printf("%d ", arrays[i][j]);

}

printf("\n");

}

free(intersection);

}

void menu() {

setlocale(LC\_ALL, "Rus");

int choice = 0;

int numArrays;

int\*\* arrays = NULL;

int\* sizes = NULL;

int fillChoice;

do {

printf("\nМеню:\n");

printf("1. Введите кол-во множеств\n");

printf("2. Создать множества\n");

printf("3. Найти пересечение множеств\n");

printf("4. Выйти\n");

printf("Ваш выбор: ");

scanf\_s("%d", &choice);

switch (choice) {

case 1:

printf("Введите количество массивов: ");

scanf("%d", &numArrays);

// Динамическое выделение памяти

arrays = (int\*\*)malloc(numArrays \* sizeof(int\*));

sizes = (int\*)malloc(numArrays \* sizeof(int));

break;

case 2:

printf("Выберите способ заполнения массивов:\n");

printf("1. Ручное заполнение\n");

printf("2. Автоматическое заполнение\n");

printf("Ваш выбор: ");

scanf("%d", &fillChoice);

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

printf("Введите размер массива %d: ", i + 1);

scanf("%d", &sizes[i]);

arrays[i] = (int\*)malloc(sizes[i] \* sizeof(int));

if (fillChoice == 1) {

// Ручное заполнение

printf("Введите элементы массива %d:\n", i + 1);

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

scanf("%d", &arrays[i][j]);

}

}

else if (fillChoice == 2) {

// Автоматическое заполнение

printf("Автоматически заполняем массив %d:\n", i + 1);

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

arrays[i][j] = rand() % 10; // случайные числа от 0 до 1

}

}

}

break;

case 3:

findIntersection(arrays, sizes, numArrays);

break;

case 4:

printf("Выход из программы.\n");

// Освобождение памяти

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

free(arrays[i]);

}

free(arrays);

free(sizes);

break;

default:

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

}

system("pause");

system("cls");

} while (choice != 4);

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

menu();

return 0;

}