**Міністерство освіти і науки України**

**Національний університет  
“Львівська політехніка”**

**Кафедра систем штучного інтелекту**

**Лабораторна робота № 2**

з дисципліни

«Дискретна математика»

**Виконав**:

студент групи КН-115  
Поставка Маркіян

**Викладач:**

Мельникова Н. І.

Львів – 2019р.

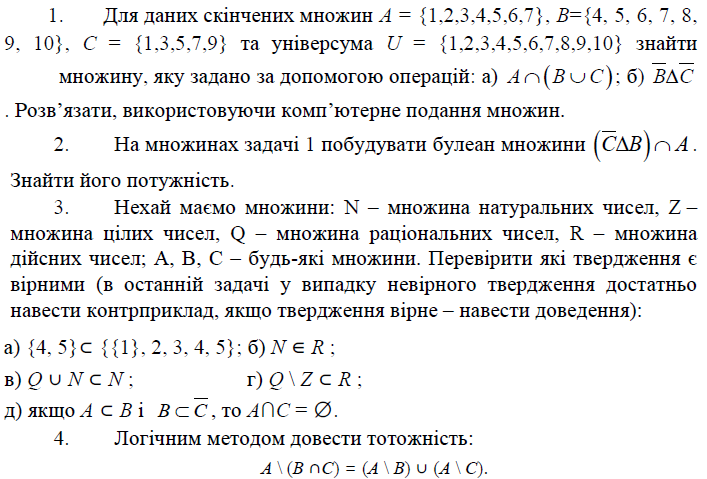
**Тема**: “Моделювання основних операцій для числових множин”

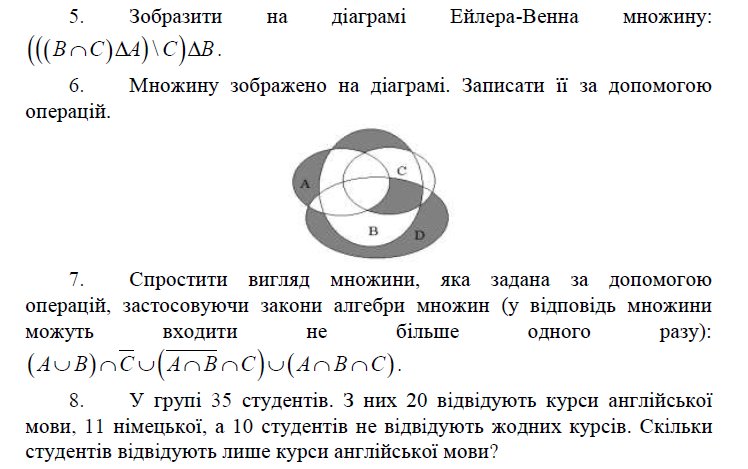
**Мета роботи:** Ознайомитись на практиці із основними теорії множин, навчитись будувати діаграми Ейлера-Венна операцій над множинами, використовувати закони алгебри множин, освоїти принцип включень-виключень для двох і трьох множин та комп’ютерне подання множин.

**Додаток 1:**

Варіант 11.

Постановка задачі:





Рішення:

1.

Комп’ютерно подамо множини:

A = 1111111000;

B = 0001111111;

C = 1010101010.

а) A ∩ (B ∪C)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| C | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| B ∪C | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| A | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| A ∩ (B ∪C) | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

б) not(B)∆not(C)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| not(B) | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| not(C) | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| not(B)∆not(C) | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

2.

F = (not(C)∆B) ∩A = 0100101000

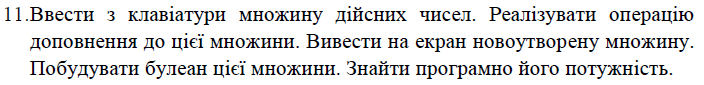
F = {2, 5, 7}

P(F) = {{},{2},{5},{7},{2,5},{2,7},{5,7},{2,5,7}} – Булеан множини F

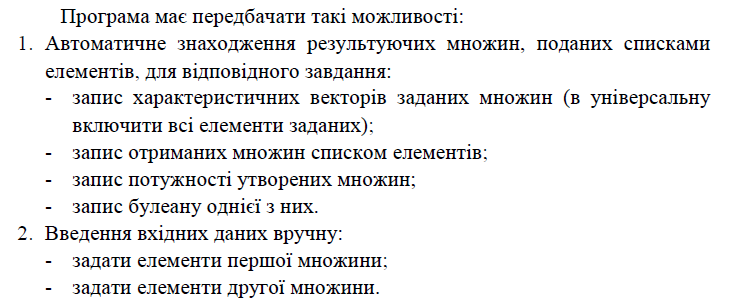
N = 2^3 = 8 – Потужність булеану

**Додаток 2:**

Постановка задачі:



Вимоги до програми:

Код програми:

#include "stdafx.h"

#include "conio.h"

#include <iostream>

#include <math.h>

#include <string.h>

using namespace std;

const int n = 5;

const long double hu = trunc(0.1\*10)/10;

struct set {

void inp\_set(double \*a) //////////

{

cout << "Enter set a with different values" << endl;

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

{

cin >> a[i];

a[i] = trunc(a[i] \* 10) / 10;

}

}

void out\_set(double \*a, int n) //////////

{

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

{

cout << a[i] << "\t";

}

cout << endl;

}

void out\_set(bool \*a, int n) //////////

{

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

{

cout << a[i] << "\t";

}

cout << endl;

}

void sort\_set(double m[n]) //////////

{

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

{

for (int i = 0; i < n - 1; i++)

{

double temp = m[i];

if (m[i + 1] < m[i])

{

m[i] = m[i + 1];

m[i + 1] = temp;

}

}

}

}

int u\_range(double \*a)

{

int r = 0;

double x = a[0];

while (x <= a[n - 1]+hu/2)

{

x += hu;

r++;

}

return r;

}

void universe(double \*a, double \*u, int u\_r)

{

double k = a[0];

for (int i = 0; i < u\_r; i++, k += hu)

{

u[i] = k;

u[i] = round(u[i] \* 10) / 10;

if (u[i] == -0) u[i] = 0;

}

}

void mask(double \*a, double \*u, bool \*a\_mask, bool \*u\_mask, int u\_r)

{

int k = 0;

for (int i = 0; i < u\_r; i++)

{

u\_mask[i] = 1;

if (a[k] == u[i])

{

a\_mask[i] = 1;

k++;

}

else a\_mask[i] = 0;

}

}

void complement(bool \*a\_mask, bool \*u\_mask, bool \*a\_mask\_comp, int u\_r)

{

for (int i = 0; i < u\_r; i++)

{

a\_mask\_comp[i] = u\_mask[i] - a\_mask[i];

}

}

void boolean(double \*a, int n) // Принцип: будується ніби табл істиності для всіх варіацій значень

// тоді ми робимо проходи по табличці і якшо є 1 то виводимо ці значення якшо 0 то ніякі не виводимо

// & - побітове порівнння а не порівняння значень напри 3(11) & 1(01) = true

// f - побітове представлення індексів

// i - це бітова маска кожного рядка в таблиці істиності яка і є булеаном

{

int nos = 1 << n; // Кількість підмножин

int f = 0;

for (int i = 0; i < nos; i++) //Перебір бітових масок(значення числа по таблиці істиності)

{

cout << ("{ ");

for (int j = 0; j < n; j++) // Перебір бітів(значення в таблиці істиності)

{

f = 1 << j; // Бітова маска(побітове представлення) індексів, а індекси в свою чергу представляють елементи масиву

if (i & f) // Якщо в значення і (по таблиці істиності) є таке двійкове f що вони мають спільні елементи(якщо двійкове f входить до i)( то виводимо ті спільні елементи)

cout << a[j] << " "; // То виводимо jй елемент множини

}

cout << "}\n";

}

}

};

int main()

{

set s;

double a[n] = {0};

s.inp\_set(a);

cout << "Set a:" << endl;

s.out\_set(a,n);

cout << endl;

cout << "Set a sorted:" << endl;

s.sort\_set(a);

s.out\_set(a,n);

cout << endl;

int u\_r = s.u\_range(a);

double \*u = new double[u\_r];

bool \*a\_mask = new bool[u\_r];

bool \*u\_mask = new bool[u\_r];

bool \*a\_mask\_compl = new bool[u\_r];

cout << "Set universe:" << endl; // Від найменшого а до найбільшого а - межі універсуму

s.universe(a, u, u\_r);

s.out\_set(u,u\_r);

cout << endl;

cout << "Mask a:" << endl;

s.mask(a, u, a\_mask, u\_mask, u\_r);

s.out\_set(a\_mask, u\_r);

cout << endl;

cout << "Mask a complemention:" << endl;

s.complement(a\_mask, u\_mask, a\_mask\_compl, u\_r);

s.out\_set(a\_mask\_compl, u\_r);

cout << endl;

///////////////////////

cout << "Boolean of a:" << endl;

s.boolean(a, n);

cout << endl;

delete[] a\_mask\_compl;

delete[] a\_mask;

delete[] u\_mask;

delete[] u;

\_getch();

return 0;

}

Результат роботи програми:



**Висновок:** Яознайомився на практиці із основними теорії множин, навчився будувати діаграми Ейлера-Венна операцій над множинами, використовувати закони алгебри множин, освоїв принцип включень-виключень для двох і трьох множин та комп’ютерне подання множин.