Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

ФИСТ Кафедра «Вычислительная техника»

Дисциплина «Математическая логика и дискретная математика»

Лабораторная работа №1

«Операции с двумя множествами»

Вариант №7

Выполнил студент

Группы ИВТИИбд-12,

Белоглазкин А. А.

Проверила:

к.т.н., доцент

Игонин А.Г.

Ульяновск, 2025

**Оглавление**

[Постановка задачи 3](#_Toc201258452)

[Реализация 3](#_Toc201258453)

[Тестирование 6](#_Toc201258454)

[Заключение 7](#_Toc201258455)

[Список литературы 8](#_Toc201258456)

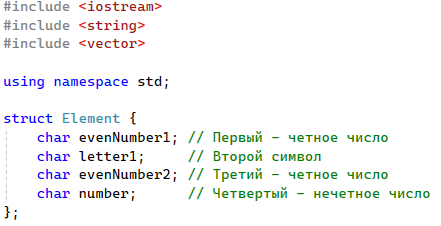
[Приложение 9](#_Toc201258457)

# **Постановка задачи**

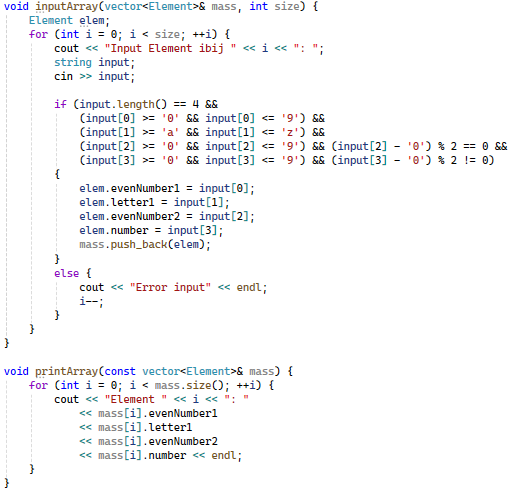
В данной работе необходимо разработать программу, которая будет принимать на ввод два множества и выполнять с ними следующие действия: объединение, пересечение, дополнение A\B и B\A и симметрическую разность. Ввод данных должен осуществляться пользователем. Вариантом был назначен типа ввода: c – цифра, b – буква, i – четная цифра, j – нечетная цифра. Наш вариант был ibij.

# **Реализация**

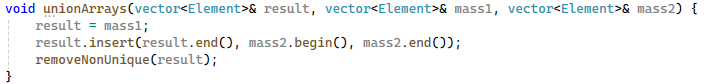
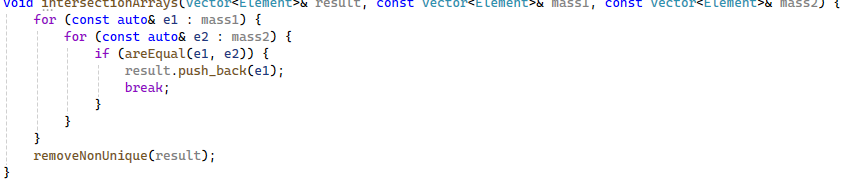
Программа начинается с подключения необходимых библиотек и описания структуры `Element`, которая содержит четыре символа: первый символ — цифра, второй — строчная латинская буква, третий — четное число, четвертый — нечетное число.

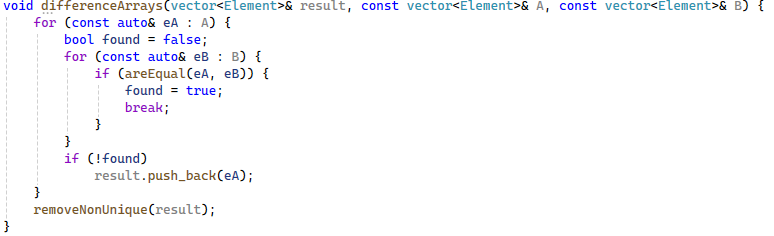


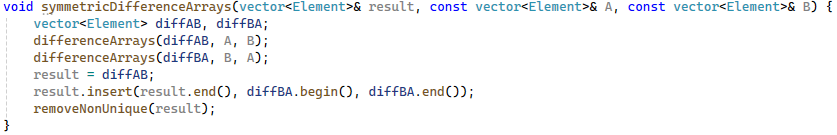
Далее идут вспомогательные функции, включая проверку корректности ввода и удаление повторяющихся элементов. Пользователь вводит два множества A и B, указывая количество элементов, после чего программа проверяет каждый введённый элемент на соответствие требованиям. Если элемент некорректен, программа выдает сообщение об ошибке и требует повторного ввода.



После ввода и очистки от дубликатов программа выполняет основные операции над множествами:

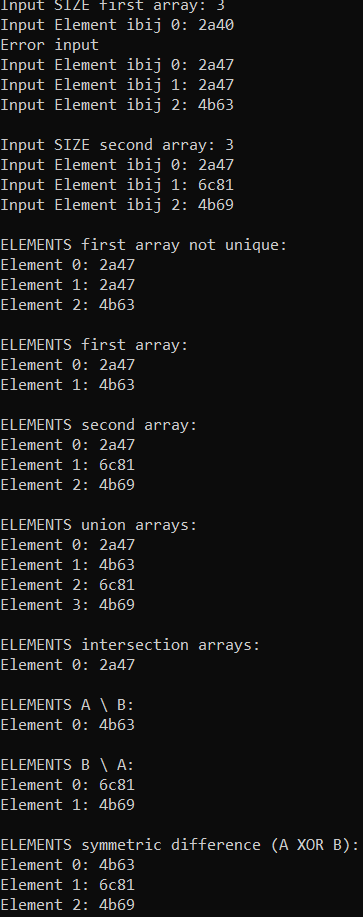
1. Объединение
2. Пересечение
3. Дополнение



1. Симметрическую разность

Результаты всех операций выводятся на экран. Таким образом, программа позволяет пользователю ввести два множества с определённой структурой элементов, выполнить над ними стандартные множественные операции и получить итоговые наборы данных для каждого случая.

# **Тестирование**



# **Заключение**

В ходе проделанной работы мы познакомились с понятием множества и научились реализовывать операции над множествами в программе. Мы разработали структуру для хранения элементов множества, состоящих из четырёх символов с определёнными условиями: первый и третий — чётные цифры, второй — строчная латинская буква, четвёртый — нечётная цифра. Далее мы реализовали функции проверки корректности ввода, удаления повторяющихся элементов и выполнения основных множественных операций: объединения, пересечения, дополнение, а также симметрической разности. Через консольный ввод мы вводили множества A и B, после чего программа отображала результат выполнения всех операций, что позволило на практике увидеть, как работают основные свойства и функции множеств.

# **Список литературы**

* + - * 1. Р. Хаггарти, Дискретная математика для программистов // Москва: Техносфера — 2003. - С. 44 — 51.

# **Приложение**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

struct Element {

char evenNumber1; // Первый - четное число

char letter1; // Второй символ

char evenNumber2; // Третий - четное число

char number; // Четвертый - нечетное число

};

void inputArray(vector<Element>& mass, int size) {

Element elem;

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

cout << "Input Element ibij " << i << ": ";

string input;

cin >> input;

if (input.length() == 4 &&

(input[0] >= '0' && input[0] <= '9') &&

(input[1] >= 'a' && input[1] <= 'z') &&

(input[2] >= '0' && input[2] <= '9') && (input[2] - '0') % 2 == 0 &&

(input[3] >= '0' && input[3] <= '9') && (input[3] - '0') % 2 != 0)

{

elem.evenNumber1 = input[0];

elem.letter1 = input[1];

elem.evenNumber2 = input[2];

elem.number = input[3];

mass.push\_back(elem);

}

else {

cout << "Error input" << endl;

i--;

}

}

}

void printArray(const vector<Element>& mass) {

for (int i = 0; i < mass.size(); ++i) {

cout << "Element " << i << ": "

<< mass[i].evenNumber1

<< mass[i].letter1

<< mass[i].evenNumber2

<< mass[i].number << endl;

}

}

void removeNonUnique(vector<Element>& mass) {

vector<Element> uniqueElements;

for (int i = 0; i < mass.size(); ++i) {

bool isUnique = true;

for (int j = 0; j < uniqueElements.size(); ++j) {

if (mass[i].evenNumber1 == uniqueElements[j].evenNumber1 &&

mass[i].letter1 == uniqueElements[j].letter1 &&

mass[i].evenNumber2 == uniqueElements[j].evenNumber2 &&

mass[i].number == uniqueElements[j].number) {

isUnique = false;

break;

}

}

if (isUnique) {

uniqueElements.push\_back(mass[i]);

}

}

mass = uniqueElements;

}

bool areEqual(const Element& a, const Element& b) {

return a.evenNumber1 == b.evenNumber1 &&

a.letter1 == b.letter1 &&

a.evenNumber2 == b.evenNumber2 &&

a.number == b.number;

}

//объединение

void unionArrays(vector<Element>& result, vector<Element>& mass1, vector<Element>& mass2) {

result = mass1;

result.insert(result.end(), mass2.begin(), mass2.end());

removeNonUnique(result);

}

//пересечение

void intersectionArrays(vector<Element>& result, const vector<Element>& mass1, const vector<Element>& mass2) {

for (const auto& e1 : mass1) {

for (const auto& e2 : mass2) {

if (areEqual(e1, e2)) {

result.push\_back(e1);

break;

}

}

}

removeNonUnique(result);

}

//дополнение

void differenceArrays(vector<Element>& result, const vector<Element>& A, const vector<Element>& B) {

for (const auto& eA : A) {

bool found = false;

for (const auto& eB : B) {

if (areEqual(eA, eB)) {

found = true;

break;

}

}

if (!found)

result.push\_back(eA);

}

removeNonUnique(result);

}

//симметричная разность

void symmetricDifferenceArrays(vector<Element>& result, const vector<Element>& A, const vector<Element>& B) {

vector<Element> diffAB, diffBA;

differenceArrays(diffAB, A, B);

differenceArrays(diffBA, B, A);

result = diffAB;

result.insert(result.end(), diffBA.begin(), diffBA.end());

removeNonUnique(result);

}

int main() {

int size1, size2;

vector<Element> mass1, mass2, mass\_unification;

cout << "Input SIZE first array: ";

cin >> size1;

inputArray(mass1, size1);

cout << "\n";

cout << "Input SIZE second array: ";

cin >> size2;

inputArray(mass2, size2);

cout << "\n";

cout << "ELEMENTS first array not unique: " << endl;

printArray(mass1);

cout << "\n";

removeNonUnique(mass1);

cout << "ELEMENTS first array: " << endl;

printArray(mass1);

cout << "\n";

cout << "ELEMENTS second array: " << endl;

printArray(mass2);

cout << "\n";

removeNonUnique(mass2);

unionArrays(mass\_unification, mass1, mass2);

cout << "ELEMENTS union arrays: " << endl;

printArray(mass\_unification);

cout << "\n";

vector<Element> mass\_intersection;

intersectionArrays(mass\_intersection, mass1, mass2);

cout << "ELEMENTS intersection arrays: " << endl;

printArray(mass\_intersection);

cout << "\n";

vector<Element> mass\_difference\_AB;

differenceArrays(mass\_difference\_AB, mass1, mass2);

cout << "ELEMENTS A \\ B: " << endl;

printArray(mass\_difference\_AB);

cout << "\n";

vector<Element> mass\_difference\_BA;

differenceArrays(mass\_difference\_BA, mass2, mass1);

cout << "ELEMENTS B \\ A: " << endl;

printArray(mass\_difference\_BA);

cout << "\n";

vector<Element> mass\_sym\_diff;

symmetricDifferenceArrays(mass\_sym\_diff, mass1, mass2);

cout << "ELEMENTS symmetric difference (A XOR B): " << endl;

printArray(mass\_sym\_diff);

cout << "\n";

return 0;

}