Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій



Звіт

Про виконання лабораторної роботи №2

Множини

з дисципліни «Дискретна математика»

студента IT-11 Лісничука Арсена

2021

Варіант 13

Мета:

Вивчення множин, операцій над множинами, комп’ютерного подання множин.

Теорія:

Множину можна задати, навівши її елементи у фігурних дужках. Наприклад, множина А = {a, e , i, o, u} містить елементи a, e , i, o, u й лише ці елементи.

Для часто використовуваних множин є спеціальні позначення:

Ø – порожня множина, що не містить жодного елемента;

Z – множина цілих чисел, Z = {…, -2, -1, 0, 1, 2, …};

R – множина дійсних чисел;

N – множина натуральних чисел, N = {1, 2, …};

– множина натуральних чисел із числом 0, = {0, 1, 2, …}

Завдання 1:

Умова:

Задана універсальна множина U={1,2,3,4,…,25} та три її підмножини A, B, C (табл. 2). Знайти F (табл. 3)

U {1, 2, …, 24,25}

A {1,2,3,4,13,14,16,17,21}

B {1,3,4,5,6,13,14,15,20,21}

C {1,9,10,11,12,13,18,19,24}

(А∪С) \ С ∪ В) \

Текст програми:

#include <iostream>

#include <algorithm>

#include <cmath>

using namespace std;

#define GET\_ARRAY\_LENGHT \_countof

int UniversalPluralSize;

int\* UniversalPlural;

// input

int A[] = { 1, 2, 3, 4, 13, 14, 16, 17, 21 };

int B[] = { 1, 3, 4, 5, 6, 13, 14, 15, 20, 21 };

int C[] = { 1, 9, 10, 11, 12, 13, 18, 19, 24 };

const int SizeA = GET\_ARRAY\_LENGHT(A);

const int SizeB = GET\_ARRAY\_LENGHT(B);

const int SizeC = GET\_ARRAY\_LENGHT(C);

// from = -100, to = 200 = {-100, -99, -98 ... 198, 199, 200 }

void SetUniversalPlural(const int from, const int to)

{

UniversalPluralSize = to - from + 1;

UniversalPlural = new int[UniversalPluralSize];

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

{

UniversalPlural[i] = from + i;

}

}

// Print array

void PrintPluar(const int\* p, const int size, const char\* addString = nullptr)

{

if (addString != nullptr)

{

cout << "[" << addString << "] ";

}

cout << "{";

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

{

cout << p[i];

if (i != size - 1)

{

cout << ", ";

}

}

cout << "}" << endl;

}

// A + B

// {0, 1, 2, 3} + {5, 8, 0, 1, 2} = {0, 0, 1, 1, 2, 2, 3, 5, 8}

int\* Add(const int\* a, const int size\_a, const int\* b, const int size\_b, int& size, bool isSort = true)

{

size = size\_a + size\_b; // 4 + 5

int\* res = new int[size]; // {0, 0, 0, 0, 0, 0, 0, 0, 0}

int total;

int index\_a = 0;

for (index\_a = 0; index\_a < size\_a; index\_a++)

{

res[index\_a] = a[index\_a]; // {0, 1, 2, 3, 0, 0, 0, 0, 0}

}

total = index\_a;

for (int i = 0; i < size\_b; i++, total++)

{

res[total] = b[i]; // {0, 1, 2, 3, 5, 8, 0, 1, 2}

}

if (isSort)

sort(res, res + size); // {0, 0, 1, 1, 2, 2, 3, 5, 8}

return res;

}

// A \ B

// {0, 4, -1, 5} \ {2, 3, 5, -2, 0} = {-1, 4}

int\* Divide(const int\* a, const int size\_a, const int\* b, const int size\_b, int& size, bool isSort = true)

{

int index = 0;

int\* tmp = new int[size\_a]; // {0, 0, 0, 0}

for (int i = 0; i < size\_a; i++)

{

for (int j = 0; j < size\_b; j++)

{

if (a[i] == b[j])

{

tmp[index] = i; // {0, 3} масив з індексами елементій, які треба видалити

index++;

}

}

}

size = size\_a - index; // 4 - 2

int\* res = new int[size]; // {0, 0}

for (int i = 0, i1 = 0; i < size\_a; i++)

{

if (i == tmp[i1])

{

i1++;

}

else

{

res[i - i1] = a[i]; // {4, -1}

}

}

delete[] tmp;

tmp = nullptr;

if (isSort)

sort(res, res + size); // {-1, 4}

return res;

}

// A u B

// {0, -1, -3, 5, 4, 9} u {3, 2, 5, 6, 7} = {-3, -1, 0, 2, 3, 4, 5, 6, 7, 9 }

int\* Unification(const int\* a, const int size\_a, const int\* b, const int size\_b, int& size, bool isSort = true)

{

int\* tmp = new int[size\_b]; // {0, 0, 0, 0, 0}

int index = 0; // змінна для кількості елементій які повторюються

for (int i = 0; i < size\_a; i++)

{

for (int j = 0; j < size\_b; j++)

{

if (a[i] == b[j])

{

tmp[index] = a[i]; // {5}

index++;

}

}

}

int s;

int\* tmp2 = Divide(b, size\_b, tmp, index, s); // {3, 2, 5, 6, 7} \ {5} = {3, 2, 6, 7}

int\* res = Add(tmp2, s, a, size\_a, size); // {3, 2, 6, 7, 0, -1, -3, 4, 9}

if (isSort)

sort(res, res + size); // {-3, -1, 0, 2, 3, 4, 5, 6, 7, 9 }

return res;

}

// !A

// {1, 3, 5, 7, 14} = { 2, 4, 6, 8, 9, 10, 11, 12, 13, 15, ... , 25 }

int\* Not(const int\* a, const int size\_a, int& size, bool isSort = true)

{

// !A = U \ A

return Divide(UniversalPlural, UniversalPluralSize, a, size\_a, size); // {1, 2, ... , 24, 25} \ {1, 3, 5, 7, 14} = {}

}

int main()

{

// U[25]

SetUniversalPlural(1, 25);

PrintPluar(UniversalPlural, UniversalPluralSize, "U");

PrintPluar(A, SizeA, "A");

PrintPluar(B, SizeB, "B");

PrintPluar(C, SizeC, "C");

cout << endl;

// A u C

int size\_p1;

int\* p1 = Unification(A, SizeA, C, SizeC, size\_p1);

PrintPluar(p1, size\_p1, "A u C");

// !C

int size\_p2;

int\* p2 = Not(C, SizeC, size\_p2);

PrintPluar(p2, size\_p2, "!C");

// (A u C) \ C

int size\_p3;

int\* p3 = Divide(p1, size\_p1, C, SizeC, size\_p3);

PrintPluar(p3, size\_p3, "(A u C) \\ C");

// ((A u C) \ C) u B

int size\_p4;

int\* p4 = Unification(p3, size\_p3, B, SizeB, size\_p4);

PrintPluar(p4, size\_p4, "((A u C) \\ C) u B");

// (((A u C) \ C) u B) \ (!C)

int size\_p5;

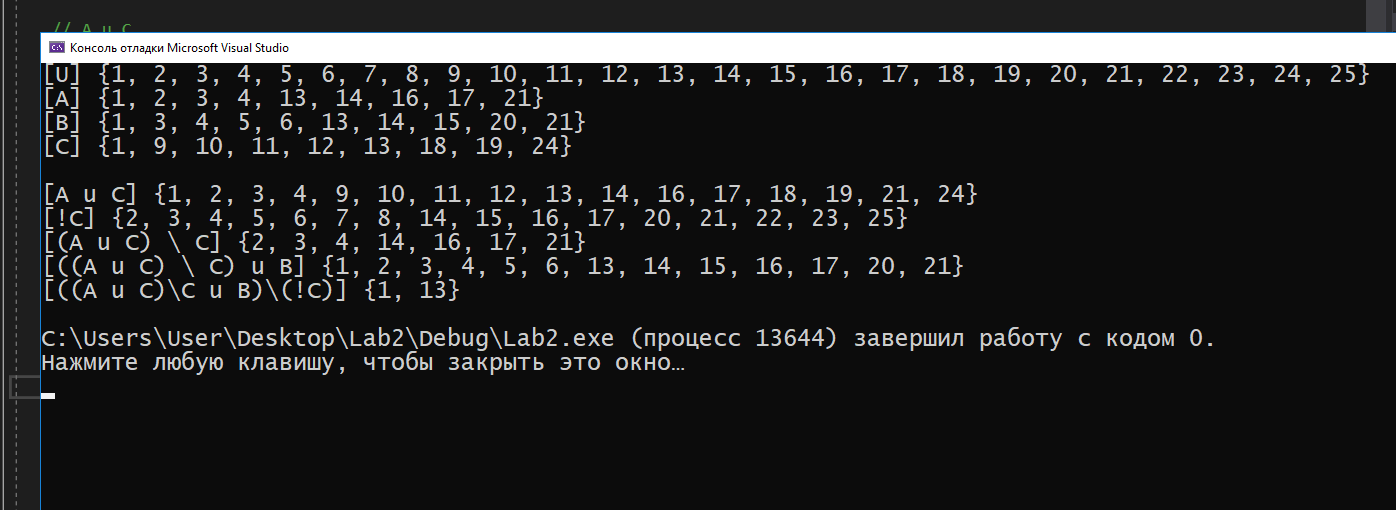
int\* p5 = Divide(p4, size\_p4, p2, size\_p2, size\_p5);

PrintPluar(p5, size\_p5, "((A u C)\\C u B)\\(!C)");

return 0;

}

Результати виконання програми:

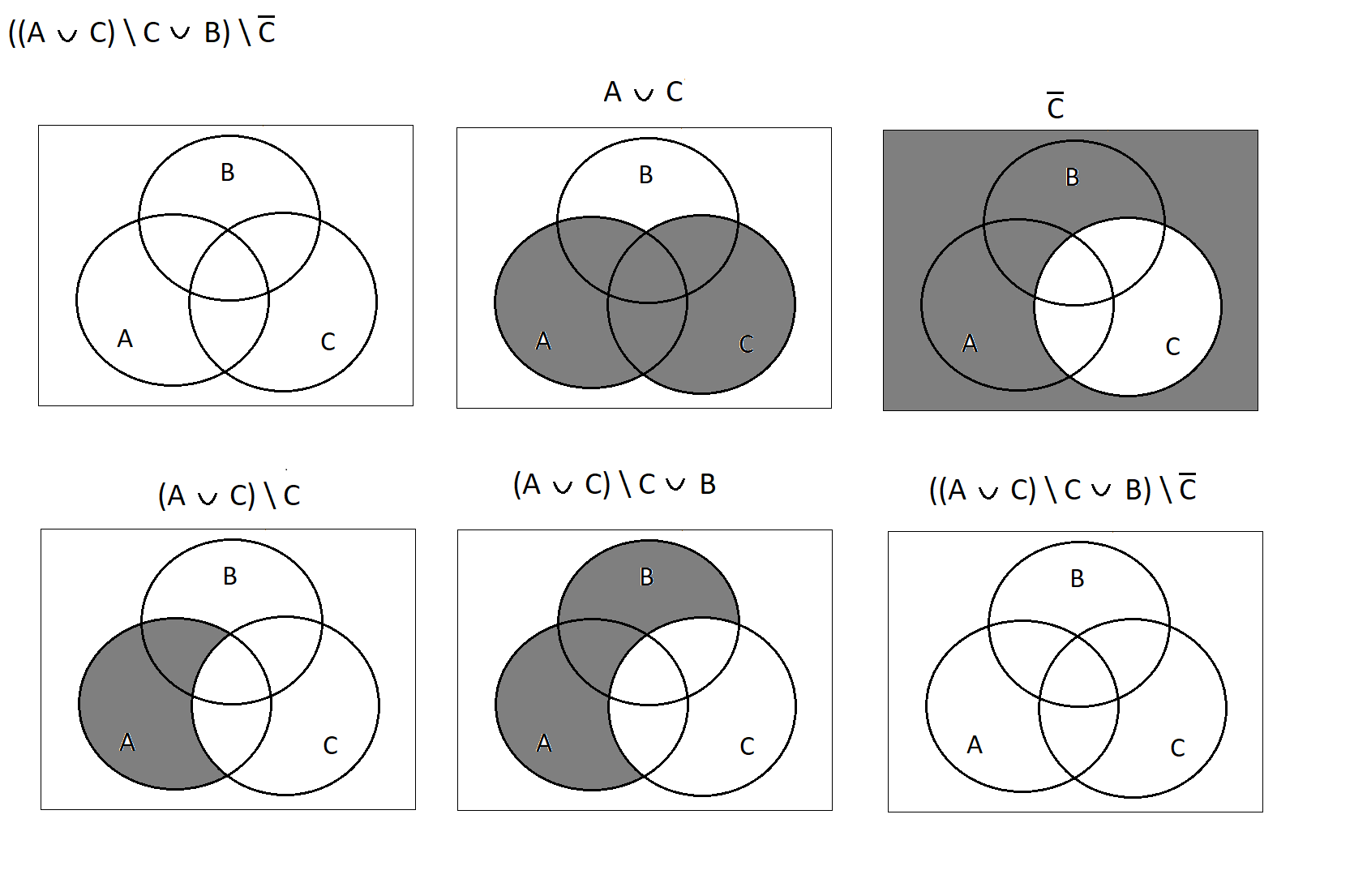


Завдання 2:

Умова:

Для формул, наведених у табл. 3, побудувати діаграми Венна за допомогою будь-якого графічного редактора.

Розвязок:



Завдання 3:

Задано множини А та В (табл. 4). Знайти декартовий добуток A× B та 2 A .

Текст програми:

#include <iostream>

#include <algorithm>

#include <cmath>

using namespace std;

#define GET\_ARRAY\_LENGHT \_countof

int UniversalPluralSize;

int\* UniversalPlural;

// input

int A[] = { 1, 8, 9 };

const int SizeA = GET\_ARRAY\_LENGHT(A);

int B[] = { 6, 8 };

const int SizeB = GET\_ARRAY\_LENGHT(B);

void SetUniversalPlural(const int from, const int to)

{

UniversalPluralSize = to - from + 1;

UniversalPlural = new int[UniversalPluralSize];

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

{

UniversalPlural[i] = from + i;

}

}

// Print array

void PrintPluar(const int\* p, const int size, const char\* addString = nullptr)

{

if (addString != nullptr)

{

cout << "[" << addString << "] ";

}

cout << "{";

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

{

cout << p[i];

if (i != size - 1)

{

cout << ", ";

}

}

cout << "}" << endl;

}

// A x B

// {0, 3, 5} x {3, 7} = { (0, 3), {0, 7}, (3, 3), (3, 7), (5, 3), (5, 7) }

int\*\*\* Multiply(const int \*a, const int size\_a, const int \*b, const int size\_b, int size[], bool isPrinted = false, const char\* addString = nullptr)

{

if (size != nullptr)

{

size[0] = size\_a;

size[1] = size\_b;

size[2] = 2;

}

int \*\*\*res = new int\*\*[size\_a];

for (int i = 0; i < size\_a; i++)

{

res[i] = new int\*[size\_b];

for (int j = 0; j < size\_b; j++)

{

int \* consist = new int[2]{ a[i], b[j] };

res[i][j] = consist;

}

}

if (isPrinted) {

if (addString != nullptr)

{

cout << "[" << addString << "] ";

}

cout << "{";

for (int i = 0; i < size\_a; i++)

{

for (int j = 0; j < size\_b; j++)

{

cout << "(" << res[i][j][0] << ", " << res[i][j][1] << ") ";

}

if (i == size\_a - 1)

{

cout << "}" << endl;

}

}

}

return res;

}

// A^2

// {0, 3, 5}^2 = {(0, 0), (0, 3), (0, 5), ()}

int\*\*\* Pow2(const int\* a, const int size\_a, int size[], bool isPrinted = false, const char\* addString = nullptr)

{

if (size != nullptr)

{

size[0] = size\_a;

size[1] = 2;

}

int\*\*\* res = new int\*\* [size\_a];

for (int i = 0; i < size\_a; i++)

{

res[i] = new int\* [size\_a];

for (int j = 0; j < size\_a; j++)

{

int\* consist = new int[2]{ a[i], a[j] };

res[i][j] = consist;

}

}

if (isPrinted) {

if (addString != nullptr)

{

cout << "[" << addString << "] ";

}

cout << "{";

for (int i = 0; i < size\_a; i++)

{

for (int j = 0; j < size\_a; j++)

{

cout << "(" << res[i][j][0] << ", " << res[i][j][1] << ") ";

}

if (i == size\_a - 1)

{

cout << "}" << endl;

}

}

}

return res;

}

// AxB, A^2

int main()

{

// U[25]

SetUniversalPlural(1, 25);

PrintPluar(A, SizeA, "A");

PrintPluar(B, SizeB, "B");

cout << endl;

int \*size = nullptr; // тому що ми не користуємось size

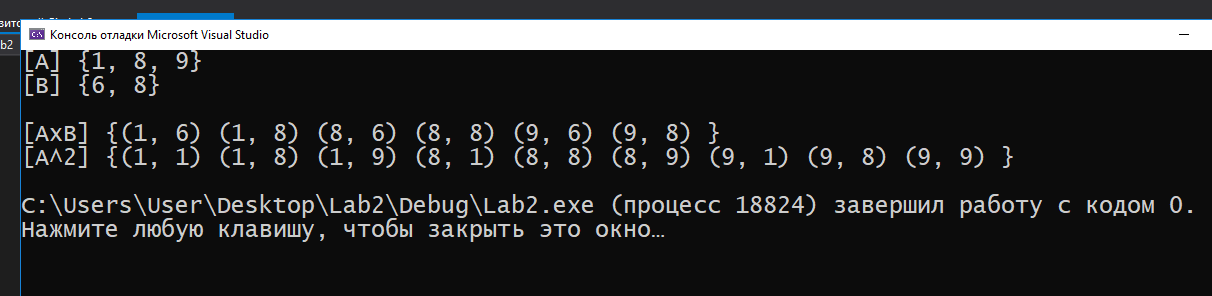
int\*\*\* pluar = Multiply(A, SizeA, B, SizeB, size, true, "AxB");

pluar = Pow2(A, SizeA, size, true, "A^2");

return 0;

}

Результати виконання програми:



Висновок: В цій лабораторній роботі я навчився працювати із множинами.