Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования

**«Пермский национальный исследовательский
политехнический университет»**

Электротехнический факультет
Кафедра «Информационные технологии и автоматизированные системы»
направление подготовки: 09.03.01– «Информатика и вычислительная техника»

Лабораторная работа
по дисциплине
«Дискретная математика и математическая логика»
на тему
«Калькулятор множеств»

Выполнил студент гр. ИВТ-23-1б
Попонин Михаил Александрович

Проверил:

_____

_____

_____ _____
(оценка) (подпись)

_____
(дата)

г. Пермь, 2024

Цель работы и задачи работы

Целью данной работы является разработка калькулятора множеств

Требования к калькулятору

1) Предоставить возможность задать более 3-х множеств
2) Множества в универсуме: диапазон от -50 до 50
3) Способы задания множеств
    a. Случайно
    b. Вручную
    c. По совокупности условий
4) Позволение написать формулу
5) Пустые множества выводить текстом или знаком

Код программы

Таблица 1 – весь код программы

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <stack>
#include <string>
#include <map>
using namespace std;

const int MAX_SIZE = 100;
const int UNIVERSE_MIN = -50;
const int UNIVERSE_MAX = 50;

void inputSetManual(int set[], int &size)
{
    cout << "Введите количество элементов множества: ";
    cin >> size;
    cout << "Введите элементы множества: ";
    for (int i = 0; i < size; i++)
    {
        cin >> set[i];
    }
}
void unionSets(int set1[], int size1, int set2[], int size2, int
resultSet[], int &resultSize)
{
    resultSize = 0;
    for (int i = 0; i < size1; i++)
    {
        resultSet[resultSize++] = set1[i];
    }
    for (int i = 0; i < size2; i++)
    {
        bool found = false;
        for (int j = 0; j < size1; j++)
        {
            if (set2[i] == set1[j])
            {
                found = true;
                break;
            }
        }
        if (!found)
        {
            resultSet[resultSize++] = set2[i];
        }
    }
}
void intersectSets(int set1[], int size1, int set2[], int size2,
int resultSet[], int &resultSize)
```

```
{
    resultSize = 0;
    for (int i = 0; i < size1; i++)
    {
        for (int j = 0; j < size2; j++)
        {
            if (set1[i] == set2[j])
            {
                resultSet[resultSize++] = set1[i];
                break;
            }
        }
    }
}
void differenceSets(int set1[], int size1, int set2[], int size2,
int resultSet[], int &resultSize)
{
    resultSize = 0;
    for (int i = 0; i < size1; i++)
    {
        bool found = false;
        for (int j = 0; j < size2; j++)
        {
            if (set1[i] == set2[j])
            {
                found = true;
                break;
            }
        }
        if (!found)
        {
            resultSet[resultSize++] = set1[i];
        }
    }
}
void symmetricDifferenceSets(int set1[], int size1, int set2[],
int size2, int resultSet[], int &resultSize)
{
    resultSize = 0;
    for (int i = 0; i < size1; i++)
    {
        bool found = false;
        for (int j = 0; j < size2; j++)
        {
            if (set1[i] == set2[j])
            {
                found = true;
                break;
            }
        }
        if (!found)
        {
            resultSet[resultSize++] = set1[i];
```

```
        }
    }
    for (int i = 0; i < size2; i++)
    {
        bool found = false;
        for (int j = 0; j < size1; j++)
        {
            if (set2[i] == set1[j])
            {
                found = true;
                break;
            }
        }
        if (!found)
        {
            resultSet[resultSize++] = set2[i];
        }
    }
}
void inputSetRandom(int set[], int &size)
{
    cout << "Введите количество элементов множества: ";
    cin >> size;
    srand(time(0));
    for (int i = 0; i < size; i++)
    {
        set[i] = UNIVERSE_MIN + rand() % (UNIVERSE_MAX -
UNIVERSE_MIN + 1);
    }
}
void inputSetConditional(int set[], int &size)
{
    int minVal, maxVal, multiple, signChoice;
    cout << "\nВведите минимальное и максимальное значения
диапазона (должны быть в пределах универсума от -50 до 50): ";
    cin >> minVal >> maxVal;
    if (minVal < UNIVERSE_MIN || maxVal > UNIVERSE_MAX)
    {
        cout << "Диапазон выходит за пределы универсума!" <<
endl;
        return;
    }
    cout << "Выберите знак элементов множества (1 -
положительные, 2 - отрицательные, 3 - без ограничения): ";
    cin >> signChoice;
    cout << "Введите кратность какому-либо числу (если не нужно,
введите 1): ";
    cin >> multiple;
    cout << "Введите количество элементов множества: ";
    cin >> size;
    int count = 0;
    for (int i = minVal; i <= maxVal && count < size; i++)
    {
```

```
        bool valid = true;
        if (signChoice == 1 && i < 0) valid = false;
        if (signChoice == 2 && i > 0) valid = false;
        if (i % multiple != 0) valid = false;
        if (valid)
        {
            set[count++] = i;
        }
    }
    if (count < size)
    {
        cout << "Не удалось сгенерировать требуемое количество
элементов!" << endl;
        size = count;
    }
}
void bubbleSort(int set[], int size)
{
    for (int i = 0; i < size - 1; i++)
    {
        for (int j = 0; j < size - i - 1; j++)
        {
            if (set[j] > set[j + 1])
            {
                int temp = set[j];
                set[j] = set[j + 1];
                set[j + 1] = temp;
            }
        }
    }
}
void printSet(int set[], int size)
{
    if (size == 0)
    {
        cout << "Пустое множество" << endl;
    }
    else
    {
        bubbleSort(set, size);
        cout << "{ ";
        for (int i = 0; i < size; i++)
        {
            cout << set[i] << (i < size - 1 ? ", " : " ");
        }
        cout << "}" << endl;
    }
}
void printSetAfterInput(int set[], int size, int setNumber)
{
    cout << "Множество " << setNumber << " после заполнения: ";
    printSet(set, size);
}
```

```
void complementSetToUniverse(int set[], int size, int
resultSet[], int &resultSize)
{
    resultSize = 0;
    for (int i = UNIVERSE_MIN; i <= UNIVERSE_MAX; i++)
    {
        bool found = false;
        for (int j = 0; j < size; j++)
        {
            if (set[j] == i)
            {
                found = true;
                break;
            }
        }
        if (!found)
        {
            resultSet[resultSize++] = i;
        }
    }
}
void processExpression(string expression, int sets[][MAX_SIZE],
int sizes[], int resultSet[], int &resultSize, int numSets)
{
    stack<int> setStack;
    stack<char> opStack;
    map<char, int> precedence = {{'\\', 1}, {'&', 2}, {'|', 3},
{'^', 4}, {'~', 5}};

    int i = 0;
    while (i < expression.length())
    {
        if (isdigit(expression[i]))
        {
            int setIndex = 0;
            while (i < expression.length() &&
isdigit(expression[i]))
            {
                setIndex = setIndex * 10 + (expression[i] - '0');
                i++;
            }
            i--;
            setIndex--;
            if (setIndex >= 0 && setIndex < numSets)
            {
                setStack.push(setIndex);
            }
            else
            {
                cout << "Ошибка: Неверный индекс множества!" <<
endl;
                return;
            }
```

```
        }
        else if (expression[i] == '\\' || expression[i] == '&' ||
expression[i] == '|' || expression[i] == '^' || expression[i] ==
'~')
        {
            opStack.push(expression[i]);
        }
        else if (expression[i] == '(' || expression[i] == ')')
        {
            continue;
        }
        i++;
    }

    while (!opStack.empty())
    {
        char op = opStack.top();
        opStack.pop();
        int tempSet[MAX_SIZE];
        int tempSize;
        if (op == '~')
        {
            int set1 = setStack.top();
            setStack.pop();
            complementSetToUniverse(sets[set1], sizes[set1],
tempSet, tempSize);
        }
        else
        {
            int set2 = setStack.top(); setStack.pop();
            int set1 = setStack.top(); setStack.pop();

            if (op == '\\')
            {
                differenceSets(sets[set1], sizes[set1],
sets[set2], sizes[set2], tempSet, tempSize);
            }
            else if (op == '&')
            {
                intersectSets(sets[set1], sizes[set1],
sets[set2], sizes[set2], tempSet, tempSize);
            }
            else if (op == '|')
            {
                unionSets(sets[set1], sizes[set1], sets[set2],
sizes[set2], tempSet, tempSize);
            }
            else if (op == '^')
            {
                symmetricDifferenceSets(sets[set1], sizes[set1],
sets[set2], sizes[set2], tempSet, tempSize);
            }
        }
```

```
        for (int i = 0; i < tempSize; i++)
        {
            sets[numSets][i] = tempSet[i];
        }
        sizes[numSets] = tempSize;
        setStack.push(numSets);
        numSets++;
    }
    int finalSetIndex = setStack.top();
    resultSize = sizes[finalSetIndex];
    for (int i = 0; i < resultSize; i++)
    {
        resultSet[i] = sets[finalSetIndex][i];
    }
}
int main()
{
    const int MAX_SETS = 10;
    int sets[MAX_SETS][MAX_SIZE];
    int sizes[MAX_SETS];
    int numSets;
    int resultSet[MAX_SIZE];
    int resultSize;
    int inputMethod;
    cout << "Введите количество множеств: ";
    cin >> numSets;
    if (numSets < 2)
    {
        cout << "Количество множеств должно быть больше 1!" <<
endl;
        return 1;
    }
    for (int i = 0; i < numSets; i++)
    {
        cout << "\nМножество " << i + 1 << ":" << endl;
        cout << "Выберите метод ввода множества: 1 - вручную, 2 -
случайно, 3 - по условиям: ";
        cin >> inputMethod;
        switch (inputMethod) {
            case 1:
                inputSetManual(sets[i], sizes[i]);
                printSetAfterInput(sets[i], sizes[i], i + 1);
                break;
            case 2:
                inputSetRandom(sets[i], sizes[i]);
                printSetAfterInput(sets[i], sizes[i], i + 1);
                break;
            case 3:
                inputSetConditional(sets[i], sizes[i]);
                printSetAfterInput(sets[i], sizes[i], i + 1);
                break;
            default:
```

```
                cout << "Неправильный выбор метода ввода!" <<
endl;
                return 1;
        }
    }
    int choice;
    cout << "\n |=====================================|\n"
            " | Выберите операцию над множествами:   |\n"
            " | 1 - Объединение                      |\n"
            " | 2 - Пересечение                      |\n"
            " | 3 - Разность                         |\n"
            " | 4 - Симметрическая разность          |\n"
            " | 5 - Дополнение до универсума         |\n"
            " | 6 - Ввод выражения                   |\n"
            " |=====================================|\n\n"
            "Ваш выбор:  ";
    cin >> choice;
    string expression;
    switch (choice)
    {
        case 1:
            unionSets(sets[0], sizes[0], sets[1], sizes[1],
resultSet, resultSize);
            for (int i = 2; i < numSets; i++) {
                unionSets(resultSet, resultSize, sets[i],
sizes[i], resultSet, resultSize);
            }
            cout << "Объединение: ";
            printSet(resultSet, resultSize);
            break;
        case 2:
            intersectSets(sets[0], sizes[0], sets[1], sizes[1],
resultSet, resultSize);
            for (int i = 2; i < numSets; i++) {
                intersectSets(resultSet, resultSize, sets[i],
sizes[i], resultSet, resultSize);
            }
            cout << "Пересечение: ";
            printSet(resultSet, resultSize);
            break;
        case 3:
            differenceSets(sets[0], sizes[0], sets[1], sizes[1],
resultSet, resultSize);
            for (int i = 2; i < numSets; i++) {
                differenceSets(resultSet, resultSize, sets[i],
sizes[i], resultSet, resultSize);
            }
            cout << "Разность: ";
            printSet(resultSet, resultSize);
            break;
        case 4:
            symmetricDifferenceSets(sets[0], sizes[0], sets[1],
sizes[1], resultSet, resultSize);
```

```
            for (int i = 2; i < numSets; i++) {
                symmetricDifferenceSets(resultSet, resultSize,
sets[i], sizes[i], resultSet, resultSize);
            }
            cout << "Симметрическая разность: ";
            printSet(resultSet, resultSize);
            break;
        case 5:
            complementSetToUniverse(sets[0], sizes[0], resultSet,
resultSize);
            cout << "Дополнение до универсума: ";
            printSet(resultSet, resultSize);
            break;
        case 6:
            cout << "\n  |==================================|\n"
                    "   | Введите выражение над множествами |\n"
                    "   | ~ - Дополнение                    |\n"
                    "   | ^ - Симметрическая разность       |\n"
                    "   | \\ - Разность                      |\n"
                    "   | | - Объединение                   |\n"
                    "   | & - Пересечение                   |\n"
                    "   |==================================|\n\n"
                    "    Ваше выражение: ";
            cin.ignore();
            getline(cin, expression);
            processExpression(expression, sets, sizes, resultSet,
resultSize, numSets);
            cout << "Результат выражения: ";
            printSet(resultSet, resultSize);
            break;
        default:
            cout << "Неправильный выбор!" << endl;
    }
    return 0;
}
```

Объяснение функций

1) inputSetManual(int set[], int &size)

Функция для ручного ввода множества. Пользователю предлагается ввести количество элементов и сами элементы множества.

2) unionSets(int set1[], int size1, int set2[], int size2, int resultSet[], int &resultSize)

Выполняет объединение двух множеств. Добавляет все элементы из первого множества и только уникальные элементы из второго множества в результат.

3) intersectSets(int set1[], int size1, int set2[], int size2, int resultSet[], int &resultSize)

Выполняет пересечение двух множеств, сохраняя только те элементы, которые присутствуют в обоих множествах.

4) differenceSets(int set1[], int size1, int set2[], int size2, int resultSet[], int &resultSize)

Вычисляет разность между двумя множествами, оставляя только элементы первого множества, которых нет во втором.

5) symmetricDifferenceSets(int set1[], int size1, int set2[], int size2, int resultSet[], int &resultSize)

Вычисляет симметрическую разность двух множеств, сохраняя элементы, которые есть только в одном из них, но не в обоих.

6) inputSetRandom(int set[], int &size)

Заполняет множество случайными числами из диапазона универсума (от -50 до 50). Пользователь задает количество элементов.

7) inputSetConditional(int set[], int &size)

Заполняет множество элементами, соответствующими условиям: диапазон значений, знак чисел и кратность. Пользователь задает параметры.

8) bubbleSort(int set[], int size)

Сортирует множество по возрастанию методом пузырьковой сортировки.

9) printSet(int set[], int size)

Выводит множество на экран, предварительно отсортировав его.

10) printSetAfterInput(int set[], int size, int setNumber)

Выводит множество после его заполнения с указанием номера множества.

11) complementSetToUniverse(int set[], int size, int resultSet[], int &resultSize)

Вычисляет дополнение множества до универсума. Универсум — это все значения от -50 до 50. В результат записываются только те элементы, которых нет в исходном множестве.

12) processExpression(string expression, int sets[][MAX_SIZE], int sizes[], int resultSet[], int &resultSize, int numSets)

Обрабатывает выражение над множествами, содержащее операции: объединение, пересечение, разность, симметрическая разность и дополнение до универсума. Использует стек для вычисления выражений.

13) main()

Главная функция программы. Позволяет пользователю выбрать количество множеств, способ ввода данных (вручную, случайно, по условиям), а затем предлагает выбрать операцию над множествами: объединение, пересечение, разность, симметрическая разность, дополнение или ввод выражения.

Примеры работы программы

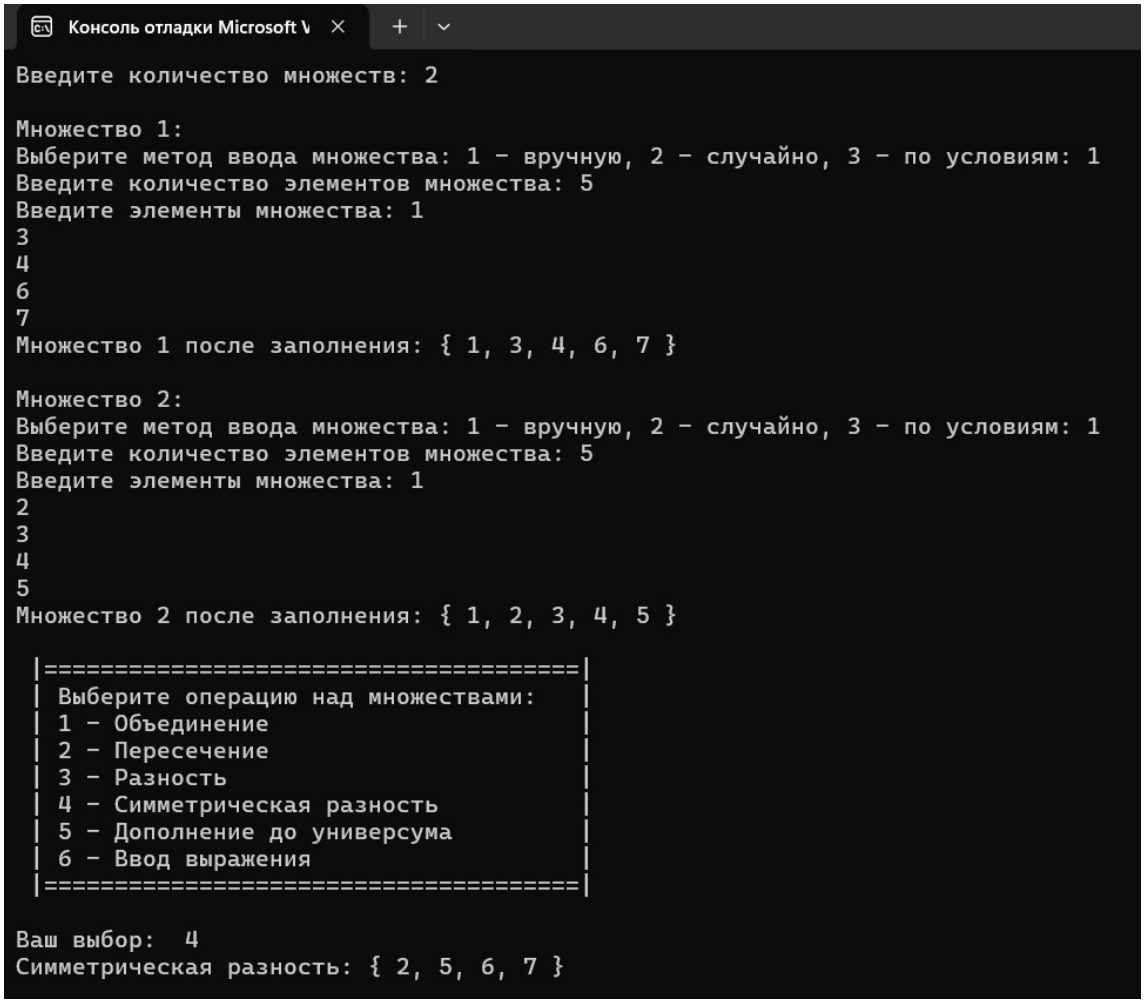

Рисунок 1 – пример первый

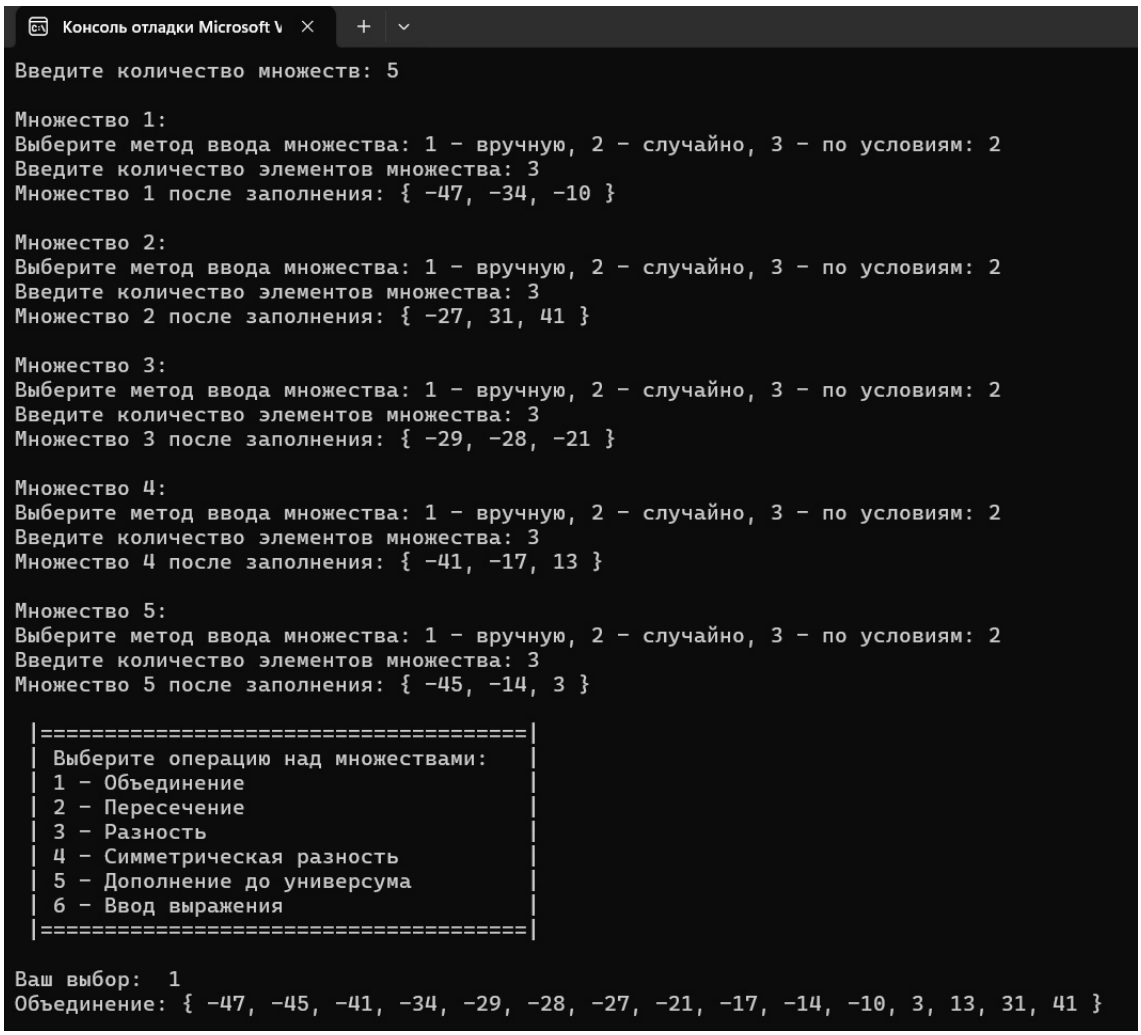

Рисунок 2 – пример второй

Рисунок 3 – пример третий