Министерство образования и науки РФ

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Дискретная математика

Лабораторная работа № 1

Тема: «Калькулятор множеств»

Выполнил: студентка группы ИВТ-23-1б

Хорошилова К. П.

Проверил: ст. преподаватель

Рустамханова Г. И.

г. Пермь – 2024**Цель работы**

Разработать калькулятор множеств на С++ с возможностью вводить более 3 множеств.

**Задачи работы**

1. Предоставить возможность задать более 3 множеств.

2. Задать множества в универсуме: диапазон от -50 до 50.

3. Задать способы задания множеств: случайное, вручную, по совокупности условий (соблюдение знака, кратность числу, проверка на универсум).

4. Реализовать операции пересечение, объединение, симметрическая разность, дополнение до универсума.

5. Позволение написать формулу.

6. Оформить отчет.

# Код программы

#include <iostream>

#include <vector>

#include <algorithm>

#include <cstdlib>

#include <ctime>

using namespace std;

const int UNIVERSE\_MIN = -50; // Нижняя граница универсума

const int UNIVERSE\_MAX = 50; // Верхняя граница универсума

// Функция для вывода множества

void printSet(const std::vector<int>& set) {

if (set.empty()) {

std::cout << "Пустое множество" << std::endl;

return;

}

std::cout << "{ ";

for (int elem : set) {

std::cout << elem << " ";

}

std::cout << "}" << std::endl;

}

// Функция для создания случайного множества

std::vector<int> generateRandomSet(int numElements) {

std::vector<int> set;

srand(static\_cast<unsigned>(time(0)));

while (set.size() < numElements) {

int elem = UNIVERSE\_MIN + rand() % (UNIVERSE\_MAX - UNIVERSE\_MIN + 1);

if (std::find(set.begin(), set.end(), elem) == set.end()) {

set.push\_back(elem);

}

}

return set;

}

// Функция для создания множества вручную

std::vector<int> createSetManually() {

int size;

std::cout << "Введите количество элементов множества: ";

std::cin >> size;

std::vector<int> set;

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

int elem;

std::cout << "Введите элемент #" << i + 1 << ": ";

std::cin >> elem;

if (elem >= UNIVERSE\_MIN && elem <= UNIVERSE\_MAX) {

set.push\_back(elem);

}

else {

std::cout << "Элемент вне диапазона универсума [-50, 50], попробуйте еще раз." << std::endl;

--i;

}

}

return set;

}

// Операции над множествами

std::vector<int> unionSets(const std::vector<int>& setA, const std::vector<int>& setB) {

std::vector<int> result = setA;

for (int elem : setB) {

if (std::find(result.begin(), result.end(), elem) == result.end()) {

result.push\_back(elem);

}

}

return result;

}

std::vector<int> intersectionSets(const std::vector<int>& setA, const std::vector<int>& setB) {

std::vector<int> result;

for (int elem : setA) {

if (std::find(setB.begin(), setB.end(), elem) != setB.end()) {

result.push\_back(elem);

}

}

return result;

}

// Функция для создания множества по условиям

std::vector<int> createSetByConditions() {

std::vector<int> set;

char sign;

int multiple;

int minRange, maxRange;

// Получение условий от пользователя

std::cout << "Задание знака (+ или -): ";

std::cin >> sign;

std::cout << "Задание кратности числу: ";

std::cin >> multiple;

std::cout << "Задание диапазона (минимум и максимум в пределах [-50, 50]): ";

std::cin >> minRange >> maxRange;

// Проверка на правильность диапазона

if (minRange < UNIVERSE\_MIN || maxRange > UNIVERSE\_MAX || minRange > maxRange) {

std::cout << "Неверный диапазон!" << std::endl;

return set; // Возвращаем пустое множество при неверных данных

}

// Формирование множества по условиям

for (int i = minRange; i <= maxRange; ++i) {

if (i % multiple == 0 && ((sign == '+' && i > 0) || (sign == '-' && i < 0))) {

set.push\_back(i); // Добавляем элементы, соответствующие знаку и кратности

}

}

return set;

}

// Функция для симметрической разности двух множеств

std::vector<int> symmetricDifferenceSets(const std::vector<int>& setA, const std::vector<int>& setB) {

std::vector<int> result;

// Добавляем элементы, которые есть в первом множестве, но отсутствуют во втором

for (int elem : setA) {

if (std::find(setB.begin(), setB.end(), elem) == setB.end()) {

result.push\_back(elem);

}

}

// Добавляем элементы, которые есть во втором множестве, но отсутствуют в первом

for (int elem : setB) {

if (std::find(setA.begin(), setA.end(), elem) == setA.end()) {

result.push\_back(elem);

}

}

return result;

}

std::vector<int> complementToUniverse(const std::vector<int>& set) {

std::vector<int> result;

for (int i = UNIVERSE\_MIN; i <= UNIVERSE\_MAX; ++i) {

if (std::find(set.begin(), set.end(), i) == set.end()) {

result.push\_back(i);

}

}

return result;

}

// Функция для обработки выражений

std::vector<int> evaluateExpression(const std::vector<std::vector<int>>& sets, char operation, int setIndexA, int setIndexB) {

std::vector<int> result;

switch (operation) {

case '+':

result = unionSets(sets[setIndexA], sets[setIndexB]);

std::cout << "Результат объединения: ";

break;

case '\*':

result = intersectionSets(sets[setIndexA], sets[setIndexB]);

std::cout << "Результат пересечения: ";

break;

case '^':

result = symmetricDifferenceSets(sets[setIndexA], sets[setIndexB]);

std::cout << "Результат симметрической разности: ";

break;

case '-':

result = complementToUniverse(sets[setIndexA]);

std::cout << "Дополнение до универсума: ";

break;

default:

std::cout << "Неверная операция." << std::endl;

return result;

}

return result;

}

int main() {

setlocale(LC\_ALL, "Russian");

std::vector<std::vector<int>> sets; // Вектор для хранения всех множеств

int choice;

do {

// Основное меню программы

std::cout << "\nМеню:\n";

std::cout << "1. Создать случайное множество\n";

std::cout << "2. Ввести множество вручную\n";

std::cout << "3. Создать множество по условиям\n";

std::cout << "4. Операции над множествами (через выражения)\n";

std::cout << "5. Выход\n";

std::cout << "Выберите действие: ";

std::cin >> choice;

switch (choice) {

case 1: {

int numElements;

std::cout << "Введите количество элементов: ";

std::cin >> numElements;

sets.push\_back(generateRandomSet(numElements));

std::cout << "Сгенерированное множество: ";

printSet(sets.back());

break;

}

case 2: {

sets.push\_back(createSetManually());

std::cout << "Введенное множество: ";

printSet(sets.back());

break;

}

case 3: {

// Создание множества по условиям

sets.push\_back(createSetByConditions());

std::cout << "Множество по условиям: ";

printSet(sets.back());

break;

}

case 4: {

if (sets.size() < 2) {

std::cout << "Для выполнения операций нужно как минимум 2 множества." << std::endl;

break;

}

// Ввод выражения

int setIndexA, setIndexB;

char operation;

std::cout << "Введите выражение (например, 0 + 1 для объединения): ";

std::cin >> setIndexA >> operation >> setIndexB;

if (setIndexA < 0 || setIndexA >= sets.size() || setIndexB < 0 || setIndexB >= sets.size()) {

std::cout << "Неверные индексы множеств." << std::endl;

break;

}

std::vector<int> result = evaluateExpression(sets, operation, setIndexA, setIndexB);

printSet(result);

break;

}

case 5:

std::cout << "Выход из программы." << std::endl;

break;

default:

std::cout << "Неверный выбор, попробуйте снова." << std::endl;

break;

}

} while (choice != 5);

return 0;

}

**Анализ кода**

Данная лабораторная работа реализует программу для работы с множествами, включая их создание, выполнение операций, и обработку выражений.

Программа начинается с объявления констант, определяющих границы универсума: UNIVERSE\_MIN и UNIVERSE\_MAX. Эти границы используются для проверки корректности элементов множеств.

Функция printSet отвечает за вывод множества на экран. Если множество пустое, выводится сообщение "Пустое множество". В противном случае элементы выводятся в фигурных скобках, разделенных пробелами. Эта функция используется во многих частях программы для отображения результатов операций.

Функция generateRandomSet генерирует случайное множество заданного размера. В процессе генерации используется функция rand, чтобы выбрать случайные числа в пределах универсума. Если сгенерированный элемент уже существует в множестве, он пропускается, чтобы избежать дублирования. Генерация завершается, когда достигается заданное количество уникальных элементов.

Функция createSetManually позволяет пользователю вручную ввести множество. Пользователь задает размер множества и вводит каждый элемент по очереди. Если введенный элемент не входит в границы универсума, программа выводит предупреждение, и элемент предлагается ввести заново.

Функция unionSets реализует операцию объединения двух множеств. Она копирует элементы первого множества, а затем добавляет элементы второго множества, которые еще не присутствуют в первом. Это гарантирует, что результат содержит все уникальные элементы из обоих множеств.

Функция intersectionSets выполняет операцию пересечения, возвращая только те элементы, которые присутствуют в обоих множествах. Для каждого элемента первого множества проверяется его наличие во втором.

Функция createSetByConditions создает множество на основе заданных пользователем условий. Пользователь задает знак чисел (+ или -), кратность определенному числу и диапазон. Программа проверяет корректность диапазона и затем добавляет в множество только те числа, которые соответствуют всем указанным условиям.

Функция symmetricDifferenceSets реализует симметрическую разность двух множеств. В результат добавляются элементы, которые присутствуют в одном из множеств, но отсутствуют в другом. Это достигается последовательной проверкой каждого множества относительно другого.

Функция complementToUniverse вычисляет дополнение множества относительно универсума. Она добавляет в результат те элементы универсума, которые отсутствуют во входном множестве. Это позволяет получить множество всех элементов, которые не входят в заданное множество.

Функция evaluateExpression обрабатывает выражения, выполняя выбранную пользователем операцию над двумя множествами. Пользователь указывает индексы множеств и тип операции (+ для объединения, \* для пересечения, ^ для симметрической разности, - для дополнения). Функция вызывает соответствующую операцию и возвращает результат.

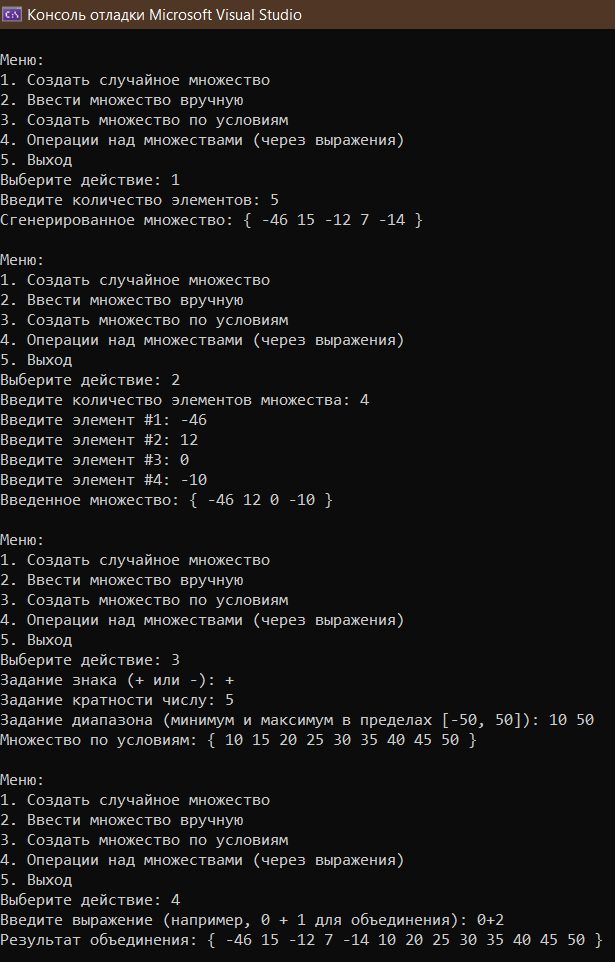
В функции main реализован основной цикл программы. Пользователю предоставляется меню с возможностью:

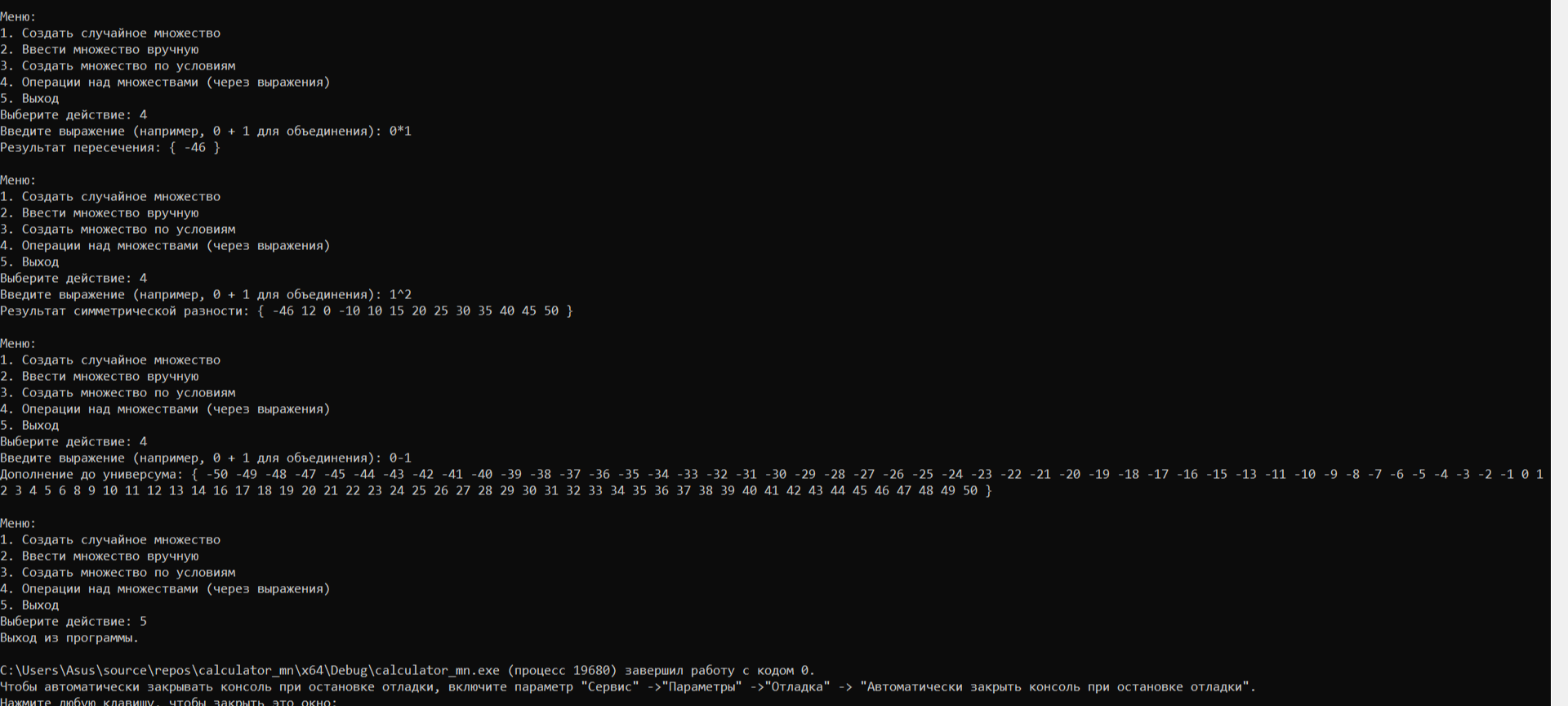
1. Создать случайное множество.
2. Ввести множество вручную.
3. Создать множество по условиям.
4. Выполнить операции над множествами.
5. Выйти из программы.

В зависимости от выбора вызывается соответствующая функция. Результаты операций и созданные множества сохраняются в векторе sets, чтобы их можно было использовать повторно.

Ссылка на Github: <https://github.com/ksinph/disc_math>

**Тесты**





# Заключение

# В ходе лабораторной работы я создала калькулятор множеств реализованный на С++.