Министерство образования и науки РФ

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Дискретная математика и математическая логика

Лабораторная работа №1

Тема: «Калькулятор множеств»

Выполнил: студент группы ИВТ-23-1б

Адаев Даниил Дмитриевич

Проверил: Рустамханова Г. И.

**Задачи работы**

Разработать калькулятор множеств.

1. Предоставить возможность задать более 3 множеств.
2. Множества в универсуме: диапазон от -50 до 50.
3. Пустые множества выводить текстом или знаком.
4. Способы задания множеств:   
   - Вручную  
   - Случайное: по совокупности условий. Соблюдение знака, кратность какому-либо числу, в заданном диапазоне с проверкой на принадлежность универсуму, количество элементов в множестве.
5. Решение формулы. Операции пересечения, объединения, разности, исключающего или, дополнения до универсума.

**Анализ задачи**

Программа реализована на С++.

Для действий над множествами создан класс Set, содержащий множество и методы отображения, вставки, размера. Set имеет перегруженные операторы &, |, -, ^, и ~, выполняющие функции AND, OR, DIF (разность), XOR и NOT (дополнение) соответственно

Заданные параметры случайного множества: нет знака, минимум: -50, максимум: 50, количество элементов: 10, кратность: 1.

Для хранения множеств используется вектор класса Set.

Меню калькулятора реализовано одноименной функцией. Введенное значение запускает соответствующую команду, затем возвращает меню.

Функция show\_sets выводит все сохраненные множества в виде:

[0](x1, x2, x3, …)

[1](x4, x5, x6, …)

…

Функция enter\_set считывает числа в строке и записывает их в множество, которое сохраняет в векторе.

Функция delete\_sets считывает числа в строке и к каждому числу удаляет соответствующее множество.

Функция random\_set создает, отображает и сохраняет в векторе случайное число, согласно заданным параметрам.

Функция change\_parameters создает подменю для изменения параметров генерации случайного множества.

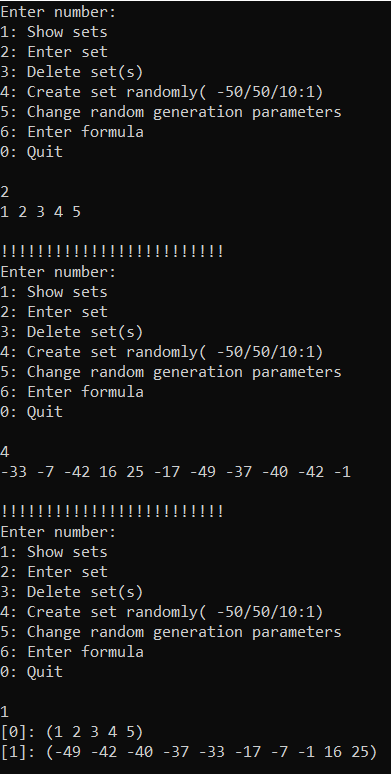
Функция enter\_formula получает строку с формулой, проверяет её функцией check, решает ее с помощью функции formula, получая множество. Это множество пересекает с универсумом для соблюдения диапазона, выводит результат и сохраняет его в векторе.

Функция check проверяет строку с формулой на ошибки: недопустимы незакрытые скобки и скобки внутри [сохраненных] и {написанных} множеств, лишние и незакрытые скобки для (приоритета операции), отсутствие операций между выражениями.

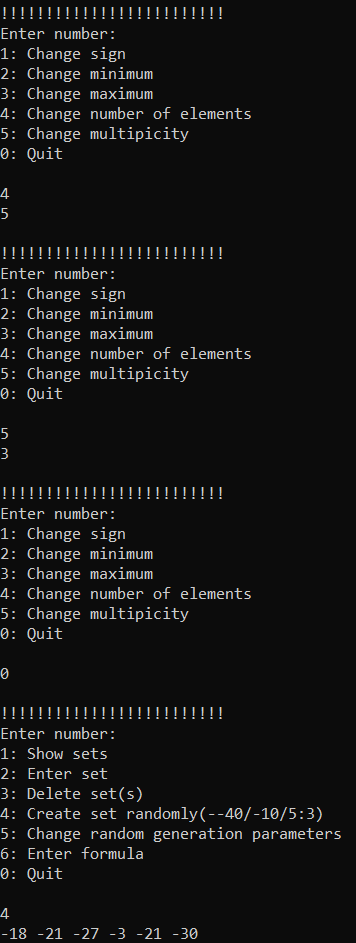
Функция formula решает формулу в строке. Алгоритм ищет в строке индекс последней операции (AND, OR, DIF, XOR), игнорируя все, что внутри скобок.  
 Если такая операция не найдена, проверяет операцию NOT и исполняет ее, иначе убирает внешние скобки, если есть. Если нет операций и внешних скобок, считывается и возвращается множество, [сохраненное по индексу] или {написанное вручную}.  
 Иначе операция существует. Строка разделяется на две строки: до и после индекса. Формулы возвращают эти строки как множества, с которыми выполняется операция. Возвращается множество как результат.

**Тесты**

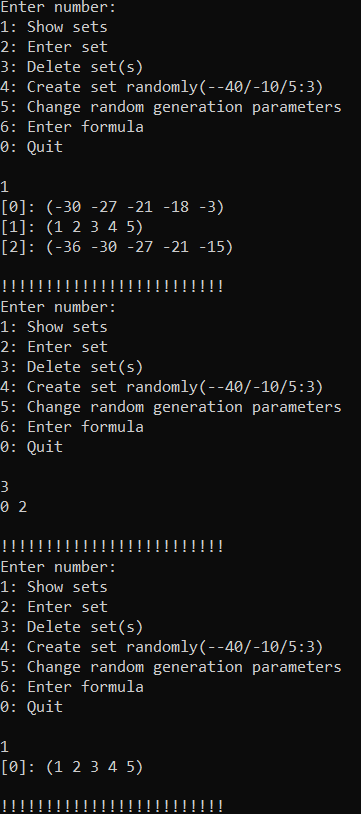
Ввод множества и генерация множества



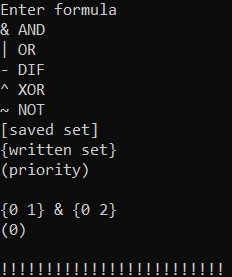
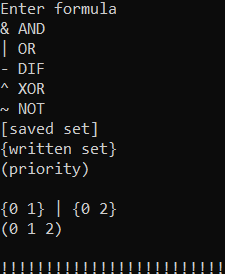
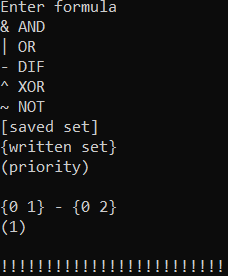
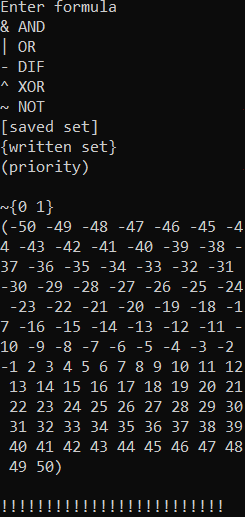
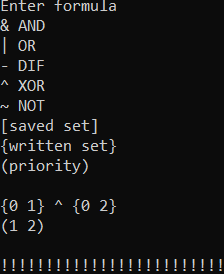
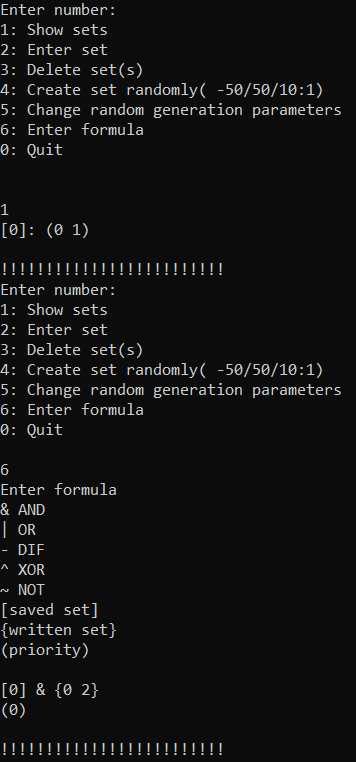
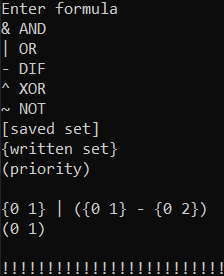
Изменение параметров: знак -, диапазон (-40;-10), количество 5, кратность 3.



Отображение сохраненных множеств, удаление множеств



Результаты формул:

**Код на C++**

#include <iostream>

#include <vector>

#include <set>

#include <string>

using namespace std;

class Set // Класс, содержащий множество, поддерживающий операции & | - ^ ~

{

private:

set<int> elements;

public:

Set() = default;

Set(const initializer\_list<int>& init) : elements(init) {}

void display() const // Отобразить множество в скобках. Если пустое - вывести текст

{

cout << "(";

for (const auto& elem : elements)

{

cout << elem << " ";

}

if (this->elements.empty())

{

cout << "Empty set ";

}

cout << "\b)\n";

}

void insert(int n) // Вставить элемент в множество

{

elements.insert(n);

}

int size() // Вернуть размер

{

return elements.size();

}

Set operator&(const Set& other) const // Оператор AND

{

Set result;

for (int i : elements)

{

if (other.elements.find(i) != other.elements.end())

{

result.insert(i);

}

}

return result;

}

Set operator|(const Set& other) const // Оператор OR

{

Set result = \*this;

for (int i : other.elements)

{

result.insert(i);

}

return result;

}

Set operator-(const Set& other) const // Оператор разности DIF

{

Set result;

for (int i : elements)

{

if (other.elements.find(i) == other.elements.end())

{

result.insert(i);

}

}

return result;

}

Set operator^(const Set& other) const // Оператор XOR

{

Set result = (\*this | other) - (\*this & other);

return result;

}

Set operator~() const // Оператор NOT

{

Set result;

for (int i = -50; i <= 50; i++)

{

if (elements.find(i) == elements.end())

{

result.insert(i);

}

}

return result;

}

};

void show\_sets(vector<Set> sets) // Вывести все сохраненные множества [0]: (x1 x2 x3...)

{ // [1]: (x4 x5 x6...)

int n = 0; // ...

for (Set str : sets)

{

int count = 0;

cout << "[" << n << "]: ";

str.display();

n++;

}

}

void enter\_set(vector<Set>& sets) // Считывает числа в строке и записывает их в множество

{

string str;

string sti;

string e = "-1234567890";

bool fl = false;

Set temp;

cin.ignore();

getline(cin, str);

str += ' ';

for (char c : str)

{

if (e.find(c) != e.npos)

{

sti += c;

fl = true;

}

else if (fl == true)

{

if (stoi(sti) <= 50 and stoi(sti) >= -50)

{

temp.insert(stoi(sti));

}

sti = "";

fl = false;

}

}

temp.display();

sets.push\_back(temp);

}

void delete\_sets(vector<Set>& sets) // Считывает числа в строке - к каждому числу удаляет соответствующее множество

{

string str;

string sti;

string e = "-1234567890";

bool fl = false;

int shift = 0;

cin.ignore();

getline(std::cin, str);

str += ' ';

for (char c : str)

{

if (e.find(c) != e.npos)

{

sti += c;

fl = true;

}

else if (fl == true)

{

int index = stoi(sti) - shift;

if (index >=0 and index < sets.size())

{

sets.erase(sets.begin() + index);

shift++;

}

sti = "";

fl = false;

}

}

}

void random\_set(vector<Set>& sets, char sign, int min, int max, int num, int mult) // Создает и сохраняет случайное число, согласно параметрам

{

Set temp;

if (sign == '-')

{

max = 0;

}

else if (sign == '+')

{

min = 0;

}

if (min % mult != 0)

{

min += mult - (min % mult);

}

if (max % mult != 0)

{

max -= max % mult;

}

while (temp.size() != num)

{

int n = rand() % (max - min + 1) + min;

if (n % mult != 0 and n + mult - (n % mult) <= max)

{

n += mult - (n % mult);

}

else

{

n -= n % mult;

}

temp.insert(n);

cout << n << " ";

}

sets.push\_back(temp);

cout << endl;

}

void change\_parameters(char& sign, int& min, int& max, int& num, int& mult) // Меню для смены параметров

{

int n;

cout << "Enter number:\n";

cout << "1: Change sign\n";

cout << "2: Change minimum\n";

cout << "3: Change maximum\n";

cout << "4: Change number of elements\n";

cout << "5: Change multipicity\n";

cout << "0: Quit\n\n";

cin >> n;

switch (n)

{

case 0: return; break;

case 1:

{

int k;

cout << "Enter number:\n";

cout << "1: +\n";

cout << "2: -\n";

cout << "3: None\n";

cout << "0: Quit\n\n";

cin >> k;

switch (k)

{

case 0: break;

case 1: sign = '+'; break;

case 2: sign = '-'; break;

case 3: sign = ' '; break;

default: cout << "Wrong number";

}

break;

}

case 2:

{

int k;

cin >> k;

if (k >= -50)

{

min = k;

}

break;

}

case 3:

{

int k;

cin >> k;

if (k <= 50)

{

max = k;

}

break;

}

case 4:

{

int k;

cin >> k;

if (k >= 0)

{

num = k;

}

break;

}

case 5:

{

int k;

cin >> k;

if ((k >= 1) and

((min % mult == 0) or

(min + mult - (min % mult) <= max)))

{

mult = k;

}

break;

}

default: cout << "Wrong number";

}

if (min > max)

{

min = max;

}

int i;

int c = 0;

if (min % mult != 0)

{

i = min + mult - (min % mult);

}

else

{

i = min;

}

for (i; i <= max; i = i + mult)

{

c++;

if (c == num)

{

break;

}

}

num = c;

cout << endl << "!!!!!!!!!!!!!!!!!!!!!!!!!" << endl;

return change\_parameters(sign, min, max, num, mult);

}

bool check(string str) // Проверяет формулу на ошибки, возвращает true, если ошибка есть

{

// Недопустимы незакрытые скобки и скобки внутри [сохранненных] и {написанных} множеств

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

{

switch (str[i])

{

case '[':

{

if (i == str.size() - 1)

{

cout << "[ at the end\n";

return true;

}

for (int j = i + 1; j < str.size(); j++)

{

if (str[j] == ']')

{

i = j;

break;

}

else if (str[j] == '(' or

str[j] == ')' or

str[j] == '{' or

str[j] == '}' or

str[j] == '[' or

j == str.size() - 1)

{

cout << "[inside]\n";

return true;

}

}

break;

}

case '{':

{

if (i == str.size() - 1)

{

cout << "{ at the end\n";

return true;

}

for (int j = i + 1; j < str.size(); j++)

{

if (str[j] == '}')

{

i = j;

break;

}

else if (str[j] == '(' or

str[j] == ')' or

str[j] == '[' or

str[j] == ']' or

str[j] == '{' or

j == str.size() - 1)

{

cout << "{inside}\n";//

return true;

}

}

break;

}

}

}

// Недопустимы лишние и незакрытые скобки для (приоритета операции)

int c = 0;

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

{

switch (str[i])

{

case '(': c++; break;

case ')': c--; break;

}

}

if (c != 0)

{

cout << "extra brackets\n";

return true;

}

// Недопустимо отсутствие операций между выражениями

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

{

if (str[i] == ')')

{

for (int j = i; j < str.size(); j++)

{

if (str[j] == '(')

{

cout << ") (\n";

return true;

}

if (str[j] == '&' or

str[j] == '|' or

str[j] == '-' or

str[j] == '^' or

j == str.size() - 1)

{

i = j;

break;

}

}

break;

}

}

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

{

if (str[i] == ']')

{

for (int j = i; j < str.size(); j++)

{

if (str[j] == '[')

{

cout << "] [\n";

return true;

}

if (str[j] == '&' or

str[j] == '|' or

str[j] == '-' or

str[j] == '^' or

j == str.size() - 1)

{

i = j;

break;

}

}

break;

}

}

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

{

if (str[i] == '}')

{

for (int j = i; j < str.size(); j++)

{

if (str[j] == '{')

{

cout << "} {\n";

return true;

}

if (str[j] == '&' or

str[j] == '|' or

str[j] == '-' or

str[j] == '^' or

j == str.size() - 1)

{

i = j;

break;

}

}

break;

}

}

return false;

}

Set formula(string str, vector<Set> sets) // Решает формулу

{

bool bracket\_flag = false;

int i;

// Поиск итератора последней операции (& | - ^) в строке с конца. Если замечена закрывающая скобка - проходит до открывающей

for (i = str.size() - 1; i >= 0; i--)

{

if (str[i] == ')')

{

bracket\_flag = true;

int c = 0;

for (int j = i; j >= 0; j--)

{

switch (str[j])

{

case '(': c++; break;

case ')': c--; break;

}

if (c == 0)

{

i = j;

break;

}

}

}

else if(str[i] == '}')

{

for (int j = i; j >= 0; j--)

{

if (str[j] == '{')

{

i = j;

break;

}

}

}

else if ( str[i] == ']')

{

for (int j = i; j >= 0; j--)

{

if (str[j] == '[')

{

i = j;

break;

}

}

}

else if (str[i] == '&' or

str[i] == '|' or

str[i] == '-' or

str[i] == '^')

{

break;

}

}

// Если операция (& | - ^) не найдена:

if (i == -1)

{

// Проверка на операцию ~, если есть, выполняет

i = 0;

while (str[i] != '(' and

str[i] != '{' and

str[i] != '[' and

i < str.size())

{

if (str[i] == '~')

{

string strNOT(str.begin() + i + 1, str.end());

Set tmpNOT = formula(strNOT, sets);

return ~tmpNOT;

}

i++;

}

// Проверка на наличие скобок (), если были - работает со строкой без внешних скобок

if (bracket\_flag == true)

{

string strB = "";

for (i = 0; i < str.size(); i++)

{

strB += str[i];

if (str[i] == '(')

{

int countB = 1;

strB.erase(strB.size() - 1);

for (int j = i + 1; j < str.size(); j++)

{

strB += str[j];

switch (str[j])

{

case '(': countB++; break;

case ')': countB--; break;

}

if (countB == 0)

{

strB.erase(strB.size() - 1);

i = j;

break;

}

}

}

}

return formula(strB, sets);

}

// Нет операций и скобок - работа с множествами

i = 0;

while (str[i] != '{' and

str[i] != '[' and

i < str.size())

{

i++;

}

// Если нет [ или { - возвращает пустое множество

if (i == str.size())

{

Set emptyset;

return emptyset;

}

switch (str[i])

{

case '{': // Составление множества из чисел в строке

{

string sti;

string e = "-1234567890";

bool fl = false;

Set temp;

for (char c : str)

{

if (e.find(c) != e.npos)

{

sti += c;

fl = true;

}

else if (fl == true)

{

temp.insert(stoi(sti));

sti = "";

fl = false;

}

}

return temp;

}

case '[': // Использование сохраненного множества по числу в строке как итератора

{

string sti;

string e = "-1234567890";

bool fl = false;

Set temp;

for (char c : str)

{

if (e.find(c) != e.npos)

{

sti += c;

fl = true;

}

else if (fl == true)

{

temp = sets[stoi(sti)];

break;

}

}

return temp;

}

}

}

// Итератор не -1 - операция существует, строка делится на две: до и после итератора.

// Формулы возвращают строки как множества, выполняется операция между ними.

// Результат - множество

string str1;

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

{

str1 += str[j];

}

str += ' ';

string str2;

for (int j = i + 1; j < str.size(); j++)

{

str2 += str[j];

}

Set temp1 = formula(str1, sets);

Set temp2 = formula(str2, sets);

switch (str[i])

{

case '&': return temp1 & temp2;

case '|': return temp1 | temp2;

case '-': return temp1 - temp2;

case '^': return temp1 ^ temp2;

}

}

void enter\_formula(vector<Set>& sets) // Получает строку с формулой, проверяет её, выполняет формулу, результат как множество пересекает с универсумом, выводит

{

string str;

cout << "Enter formula\n";

cout << "& AND\n";

cout << "| OR\n";

cout << "- DIF\n";

cout << "^ XOR\n";

cout << "~ NOT\n";

cout << "[saved set]\n";

cout << "{written set}\n";

cout << "(priority)\n\n";

cin.ignore();

getline(cin, str);

if (check(str))

{

cout << "\nincorrect formula\n\n";

enter\_formula(sets);

}

else

{

Set U;

for (int i = -50; i <= 50; i++)

{

U.insert(i);

}

Set result = formula(str, sets);

result.display();

sets.push\_back(result);

}

}

void menu(vector<Set> sets, char sign, int min, int max, int num, int mult) // Меню операций

{

srand(time(0));

int n;

cout << "Enter number:\n";

cout << "1: Show sets\n";

cout << "2: Enter set\n";

cout << "3: Delete set(s)\n";

cout << "4: Create set randomly(" << sign << min << '/'<< max << '/' << num << ':' << mult << ")\n";

cout << "5: Change random generation parameters\n";

cout << "6: Enter formula\n";

cout << "0: Quit\n\n";

cin >> n;

switch (n)

{

case 0: return; break;

case 1: show\_sets(sets); break;

case 2: enter\_set(sets); break;

case 3: delete\_sets(sets); break;

case 4: random\_set(sets, sign, min, max, num, mult); break;

case 5: change\_parameters(sign, min, max, num, mult); break;

case 6: enter\_formula(sets); break;

default: cout << "Wrong number";

}

cout << endl << "!!!!!!!!!!!!!!!!!!!!!!!!!" << endl;

return menu(sets, sign, min, max, num, mult);

}

int main()

{

// Заданные параметры

char sign = ' ';

int min = -50;

int max = 50;

int num = 10;

int mult = 1;

vector<Set> sets;

//Вывод меню

menu(sets, sign, min, max, num, mult);

return 0;

}