Лабораторна робота № 1

Теоретичні відомості. Основи логіки висловлювань

Висловлюванням називають розповідне речення, про яке можна сказати, що воно є або істинне, або хибне, але не одне й інше одночасно. Висловлювання та їхні властивості вивчає *логіка висловлювань*.

Приклад.

1. Трава зелена (істинне твердження).
2. Небо зелене (хибне твердження).
3. Як Вас звати? (це запитання).
4. Розв'яжіть цю задачу (це наказове речення).

Значення „істина" чи „хибність", яких набуває висловлювання, називають його *значенням істинності*. Значення „істина" позначають буквою T (від англ. *truth*), а „хибність" – буквою F (від англ. *false*). Для позначення висловлювань будемо використовувати малі латинські літери. Символи, що використовуються для позначення висловлювань, називають *атомарними формулами* чи *атомами*.

Приклад.

1. *a*: Трава зелена.
2. *b*: Небо зелене.

Тут символи *a* і *b* – атомарні формули.

Багато речень утворюються об'єднанням одного чи декількох елементарних висловлювань. Таке отримане висловлювання називають *складним*. Складне висловлювання утворюють із елементарних висловлювань за допомогою *логічних операцій*. У логіці висловлювань використовують шість логічних операцій:

1. *заперечення* (логічне „ні", позначаення „$\neg$", комп'ютерна операція NOT);
2. *кон'юнкція* (логічне „і", позначення „$\wedge$", комп'ютерна операція AND);

3. *диз'юнкція* (логічне „або", позначення „$\vee$", комп'ютерна операція OR);
4. *імплікація* (читають „якщо..., то…", позначення „$\rightarrow$");
5. *еквівалентність* (читають „тоді й лише тоді…", позначення „$\sim$");
6. *виключаюче „або"* (позначення „$\oplus$", комп'ютерна операція XOR).

Приклад.

Розглянемо такі висловлювання: *a*: „Студент відвідує всі заняття", *b*: „Студент виконує всі завдання", *c*: „Студент отримає залік". Тоді речення „Якщо студент відвідує заняття і виконує всі завдання, то він отримає залік", можна записати складним висловлюванням $\left((a \wedge b) \rightarrow c\right)$.

У логіці висловлювань атом чи складне висловлювання називають *правильно побудованою формулою* або *формулою.* Вивчаючи формули, розглядають два аспекти – синтаксис і семантику.

Синтаксис – це сукупність правил, за якими будують та розпізнають правильні формули.

Приклад.

1. $(a \rightarrow b), (a \wedge b), (\neg a), (a \vee b)$ – формули.
2. $(a \rightarrow), (b \wedge), (a \neg), (\vee b)$ – не формули.

Зауваження.

Часто заперечення висловлювання *a* позначають також $\overline{a}$. Такий спосіб запису заперечення не потребує дужок. Якщо не виникає непорозумінь, то зовнішні дужки у формулах можна пропустити.

Операції кон'юнкції, диз'юнкції, імплакації, еквівалентності та виключаючого „або" відносяться до *бінарних*, тобто таких, що складаються з двох величин – операндів, над якими виконується певна дія. Символи таких операцій, як правило, записуються між операндами. Операція

заперечення є *унарною*, тобто такою, що застосовується до одного операнда (про форми запису виразів див. лабораторну роботу № 8).

Семантика – це сукупність правил, за якими формулам надають значення істинності. Семантику логічних операцій зручно задавати за допомогою таблиць, які містять значення істинності формул залежно від значень істинності їх атомів. Такі таблиці називають *таблицями істинності.* Семантику основних логічних операцій у формі таблиці істинності наведено в таблиці 1.1.

Таблиця 1.1

Таблиця істинності основних логічних операцій

a	b	$\bar{a}$ або $\neg a$	$a \wedge b$	$a \vee b$	$a \to b$	$a \sim b$	$a \oplus b$
T	T	F	T	T	T	T	F
T	F	F	F	T	F	F	T
F	T	T	F	T	T	F	T
F	F	T	F	F	T	T	F

Для знаходження значення істинності складного висловлювання потрібно надати значення істинності всім атомам, які містить відповідна формула. Набір значень істинності всіх атомів формули називають її *інтерпретацією.* Для обчислення значень істинності формули, яка відображає складне висловлювання, потрібно знаходити значення логічних операцій, визначених у таблиці 1.1. Послідовність виконання обчислень задають парами дужок. Якщо формула має n атомів, то існує 2^n способів надати значення істинності її атомам, тобто така формула має 2^n інтерпретацій, а всі її значення можна звести в таблицю істинності з 2^n рядками. Формулу, яка містить n атомів, називають n-місною.

Формулу називають *виконанною,* якщо існує принаймні одна інтерпретація, у якій ця формула набуває значення „істина”. У такому разі говорять, що формула *виконується* в цій інтерпретації. Формулу називають *тавтологією,* якщо вона виконується в усіх інтерпретаціях. Формулу, хибну в усіх інтерпретаціях, називають *заперечуваною* або *невиконанною.*

Оскільки кожна формула логіки висловлювань має скінченну кількість інтерпретацій, то завжди можна перевірити її на тавтологію або заперечуваність, знайшовши значення істинності в усіх можливих інтерпретаціях.

В таблицях 1.2 та 1.3 наведено таблиці істинності тавтології та заперечуваної формули.

Таблиця 1.2

Таблиця істинності формули $((a \to b) \wedge a) \to b$

a	b	$a \to b$	$((a \to b) \wedge a)$	$((a \to b) \wedge a) \to b$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

Таблиця 1.3

Таблиця істинності формули $(a \to b) \wedge (a \wedge \overline{b})$

a	b	$a \to b$	$\overline{b}$	$a \wedge \overline{b}$	$(a \to b) \wedge (a \wedge \overline{b})$
T	T	T	F	F	F
T	F	F	T	T	F
F	T	T	F	F	F
F	F	T	T	F	F

Проінтерпретувати логічну формулу часом можна не тільки за допомогою побудови для неї таблиці істинності, а спростивши її за допомогою *еквівалентних перетворень*. У таблиці 1.4 наведено основні еквівалентні перетворення логіки висловлювань.

Таблиця 1.4

Основні еквівалентні перетворення логіки висловлювань

Назва	Формулювання
1. Закони комутативності	*а)* $a \vee b = b \vee a$; *б)* $a \wedge b = b \wedge a$
2. Закони асоціативності	*а)* $(a \vee b) \vee c = a \vee (b \vee c)$; *б)* $(a \wedge b) \wedge c = a \wedge (b \wedge c)$
3. Закони дистрибутивності	*а)* $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$; *б)* $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
4. Закон суперечності	$a \wedge \overline{a} = \mathrm{F}$
5. Закон виключеного третього	$a \vee \overline{a} = \mathrm{T}$

Назва	Формулювання
6. Закон подвійного заперечення	$\overline{\overline{a}} = a$
7. Закони ідемпотентності	*а)* $a \vee a = a$; *б)* $a \wedge a = a$
8. Закони де Моргана	*а)* $\overline{a \vee b} = \overline{a} \wedge \overline{b}$; *б)* $\overline{a \wedge b} = \overline{a} \vee \overline{b}$
9. Закони поглинання	*а)* $(a \vee b) \wedge a = a$; *б)* $(a \wedge b) \vee a = a$
10. Співвідношення для сталих	*а)* $a \vee \mathrm{T} = \mathrm{T}$; *б)* $a \wedge \mathrm{T} = p$; ✓ *в)* $a \vee \mathrm{F} = p$; *г)* $a \wedge \mathrm{F} = \mathrm{F}$

Комп'ютери опрацьовують інформацію за допомогою бітів. *Біт* має два можливі значення – 0 (нуль) і 1 (одиниця). Його можна використовувати для подання значень істинності T й F. Зазвичай 1 використовують для зображення „істина", а 0 – для зображення „хибність". Змінну називають *булевою,* якщо вона набуває значення T (1) або F (0). Отже, булеві змінні можна подати за допомогою бітів.

Комп'ютерні операції над бітами відповідають логічним операціям. Замінивши T на 1, а F – на 0 у таблицях істинності для логічних операцій $\neg$, $\wedge$, $\vee$ та $\oplus$ отримаємо таблиці відповідних операцій над бітами. Будемо використовувати операції NOT, AND, OR та XOR відповідно для логічних операцій $\neg$, $\wedge$, $\vee$ та $\oplus$ як у багатьох мовах програмування. Значення операцій NOT, AND, OR та XOR над бітами наведено в таблиці 1.5.

Таблиця 1.5

Таблиця істинності операцій NOT, AND, OR та XOR над бітами

a	*b*	NOT *a*	*a* AND *b*	*a* OR *b*	*a* XOR *b*
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0

Бітовим рядком називають скінченну послідовність бітів. Розглядають також рядок, який не містить жодного біта (порожній рядок). *Довжиною* бітового рядка називають кількість бітів у ньому. Наприклад, 110010100 – бітовий рядок довжиною дев'ять.

Операції над окремими бітами можна узагальнити на бітові рядки. Визначимо операції *порозрядне* NOT, *порозрядне* AND, *порозрядне* OR і *порозрядне* XOR двох бітових рядків з однією довжиною, як бітовий рядок, який має таку саму довжину, а його біти – відповідно результати операцій NOT, AND, OR та XOR над відповідними бітами цих рядків.

Приклад.

Знайдемо результати операцій порозрядного AND, порозрядного OR і порозрядного XOR бітових рядків 1011000011 і 1101010101.

1011000011
1101010101
1001000001 – порозрядне AND
1111010111 – порозрядне OR
0110010110 – порозрядне XOR

Хід роботи:

Частина 1. Побудова логічних зв'язків

1. Створити бібліотеку „Логічна консоль" `LogCon` (файли `LogCon.h`, `LogCon.cpp`).
2. У бібліотеці `LogCon` (файл `LogCon.cpp`) реалізувати функції, що відповідають логічним зв'язкам: „заперечення" у вигляді функції `NOT(a)`; „кон'юнкція" у вигляді функції `AND(a,b)`; „диз'юнкція" у вигляді функції `OR(a,b)`; „імплікація" у вигляді функції `IMP(a,b)`; „еквівалентність" у вигляді функції `EQU(a,b)`; „виключаюче „або" у вигляді функції `XOR(a,b)`. Функції повинні отримувати аргументи та повертати значення логічного типу. Реалізація функцій повинна містити конструкцію типу: `if (<умова>) <оператор1>; else <оператор2>;`. У заголовному файлі `LogCon.h` запрограмувати інтерфейси цих функцій.
3. Створити новий проект `Lab_1` та підключити до нього бібліотеку `LogCon`.

4. У функції `main()` проекту запрограмувати вивід у консоль таблиці 1.1. з використанням функцій модуля `LogCon`. Для цього у файлі з функцією `main()`, окрім бібліотек стандартного вводу/виводу потрібно підключити файл `LogCon.h`.
5. Відкомпілювати проект та запустити програму на виконання. Продемонструвати результат викладачеві.

Частина 2. Побудова таблиць істинності логічних функцій

1. У бібліотеці `LogCon` запрограмувати реалізацію логічної функції, заданої викладачем. Функція повинна отримувати аргументи та повертати значення логічного типу. Рекомендується назвати функцію `Fномер варіанту` (варіанти завдань наведені нижче).
2. У функції `main()` проекту `Lab_1` запрограмувати вивід у консоль таблиці істинності заданої логічної функції. Для цього потрібно почергово задавати аргументам функції всіх можливих значень істинності (тобто проінтерпретувати логічну функцію). Таблиця повинна містити колонки зі значеннями аргументів та колонку зі значенням функції.
3. Відкомпілювати проект та запустити програму на виконання. Для перевірки результатів побудувати таблицю істинності заданої логічної функції у зошиті. Продемонструвати результат викладачеві.

Варіанти завдань:

1. $f_1 = (c \vee a) \sim \left((a \to b) \oplus \left(a \vee \overline{b}\right)\right)$
2. $f_2 = \left(a \to (b \wedge c)\right) \sim \left((a \to b) \wedge (a \to c)\right)$
3. $f_3 = \left((a \wedge \overline{c}) \wedge b\right) \to \left((b \to c) \vee a\right)$
4. $f_4 = \left(\left(a \vee \overline{b}\right) \wedge a\right) \oplus (c \wedge b)$
5. $f_5 = \left((a \wedge b) \vee c\right) \sim (c \wedge \overline{a})$

6. $f_6 = \left(\overline{\left(a \wedge \overline{b}\right)} \vee c\right) \oplus \left((a \vee b) \vee c\right)$

7. $f_7 = \left(\overline{(\overline{a} \vee b)} \wedge c\right) \sim \left(\overline{b} \vee c\right)$

8. $f_8 = \left(a \vee (c \vee \overline{a})\right) \sim \left(\left(\overline{b} \wedge a\right) \oplus \left(\overline{b} \vee c\right)\right)$

9. $f_9 = \left(\overline{(a \wedge b)} \wedge c\right) \sim \left(\left(a \to \overline{b}\right) \vee c\right)$

10. $f_{10} = \left((b \wedge c) \to (a \vee c)\right) \sim \left((b \wedge \overline{c}) \to (c \wedge a)\right)$

11. $f_{11} = (a \to b) \wedge \left(\overline{c} \to \left(\overline{a} \oplus \overline{b}\right)\right)$

12. $f_{12} = \left(b \to (a \wedge c)\right) \sim \left((b \to a) \wedge (b \to c)\right)$

13. $f_{13} = \left((a \to b) \vee c\right) \to \left(\overline{a} \vee \overline{b}\right)$

14. $f_{14} = \left(\overline{(a \to b) \wedge \overline{c}}\right) \to (a \oplus \overline{c})$

15. $f_{15} = \left((a \to b) \wedge \left(\overline{c \vee a}\right)\right) \to \left(\overline{a} \vee \overline{b}\right)$

16. $f_{16} = \left((a \to b) \wedge c\right) \oplus \left(\left(\overline{a} \to \overline{b}\right) \vee c\right)$

17. $f_{17} = \left(a \wedge \left(\overline{b \vee \overline{c}}\right)\right) \sim (a \to b)$

18. $f_{18} = \left((a \oplus b) \wedge (b \to \overline{c})\right) \to (c \to a)$

19. $f_{19} = \left(a \wedge (b \vee c)\right) \to \left((a \wedge b) \oplus (b \wedge c)\right)$

20. $f_{20} = \left(\left(\overline{a \wedge \overline{c}}\right) \vee b\right) \to (b \vee c)$

21. $f_{21} = \left(\left(\overline{a \wedge \overline{b}}\right) \vee c\right) \sim (c \to b)$

22. $f_{22} = \left(\overline{a \to b}\right) \to \left(\left(a \wedge \overline{b}\right) \to c\right)$

23. $f_{23} = (a \vee b) \oplus \left(a \to (b \wedge \overline{c})\right)$

24. $f_{24} = \left((a \vee c) \to b\right) \to \left((a \to b) \vee (a \to c)\right)$

25. $f_{25} = \left(a \oplus (b \wedge c)\right) \sim \left((a \oplus b) \wedge (a \oplus c)\right)$