Міністерство освіти і науки України

Національний університет
“Львівська політехніка”

Кафедра систем штучного інтелекту

Лабораторна робота № 1

з дисципліни

«Дискретна математика»

Виконав:

студент групи КН-115
Поставка Маркіян

Викладач:
Мельникова Н. І.

Львів – 2019р.

Тема: “Моделювання основних логічних операцій”

Мета роботи: Ознайомитись на практиці із основними поняттями математичної логіки, навчитися будувати складні висловлювання за допомогою логічних операцій та знаходити їхні істинностні значення таблицями істинності, використовувати закони алгебри логіки, освоїти методи доведень.

Теоретичні відомості:

1.1. Основні поняття математичної логіки. Логічні операції

Просте висловлювання (атомарна формула, атом) – це розповідне речення, про яке можна сказати, що воно *істинне* (Т або 1) або *хибне* (F або 0), але не те й інше водночас.

Складне висловлювання – це висловлювання, побудоване з простих за допомогою *логічних операцій* (*логічних зв'язок*). Найчастіше вживаними операціями є 6: **заперечення** (читають «**не**», позначають $\neg$, $\bar{\ }$), **кон'юнкція** (читають «**і**», позначають $\wedge$), **диз'юнкція** (читають «**або**», позначають $\vee$), **імплікація** (читають «**якщо ..., то**», позначають $\Rightarrow$), **альтернативне «або»** (читають «**додавання за модулем 2**», позначають $\oplus$), **еквівалентність** (читають «**тоді і лише тоді**», позначають $\Leftrightarrow$).

Запереченням довільного висловлювання P називають таке висловлювання $\neg P$, істиносне значення якого строго протилежне значенню P. **Кон'юнкцією** або **логічним множенням** двох висловлювань P та Q називають складне висловлювання $P \wedge Q$, яке набуває істинного значення тільки в тому випадку, коли істинні *обидві* його складові. **Диз'юнкцією** або **логічним додаванням** двох висловлювань P та Q називають складне висловлювання $P \vee Q$, яке набуває істинного значення в тому випадку, коли істинною є *хоча б одна* його складова. **Імплікацією** двох висловлювань P та Q називають умовне висловлювання «**якщо *P*, то *Q***» ($P \Rightarrow Q$), яке прийнято вважати хибним тільки в тому випадку, коли *передумова* (*антецедент*) ***P*** *істинна*, а *висновок* (*консеквент*) ***Q*** *хибний*. У будь-якому іншому випадку його вважають істинним. **Альтернативним “або”** двох висловлювань P та Q називають складне висловлювання $P \oplus Q$, яке набуває істинного значення тоді і лише тоді, коли P та Q мають *різні* логічні значення, і є хибним в протилежному випадку. **Еквіваленцією** двох висловлювань P та Q називають складне висловлювання $P \Leftrightarrow Q$, яке

набуває істинного значення тоді і лише тоді, коли P та Q мають *однакові* логічні значення, і є хибним в протилежному випадку, тобто *логічно еквівалентні* складні висловлювання – це висловлювання, які набувають однакових значень істинності *на будь-якому* наборі істиносних значень своїх складових.

Тавтологія – формула, що виконується у всіх інтерпретаціях (тотожно істинна формула). **Протиріччя** – формула, що не виконується у жодній інтерпретації (тотожно хибна формула). Формулу називають **нейтральною**, якщо вона не є ні тавтологією, ні протиріччям (для неї існує принаймні один набір пропозиційних змінних, на якому вона приймає значення T, і принаймні один набір, на якому вона приймає значення F). **Виконана формула** – це формула, що не є протиріччям (інакше кажучи, вона принаймні на одному наборі пропозиційних змінних набуває значення T).

1.2. Закони логіки висловлювань

A	B
Закони асоціативності	
$(P \vee Q) \vee R = P \vee (Q \vee R)$	$(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$
Закони комутативності	
$P \vee Q = Q \vee P$	$P \wedge Q = Q \wedge P$
Закони ідемпотентності	
$P \vee P = P$	$P \wedge P = P$
Закони дистрибутивності	
$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$	$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$
Закони доповнення	
закон виключення третього: $P \vee (\overline{P}) = \mathrm{T}$	закон протиріччя: $P \wedge (\overline{P}) = \mathrm{F}$
закон подвійного заперечення $\overline{\overline{P}} = P$	
Закони де Моргана	
$\overline{(P \vee Q)} = \overline{P} \wedge \overline{Q}$	$\overline{(P \wedge Q)} = \overline{P} \vee \overline{Q}$
Закони поглинання	
$(P \vee Q) \wedge P = P$	$(P \wedge Q) \vee P = P$
Співвідношення для сталих (закони тотожності та домінування)	

$P \vee \mathrm{T} = \mathrm{T}$	$P \wedge \mathrm{T} = P$(тот)
$P \vee \mathrm{F} = P$ (тот)	$P \wedge \mathrm{F} = \mathrm{F}$

1.3. Логіка першого ступеня. Предикати і квантори. Закони логіки першого ступеня

Предикат – це твердження, яке містить змінні та приймає значення істини чи фальші залежно від значень змінних; ***n*-місний предикат** – це предикат, що містить n змінних $x_1,\ldots, x_n$.

Квантор - логічний оператор, що перетворює будь-який предикат на предикат меншої місності, зв'язуючи деякі змінні початкового предиката. Вживаються два квантори: узагальнення (універсальний) (позначається $\forall$) та приналежності (екзистенціальний) (позначається $\exists$). Для будь-якого предиката $P(x)$ вирази $\forall x P(x)$ та $\exists x P(x)$ читаються як «всі x мають властивість $P(x)$» та «існує (бодай один) x, що має властивість $P(x)$» відповідно.

Перехід від $P(x)$ до $\forall x\ P(x)$ або $\exists x\ P(x)$ називають *зв'язуванням* предметної змінної x, а саму змінну x – *зв'язаною (заквантованою)*. Незв'язану змінну називають *вільною*. У виразах $\forall x\ P(x)$ або $\exists x\ P(x)$ предикат належить області дії відповідного квантора. Формулу, що не містить вільних змінних, називають *замкненою*.

Якщо $D=\{a_1,\ldots, a_n\}$ – скінченна предметна область змінної x у предикаті $P(x)$, то можна скористатись *логічними еквівалентностями*

$$\forall x\ P(x) = P(a_1)\wedge\ldots\wedge P(a_n) \text{ та } \exists x\ P(x) = P(a_1)\vee\ldots\vee P(a_n).$$

Обчислення предикатів, у якому квантори можуть зв'язувати лише предметні змінні, але не можуть зв'язувати предикати, називають обчисленням *першого порядку*. Обчислення, у яких квантори можуть зв'язувати не лише предметні змінні, але й предикати, функціональні символи чи інші множини об'єктів, називають обчисленнями *вищих порядків*.

Закони 1-2 дозволяють будувати заперечення формул з кванторами.
Закони 3-4 виражають закони дистрибутивності квантора загальності відносно диз'юнкції.$\exists$ відносно кон'юнкції та квантора існування $\forall$.
Закони 5-8 дозволяють виносити за межі дії квантора, що зв'язує змінну *x* та формулу, яка не містить *x.*
Закони 9-10 свідчать про комутативність однойменних кванторів. Тобто однойменні квантори можна міняти місцями, а різнойменні – ні.

Основні закони логіки першого ступеня (логіки предикатів):

1. $\neg(\forall x P(x)) = \exists x(\neg P(x))$, $\forall x P(x) = \neg\exists x(\neg P(x))$.
2. $\neg(\exists x P(x)) = \forall x(\neg P(x))$, $\exists x P(x) = \neg\forall x(\neg P(x))$.
3. $\forall x(P(x)\wedge Q(x)) = \forall x P(x)\wedge \forall x Q(x)$.
4. $\exists x(P(x)\vee Q(x)) = \exists x P(x)\vee \exists x Q(x)$

5. $\forall x(P(x) \wedge Q) = \forall x P(x) \wedge Q$.
6. $\forall x(P(x) \vee Q) = \forall x P(x) \vee Q$
7. $\exists x(P(x) \wedge Q) = \exists x P(x) \wedge Q$.
8. $\exists x(P(x) \vee Q) = \exists x P(x) \vee Q$.
9. $\forall x \forall y P(x, y) = \forall y \forall x P(x, y)$.
10. $\exists x \exists y P(x, y) = \exists y \exists x P(x, y)$.
11. $\forall x P(x) = \forall t P(t)$, $\exists x P(x) = \exists t P(t)$.
12. $\forall x P = P$, $\exists x P = P$.

Випереджена нормальна форма – формула, записана у вигляді $Q_1x_1Q_2x_2...Q_nx_nM$, де кожне Q_ix_i ($i = 1,2,...,n$) – це $\forall x_i$ або $\exists x_i$, а формула M не містить кванторів. Вираз $Q_1x_1...Q_nx_n$ називають префіксом, а M – *матрицею формули*, записаної у випереджeній нормальній формі.

1.4. Методи доведень

При доведенні теорем застосовують логічну аргументацію. Доведення в інформатиці – невід'ємна частина перевірки коректності алгоритмів. Необхідність доведення виникає, коли нам потрібно встановити істинність висловлювання виду ($P \Rightarrow Q$). Існує декілька стандартних типів доведень.

1. **Пряме міркування.** Допускаємо, що висловлювання P істинне і показуємо справедливість Q. Такий спосіб доведення виключає ситуацію, коли P істинне, а Q хибне, оскільки саме в цьому і лише в цьому випадку імплікація $P \Rightarrow Q$ набуває хибного значення (див. табл. 1.1).
2. **Обернене міркування.** Допускаємо, що висловлювання Q хибне і показуємо помилковість P. Фактично прямим способом перевіряємо істинність імплікації ($\neg Q \Rightarrow \neg P$), що згідно з прикладом 1.5 (правилом контрапозиції) логічно еквівалентне істинності вихідного твердження ($P \Rightarrow Q$).
3. Метод **«від протилежного»**. У допущенні, що висловлювання P істинне, а Q хибне, використовуючи аргументоване міркування, одержимо протиріччя. Цей спосіб заснований на тому, що імплікація ($P \Rightarrow Q$) набуває хибного значення лише тоді, коли P істинне, а Q хибне.

4. ***Принцип математичної індукції*** – це така **теорема**:

Теорема. Нехай P(n) – предикат, визначений для всіх натуральних n. Допустимо, що

1) *P(1) істинне і*
2) *$\forall\ k \geq 1$ імплікація (P(k) $\Rightarrow$ P(k+1)) є вірною.*

Тоді P(n) істинне при будь-якому натуральному n.

Означення 7.1. Випереджена нормальна форма – формула, записана у вигляді $Q_1x_1Q_2x_2...Q_nx_nM$, де кожне Q_ix_i (i = 1,2,...,n) –це $\forall x_i$ або $\exists x_i$, а формула *M* не містить кванторів. Вираз $Q_1x_1...Q_nx_n$ називають префіксом, а *M* – *матрицею формули*, записаної у випередженій нормальній формі**. Фактично, це запис формули з винесеними кванторами за дужки.**

Приклад 7.1. Наведемо приклади формул, записаних у випередженій нормальній формі.

1. $\forall x \forall y(P(x,y\wedge)Q(y))$.

2. $\forall x \exists y(P(x\vee)Q(y))$.

3. $\forall x \forall y \exists z(Q(x,y\wedge)R(z))$.

4. $\forall x \forall y \forall z \exists u(P(x,z\vee)P(y,z\vee)Q(x,y,u))$. ▲

Для того, щоб перевести формулу у випереджену нормальну форму, необхідно виконати наступні перетворення:

1. Використати правила усунення імплікації ($P\rightarrow Q= \overline{P} \vee Q$) та еквівалентності ($P$~$Q=(P\rightarrow Q(\wedge)Q\rightarrow P)$).
2. Застосувати закон подвійного заперечення ($\overline{\overline{P}} = P$) та закони де Моргана ($\overline{P \vee Q} = \overline{P} \wedge \overline{Q}, \overline{P \wedge Q} = \overline{P} \vee \overline{Q}$).
3. Застосувати закони: $\neg(\forall x P(x))= \exists x \overline{P}(x)$ та $\neg(\exists x P(x))= \forall x \overline{P}(x)$.
4. Застосувати закони логіки першого ступеня 3-8.
5. Винести квантори у префікс, для чого скористатись законами логіки першого ступеня 3-8.

16. Доведіть кожне з висловлювань методом математичної індукції:

а) $1+5+9+\ldots+(4n\text{-}3)=n(2n\text{-}1)$ для всіх натуральних чисел n;

б) $1^2+2^2+\ldots+n^2=n(n+1)(2n+1)/6$ для всіх натуральних чисел n;

Розв'язання.

а) Позначимо предикат $1+5+9+\ldots+(4n\text{-}3)=n(2n\text{-}1)$ через $P(n)$.

При $n=1$ ліва частина рівності містить лише 1. Права частина після підстановки $n=1$ теж буде рівною 1:

$$n(2n\text{-}1)=1(2\cdot1\text{-}1)=1.$$

Тому висловлювання $P(1)$ є істинним.

Допустимо, що $P(k)$ є істинним при деякому $k\geq1$:

$$1+5+9+\ldots+(4k\text{-}3)=k(2k\text{-}1).$$

Нам треба показати, що з такого допущення випливає істинність $P(k+1)$. Тому

$1+\ldots+(4k\text{-}3)+(4(k+1)\text{-}3)= k(2k\text{-}1)+(4k+1)=2k^2+3k+1$– ліва частина;

$(k+1)(2(k+1)\text{-}1)=(k+1)(2k+1)=2k^2+3k+1$– права частина.

Оскільки ліва і права частини виразу $P(k+1)$ співпадають, згідно принципу математичної індукції $P(n)$ є істинним для будь-якого $n\geq1$.

б) Тут $P(n)$ буде позначати предикат:

$$1^2+2^2+\ldots+n^2=n(n+1)(2n+1)/6.$$

Оскільки $1^2=1$ і $n(n+1)(2n+1)/6=1\cdot2\cdot3/6=1$ (при $n=1$), то висловлювання $P(1)$ є істинним.

Допустимо, що $P(k)$ є істинним при деякому $k\geq1$:

$$1^2+2^2+\ldots+k^2=k(k+1)(2k+1)/6,$$

і покажемо, що звідси випливає істинність $P(k+1)$:

$1^2+2^2+\ldots+k^2+(k+1)^2= k(k+1)(2k+1)/6+(k+1)^2=\frac{1}{6}(k+1)(k(2k+1)+6(k+1))=$

$=\frac{1}{6}(k+1)(2k^2+7k+6)$ – ліва частина;

$\frac{1}{6}(k+1)((k+1)+1)(2(k+1)+1)=\frac{1}{6}(k+1)(k+2)(2k+3)$ – права частина.

Оскільки ліва і права частини виразу $P(k+1)$ співпадають, за індукцією робимо висновок, що $P(n)$ є істинним для всіх натуральних чисел $n\geq1$.

Додаток 1:

Варіант 11.

Постановка задачі:

1. Формалізувати речення. Якщо Василь не прийде на іспит, то він не зможе отримати позитивну оцінку
2. Побудувати таблицю істинності для висловлювань: $\left(x \vee \overline{y}\right) \Rightarrow \left(\left(y \wedge \overline{z}\right) \Rightarrow \left(x \vee y\right)\right)$;
3. Побудовою таблиць істинності вияснити, чи висловлювання є тавтологією або протиріччям: $\left((p \to q) \wedge \overline{(\overline{q} \to r)}\right) \leftrightarrow (p \to \overline{r})$.
4. За означенням без побудови таблиць істинності та виконання еквівалентних перетворень перевірити, чи є тавтологією висловлювання: $((p \to q) \wedge (q \to r)) \to (p \to r)$
5. Довести, що формули еквівалентні: $(p \wedge q) \to (p \wedge r)$ та $(p \wedge r) \leftrightarrow (q \wedge r)$.

Рішення:

1.

P – прийти на іспит.

B – отримати позитивну оцінку.

X – Василь.

$$\neg P(x) \to \neg B(x)$$

2.

x	y	z	x∨¬y	y∧¬z	x∨y	y∧¬z→(x∨y)	x∨¬y→(y∧¬z→(x∨y))
0	0	0	1	0	0	1	1
0	0	1	1	0	0	1	1
0	1	0	0	1	1	1	1
0	1	1	0	0	1	1	1
1	0	0	1	0	1	1	1
1	0	1	1	0	1	1	1
1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1

3.

p	q	r	p→q	¬q→r	¬(¬q→r)	(p→q)∧¬(¬q→r)	p→¬r	(p→q)∧¬(¬q→)r≡(p→¬r)
0	0	0	1	0	1	1	1	1
0	0	1	1	1	0	0	1	0
0	1	0	1	1	0	0	1	0
0	1	1	1	1	0	0	1	0
1	0	0	0	0	1	0	1	0
1	0	1	0	1	0	0	0	1
1	1	0	1	1	0	0	1	0
1	1	1	1	1	0	0	0	1

Висловлювання не є тавтологією і не є протиріччям. Воно нормальної форми.

4.

Виконуємо завдання за допомогою методу відшукання контр прикладу.

Припускаємо, що формула не є тавтологією.

$$\big((p \to q) \land (q \to r)\big) \to (p \to r) = F;$$

Тоді

$$\big((p \to q) \land (q \to r)\big) = T;$$

$$(p \to r) = \ F;\ p = T; r = F;$$

Підставляємо значення p і r у висловлювання

$$\big((T \to q) \land (q \to F)\big) = T;$$

$$(T \to q) = T;\ (q \to F) = T;$$

Спробуємо підставити q=T або q=F і бачимо що при жодному із значень q вираз не буде правдою.

$$\big((T \to T) \land (T \to F)\big) \neq T;$$

$$\big((T \to F) \land (F \to F)\big) \neq T;$$

З цього робимо висновок, що дане висловлювання не буде протиріччям при будь-яких значеннях q.

Отже, воно є тавтологією, що і потрібно було довести.

5.

Для доведення складемо таблицю істинності.

p	q	r	p∧q	p∧r	p∧q→p∧r	q∧r	p∧r≡q∧r	p∧q→p∧r≡(p∧r≡q∧r)
0	0	0	0	0	1	0	1	1
0	0	1	0	0	1	0	1	1
0	1	0	0	0	1	0	1	1
0	1	1	0	0	1	1	0	0
1	0	0	0	0	1	0	1	1
1	0	1	0	1	1	0	0	0
1	1	0	1	0	0	0	1	0
1	1	1	1	1	1	1	1	1

З таблиці ми бачимо, що дане висловлювання не є тавтологією. Воно нормальної форми.

Додаток 2:

Постановка задачі:

Написати на будь-якій відомій студентові мові програмування програму для реалізації програмного визначення значень таблиці істинності логічних висловлювань при різних інтерпретаціях для наступної формули:

$$11.\ (x \vee \overline{y}) \Rightarrow ((y \wedge \overline{z}) \Rightarrow (x \vee y));$$

Вимоги до програми:

Програма має передбачати такі можливості:

1. Автоматичне знаходження істинносних значень (із записом таблиці істинності) складного висловлювання для всіх інтерпретацій простих висловлювань, які входять в нього, для відповідного завдання.
2. Введення вхідних даних вручну.
3. Перевірку на некоректне введення даних.

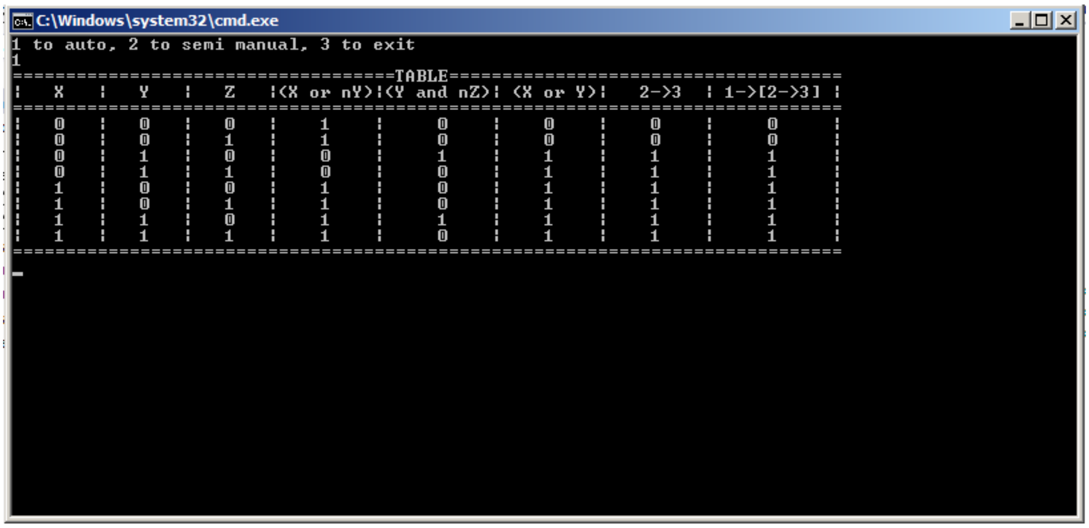
Код програми:

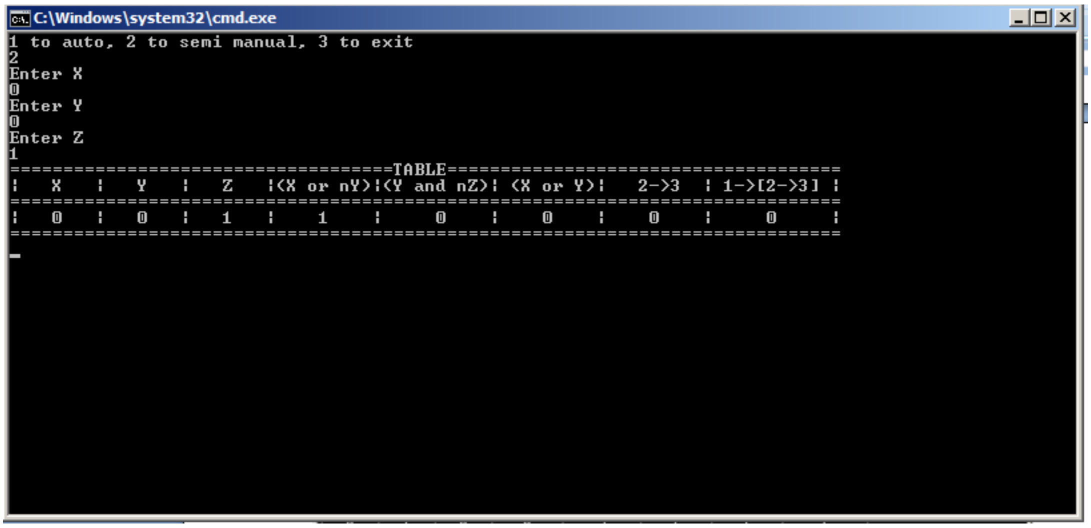
```
#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;

int input_var(int a, int code) // Метод для перевірки на коректність вхідних даних(
{
	int ck = 0; // змінна для перевірки

	while (ck != 2)
```

```cpp
	{
		a = 0;
		cout << "Enter " << (char)code << endl; // Виведення назви змінної типу x, y, z через інкрементацію ascii коду в виклику функції
		cin >> a;
		if (a == 1 || a == 0)
		{
			ck = 2;

		}
		else
		{
			cout << endl;
			cout << "Again ";
		}
	}
	return a;
}

int and(int a, int b)
{
	if (a*b == 0) return 0;
	if (a*b == 1) return 1;
}

int or (int a, int b)
{
	if (a + b == 0) return 0;
	if (a + b>0) return 1;
}

int not(int a)
{
	if (a == 0) return 1;
	else if (a == 1) return 0;
}

int impl(int a, int b)
{
	if (b == 0) return 0;
	else return 1;
}


void auto_(int rez)  // Метод для виводу всієї таблиці істинності
{
	int x = 0; int y = 0; int z = 0;
	cout << "====================================TABLE=====================================" << endl;
	cout << "|   X   |   Y   |   Z   |(X or nY)|(Y and nZ)| (X or Y)|   2->3   | 1->[2->3] |" << endl;
	cout << "==============================================================================" << endl;
	x = 0;
	while (x < 2)
	{
		y = 0;
		while (y < 2)
		{
			z = 0;
			while (z < 2)
			{
				int hp = impl(and (y, not(z)), or (x, y));   // Побічна допоміжна змінна
				rez = impl(or (x, not(y)), hp);
				cout << "|   " << x << "   |   " << y << "   |   " << z << "   |    "<< or(x, not(y)) << "    |     " << and (y, not(z)) << "    |    "<<or(x, y) << "    |    " << hp << "    |     " << rez << "     |" << endl;
				z++;
			}
			y++;
		}
		x++;
	}
	cout << "==============================================================================" << endl;
}

void s_manual(int rez)  // Метод для виведення значень тыльки при ваших вхыдних даних
{
	int code = 88;
	int x = 0; int y = 0; int z = 0;
	x = input_var(x, code++);
	y = input_var(y, code++);
	z = input_var(z, code++);
```

```
	cout << "==================================TABLE===================================" << endl;
	cout << "|    X    |    Y    |    Z    |(X or nY)|(Y and nZ)| (X or Y)|    2->3    | 1->[2->3] |" << endl;
	cout << "==========================================================================" << endl;
	int hp = impl(and (y, not(z)), or (x, y));   // Побічна допоміжна змінна
	rez = impl(or (x, not(y)), hp);
	cout << "|    " << x << "    |    " << y << "    |    " << z << "    |    "<< or(x, not(y)) << "    |    "
<< and (y, not(z)) << "    |    "<<or(x, y) << "    |    " << hp << "    |    " << rez << "    |" << endl;
	cout << "==========================================================================" << endl;
}

int main()
{
	system("MODE CON: COLS=100 LINES=30"); // Настройки консольки: довжина і висота
	int a[5]; // Array for input
	int n = 0;
	int rez = 0; //

	int ck = 0; // Menu start

	while (ck != 3)
	{
		ck = 0;
		system("@cls||clear"); // Очистка екрану
		cout << "1 to auto, 2 to semi manual, 3 to exit" << endl;
		cin >> ck;
		if (ck == 1) auto_(rez); // Виклик авто - методу

		else if (ck == 2) s_manual(rez); // Виклик ручного методу
		getch();
	}

	_getch(); // Затримка екрану
	return 0;
}
```

Результат роботи програми:





Висновок: Я ознайомився на практиці із основними поняттями математичної логіки, навчився будувати складні висловлювання за допомогою логічних операцій та знаходити їхні істинностні значення таблицями істинності, використовувати закони алгебри логіки, освоїв методи доведень.