Министерство науки и высшего образования РФ

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»

Выполнил студент

группы 21ВВ1.2:

Пивкин Д.А.

Приняли:

д.т.н., профессор Митрохин М.А.

к.т.н., доцент Юрова О.В.

Пенза 2022

Оглавление

[Реферат 5](#_Toc123109244)

[Введение 6](#_Toc123109245)

[1. Постановка задачи 7](#_Toc123109246)

[2. Теоретическая часть 8](#_Toc123109247)

[3. Описание алгоритма программы 9](#_Toc123109248)

[4. Описание программы 10](#_Toc123109249)

[5. Тестирование 13](#_Toc123109250)

[6. Ручной расчёт программы 17](#_Toc123109251)

[Заключение 18](#_Toc123109252)

[Список литературы 19](#_Toc123109253)

[Приложение А 20](#_Toc123109254)

[Листинг программы 20](#_Toc123109255)

# Реферат

Отчет 21 страница, 10 рисунков

РЕАЛИЗАЦИЯ ОПЕРАЦИИ НАХОЖДЕНИЯ ПЕРЕСЕЧЕНИЯ ДВУХ И БОЛЕЕ МНОЖЕСТВ

Цель исследования – реализация операции нахождения пересечения двух и более множеств.

В работе представлена операция нахождения пересечения двух или более множеств.

# Введение

В математике существует несколько операции с множествами над массивами чисел. В данной курсовой работе была рассмотрена операция пересечения, а именно двух или более множеств.

**Пересечение множеств -** это множество, которому принадлежат те и только те элементы, которые одновременно принадлежат всем данным множествам.

Понятие множества обычно принимается заодно из исходных (аксиоматических) понятий, то есть несводимое к другим понятиям, а значит, и не имеющее определения; для его объяснения используются описательные формулировки, характеризующие множество как совокупность различных элементов, мыслимую как единое целое.

В качестве среды разработки мною была выбрана *Microsoft VisualStudio2019*, язык программирования – C++.

Целью данной курсовой работы является разработка программы на языке С++, который является широко используемым. Именно с его помощью в данном курсовом проекте реализуется операция нахождения пересечения двух или более множеств.

# Постановка задачи

Требуется реализовать программу, которая осуществит поиск пересечения двух или более множеств.

Программа должна работать так, чтобы пользователь вводил количество множеств для генерации их в программе. После обработки этих данных на экран должны выводиться заданное количество множеств, сгенерированные случайно или введенные вручную, данные множеств и сам результат пересечения. Необходимо предусмотреть различные варианты поиска, чтобы программа не выдавала ошибок и работа правильно. Устройство ввода - клавиатура и мышь.

# Теоретическая часть

Создается массив, в который мы записываем данные множеств. Пересечение двух наборов формируется только элементами, которые присутствуют в обоих наборах. Элементы, скопированные функцией, всегда поступают из первого диапазона в том же порядке. Элементы в обоих диапазонах уже должны быть упорядочены.

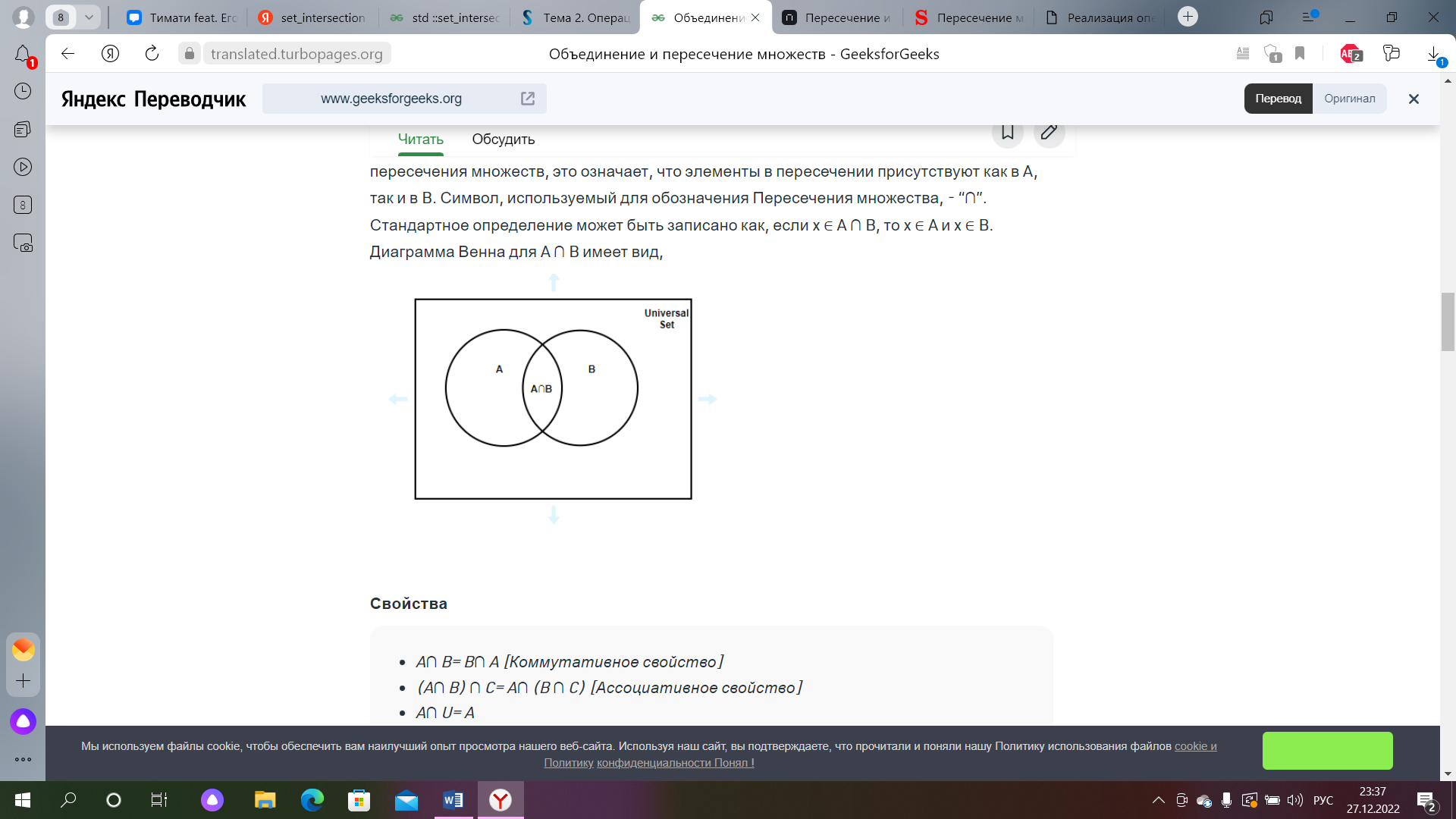


Рисунок 1 – Пересечение множеств в виде кругов Эйлера

На вход алгоритму поступает массив, где происходит работа с итераторами. Где каждый элемент сравнивается друг с другом. Если элементы разные, то они удаляются, если одинаковые, то выводятся на экран программой.

Результатом работы алгоритма является одинаковые элементы пересечения множеств.

# Описание алгоритма программы

Для программной реализации алгоритма понадобятся один массив: arr – массив для хранения данных.

Также нам понадобится работа с классом set для хранения и извлечения данных из массива. Его применение обуславливается тем, чтобы наши переменные в массиве и в переменную для сравнения являлись одним и тем же.

Сам алгоритм работает при помощи 2-ух итераторов и массива. В цикле мы заносим переменные первого множества в один итератор, а переменные следующего множества в другой. После чего в массиве происходит сравнение двух итераторов. Если они равны, то значениях этих итераторов передаем в результат работы. Если же они разные, значение одного итератора меняется и происходит сравнение с другими данными этого множества.

Алгоритм выполняется до тех пор, пока не сравнит все значения множества.

# Описание программы

Для написания данной программы использован язык программирования С++. Язык программирования С++ – 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 *Win32(Visual C++).*

Данная программа многомодульная, поскольку состоит из нескольких функций: main, Peresech.

Работа программы начинается с запроса количества множеств. Если пользователь выбрал количество множеств, то на экран выводится запрос на выбор режима. Если пользователь выбрал пункт «Случайное заполнение», то на экран выводятся сгенерированные случайным образом множества. Если же пользователь выбрал пункт «Заполнить самому», то на экран выводится запрос на заполнение множеств данными. Также предусмотрен выход из программы.

cout << "Введите кол-во множест: "; cin >> z;

if (z >= 2) break;

else { cout << "Ошибка. Число должно быть больше или равно 2\n"; }

system("pause");

}

set<int>\* arr = new set<int>[z]; // Наши множества

// используем set чтобы не повторялись множества в них(одно и тоже число)

system("cls");

for (int m = 0; m < z; m++) {

arr[m].erase(arr[m].begin(), arr[m].end()); // очистка списков для повтора

}

cout << "Количество множеств: " << z

<< "\n Выберете способ заполнения множеств: \n 1. Случайное заполнение \n" << " 2. Заполнить самому \n 3. Выход\n Введите: ";

cin >> n;

if (n == 1) {

for (int i = 0; i < z; i++)

{

for (int j = 0; j < 2 + rand() % 8; j++)

{

arr[i].insert(rand() % 20);

}

cout << endl << "Множество №" << i + 1 << ": " << endl;

for (auto k : arr[i])

cout << k << " ";

}

Peresech(arr, z);

cout << endl;

system("pause");

}

if (n == 2) {

system("cls");

cout << "Заполните ваши множества\n";

for (int i = 0; i < z; i++)

{

cout << endl << "Множество №" << i + 1 << "\nЧтобы остановить ввод введите \*" << endl;

string s;

while (true) {

cin >> s;

if (s == "\*") break;

else arr[i].insert(atoi(s.c\_str()));

}

}

Peresech(arr, z);

cout << endl;

system("pause");

}

if (n == 3) {

exit(0);

}

delete[]arr;

}

return 0;

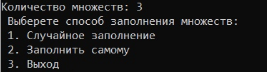
}

Рисунок 2 – Приветствие

Ниже можно увидеть оформление начального запроса и дальнейшего действия с ним:

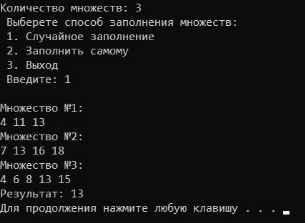


Рисунок 3 – Автоматическая генерация матрицы

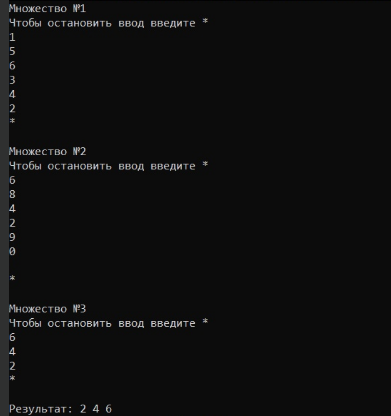


Рисунок 4 – Ручной ввод

# Тестирование

Среда разработки Microsoft Visual Studio 2019 представляет все средства, необходимые при разработке и отладки многомодульных программ.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество ошибок и проблем, связанных с вводом данных, изменением дизайна вводимых данных, алгоритмом программы, взаимодействием функций.

Ниже продемонстрирован результат тестирования программы при вводе пользователем различное количество множеств.

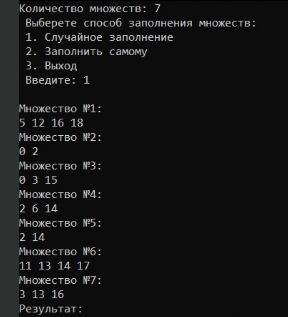
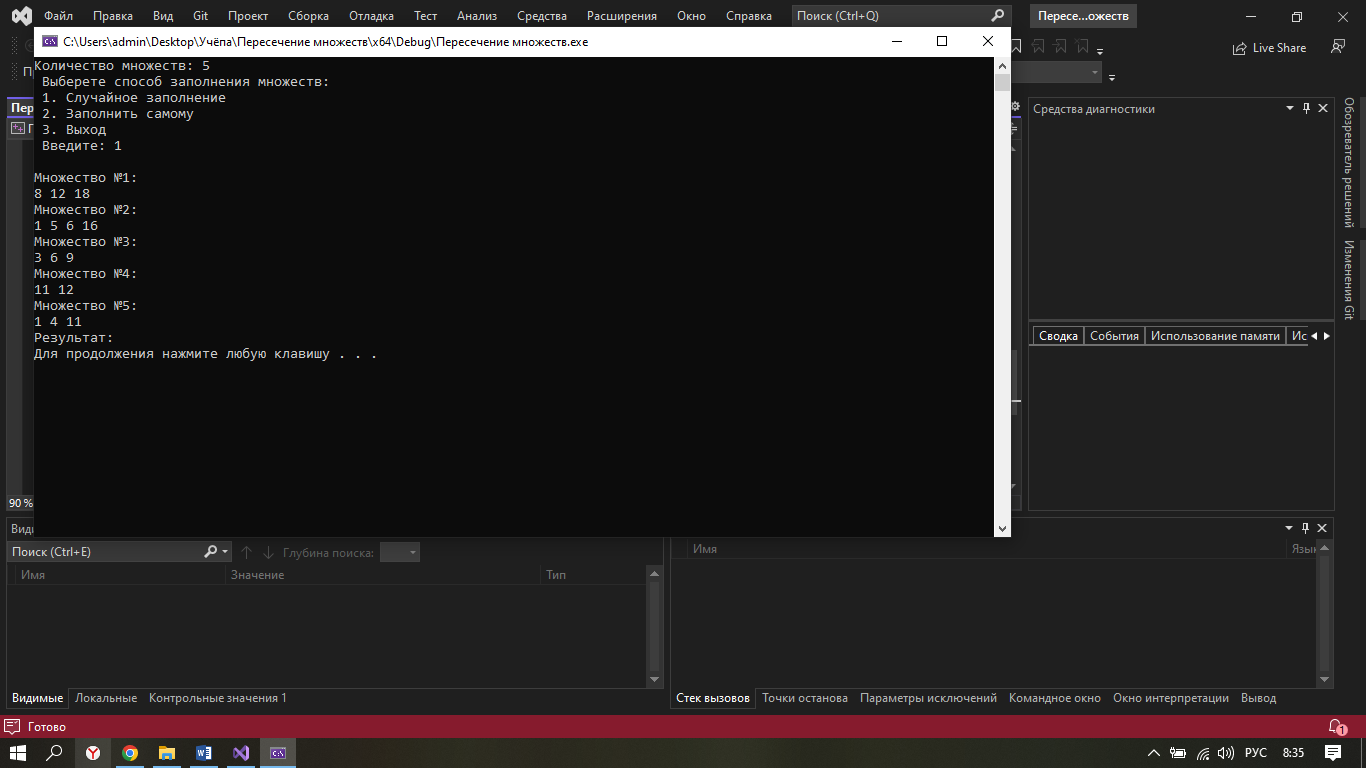


Рисунок 5 – Работа программы при автоматической генерации семи множеств



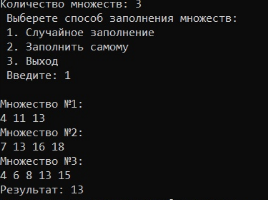
Рисунок 6 - Работа программы при автоматической генерации пяти множеств

Рисунок 7 - Работа программы при автоматической генерации трех множеств

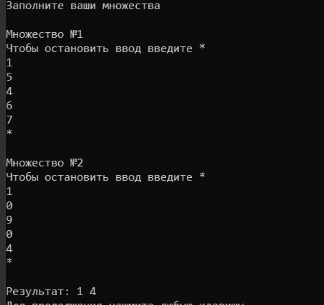


Рисунок 8 - Работа программы при ручном вводе двух множеств

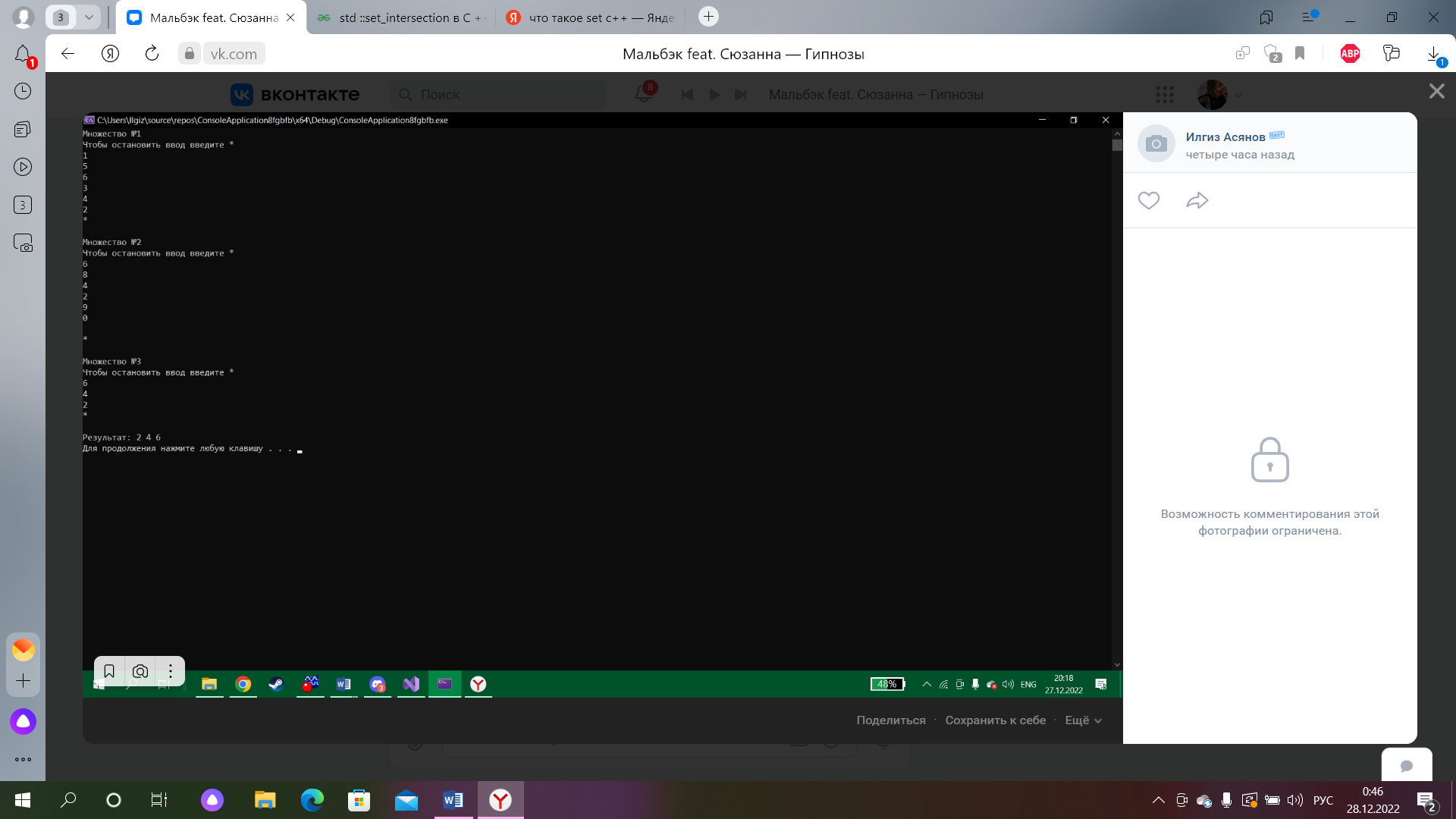


Рисунок 9 - Работа программы при ручном вводе трех множеств

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод меню, вывод сообщения « Введите кол-во множеств» | Верно |
| Выбор генерации матрицы | Вывод сообщения о количестве множеств | Верно |
| Ввод матрицы с клавиатуры | Вывод сообщения о количестве множеств, ввод элементов, вывод элементов | Верно |
| Вывод результата | Вывод правильного результата на разно-размерных множествах, идентичность с ручным расчетом | Верно |
| Правильность работы алгоритма | Совпадение ручных расчетов с результатом работы алгоритма | Верно |

**Таблица**

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

# Ручной расчёт программы

Проведем проверку программы посредством ручных вычислений на примере пересечения трех множеств (рисунок 7).

Мы можем наблюдать, что при генерации множеств случайном образом были получены следующие данные. В первом множестве находятся числа 4,11,13. Во втором множестве числа 7,13,16,18, в третьем – 4,8,13,15.

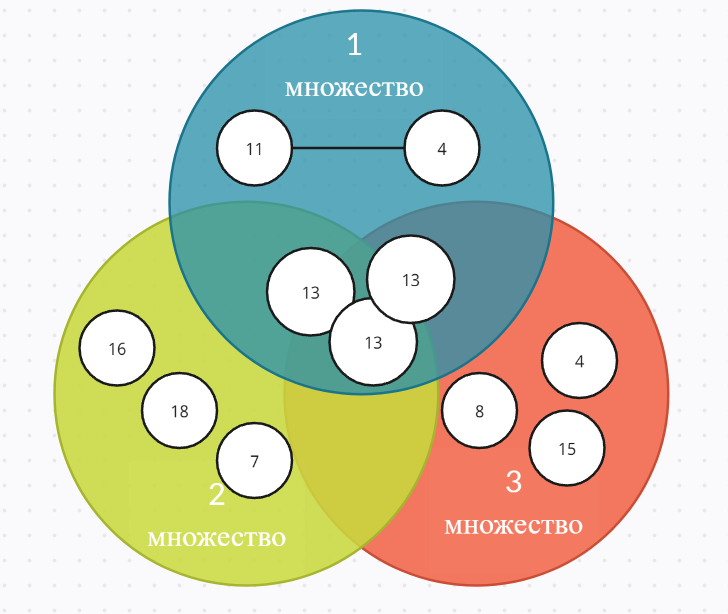
Если построить круги Эйлера, то можно увидеть, что в каждом множестве присутствует число 13.

Рисунок 10 – Круги Эйлера

Результат ручных расчетов совпадает с результатом работы алгоритма, таким образом можно сделать вывод, что программа работает верно.

# Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая операцию нахождения пересечения двух и более множеств в *MicrosoftVisualStudio 2019*.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежности, а также работы с новыми алгоритмами. Углублены навыки знания языка программирования С++.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

# Список литературы

1. Н.Я.Виленкин «Рассказы о множествах»,2002, 151 с.
2. Герберт Шилдт «Полный справочник по С++» - вильямс,2006
3. А.В.Маркова «Элементы теории множеств» 2014, 126 с.

# Приложение А

# Листинг программы

#include <iostream>

#include <set>

#include "Windows.h"

using namespace std;

void Peresech(set<int>\* arr, int z) {

set<int>end; // Конечный результат

end = arr[0];

set<int>::iterator it1;

for (int i = 1; i < z; i++)

{

it1 = end.begin();

set<int>::iterator it2 = arr[i].begin();

while ((it1 != end.end()) && (it2 != arr[i].end())) {

if (\*it1 < \*it2) {

end.erase(it1++);

}

else if (\*it2 < \*it1) {

++it2;

}

else { // \*it1 == \*it2

++it1;

++it2;

}

}

}

end.erase(it1, end.end());

cout << endl << "Результат: ";

for (auto i : end)

cout << i << " ";

}

int main()

{

setlocale(LC\_ALL, "");

srand(time(NULL));

for (;;) {

int z, n;

while (true) {

system("cls");

cout << "Введите кол-во множеств: "; cin >> z;

if (z >= 2) break;

else { cout << "Ошибка. Число должно быть больше или равно 2\n"; }

system("pause");

}

set<int>\* arr = new set<int>[z]; // Наши множества

// используем set чтобы не повторялись множества в них(одно и тоже число)

system("cls");

for (int m = 0; m < z; m++) {

arr[m].erase(arr[m].begin(), arr[m].end()); // очистка списков для повтора

}

cout << "Количество множеств: " << z

<< "\n Выберете способ заполнения множеств: \n 1. Случайное заполнение \n" << " 2. Заполнить самому \n 3. Выход\n Введите: ";

cin >> n;

if (n == 1) {

for (int i = 0; i < z; i++)

{

for (int j = 0; j < 2 + rand() % 8; j++)

{

arr[i].insert(rand() % 20);

}

cout << endl << "Множество №" << i + 1 << ": " << endl;

for (auto k : arr[i])

cout << k << " ";

}

Peresech(arr, z);

cout << endl;

system("pause");

}

if (n == 2) {

system("cls");

cout << "Заполните ваши множества\n";

for (int i = 0; i < z; i++)

{

cout << endl << "Множество №" << i + 1 << "\nЧтобы остановить ввод введите \*" << endl;

string s;

while (true) {

cin >> s;

if (s == "\*") break;

else arr[i].insert(atoi(s.c\_str()));

}

}

Peresech(arr, z);

cout << endl;

system("pause");

}

if (n == 3) {

exit(0);

}

delete[]arr;

}

return 0;

}