

Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки

та комп'ютерних технологій

Звіт

про виконання лабораторної роботи № 1

з курсу “Дискретна математика”

«Логіка висловлювань»

Виконав:

студент групи Фех-13

Назаренко Олександр Іванович

Перевірив:

доц. Романишин Р. І.

Львів-2024

Мета роботи: Навчитися визначати хибність і вірність одного твердження але не одне й інше одночасно. Навчитися працювати з логічними операціями. Створювати таблиці істинності одного із виразів.

### Хід роботи

- Створити бібліотеку „Логічна консоль” LogCon (файли LogCon.h, LogCon.cpp).
- У бібліотеці LogCon (файл LogCon.cpp) реалізуєм функції, що відповідають логічним зв'язкам

```
bool Not(bool a){ // return !a
    if (a){
        return false;
    }
    return true;
}

bool And(bool a, bool b){ // return a && b;
    if (a){
        if (b){
            return true;
        }
    }
    return false;
}

bool Or(bool a, bool b){ // return a || b;
    if (a){
        return true;
    }else if (b){
        return true;
    }
    return false;
}

bool If(bool a, bool b){ // if (a) {return b;} else {return !b;}
    if (a){
        return b;
    }
    return Not(b);
}

bool Xnor(bool a, bool b){ // return a==b;
    if (a) {
        if (b) {
            return true;
        }
        return false;
    } else if (b) {
        return false;
    }
    return true;
}

bool Xor(bool a, bool b){ // return a!=b;
    if (a) {
        if (b) {
            return false;
        }
        return true;
    } else if (b) {
        return true;
    }
    return false;
}
```

- У заголовному файлі LogCon.h запрограмуємо інтерфейси цих функцій.

```
bool Not(bool a);
bool And(bool a, bool b);
bool Or(bool a, bool b);
bool If(bool a, bool b);
bool Xnor(bool a, bool b);
bool Xor(bool a, bool b);
```

- Створюємо новий проект Lab1.cpp та підключаємо до нього бібліотеку LogCon.h

```
#include "LogCon.h"
```

- Створюємо загальну функцію обчислення і виведення результатів до консолі

```
void solve_variant(string header[], int sizeofheader, int spaces[], function<bool(bool a, bool b, bool c)> funcs[], int var_num) {
    bool elements[] = {true, false};

    cout << " " << "Варіант " << var_num << "\n";

    cout << " | ";
    for (int i=0; i<sizeofheader; i++) {
        cout<< header[i] << " | ";
    }
    cout << "\n";

    //sort(elements.begin(), elements.end());
    for (bool c : elements) {
        for (bool b : elements) {
            for (bool a : elements) {
                cout << " | ";
                for (int iv=0; iv<sizeofheader; iv++) {
                    bool can_skip = true;
                    for (int v=0; v<spaces[iv]/2; v++) {
                        if (can_skip && spaces[iv]%2==0) {
                            can_skip = false;
                            continue;
                        }
                        cout << " ";
                    }
                    cout << (funcs[iv](a, b, c) ? "1" : "0");
                    for (int v=0; v<spaces[iv]/2; v++) { cout << " ";}
                    cout << " | ";
                }
                cout << endl;
            }
        }
    }
    cout << endl;
}
```

- У функції logic\_operators\_variant модуля LogCon варіант таблиці з використанням функцій логічних операцій модуля LogCon.

```

void logic_operators_variant() { // (a→b)∨(¬c→(¬a⊕¬b))
    auto one = [](bool a, bool b, bool c) {return a;};
    auto two = [](bool a, bool b, bool c) {return b;};
    auto three = [](bool a, bool b, bool c) {return Not(a);};
    auto four = [](bool a, bool b, bool c) {return And(a, b);};
    auto five = [](bool a, bool b, bool c) {return Or(a, b);};
    auto six = [](bool a, bool b, bool c) {return If(a, b);};
    auto seven = [](bool a, bool b, bool c) {return Xnor(a, b);};
    auto eight = [](bool a, bool b, bool c) {return Xor(a, b);};
    function<bool(bool a, bool b, bool c)> funcs[] = {one, two, three, four, five, six, seven, eight};

    string header[] = {"a", "b", "¬a", "a∧b", "a∨b", "a→b", "a~b", "a⊕b"};

    int spaces[] = {1, 1, 2, 3, 3, 3, 3, 3};

    string var_num = "логічні Операції";

    int sizeofheader = sizeof(header)/24;

    solve_variant(header, sizeofheader, spaces, funcs, var_num);
}

```

- У функції main проекту Lab\_1 запрограмуємо виклик функції варіанту.

```

int main(){
    SetConsoleOutputCP(CP_UTF8);
    // variant5();
    // variant8();
    // variant10();
    // additional_variant();
    logic_operators_variant();

    return 0;
}

```

- Компілюємо проект та запускаємо програму на виконання.

Варіант Логічні Операції									
a	b	¬a	a∧b	a∨b	a→b	a~b	a⊕b		
1	1	0	1	1	1	1	0		
0	1	1	0	1	0	0	1		
1	0	0	0	1	0	0	1		
0	0	1	0	0	1	1	0		

Частина №2

- У бібліотеці LogCon запрограмуємо реалізацію логічної функції  $(a \rightarrow b) \vee (\neg c \rightarrow (\neg a \oplus \neg b))$ . У функції solve\_variant проекту LogCon запрограмовано вивід у консоль таблиці істинності заданої логічної функції

```
void additional_variant() { // (a→b)∨(¬c→(¬a⊕¬b))
    auto one = [](bool a, bool b, bool c) {return a;};
    auto two = [](bool a, bool b, bool c) {return b;};
    auto three = [](bool a, bool b, bool c) {return c;};
    auto four = [](bool a, bool b, bool c) {return If(a, b);};
    auto five = [](bool a, bool b, bool c) {return Not(a);};
    auto six = [](bool a, bool b, bool c) {return Not(b);};
    auto seven = [](bool a, bool b, bool c) {return Not(c);};
    auto eight = [](bool a, bool b, bool c) {return Xor(Not(a), Not(b));};
    auto nine = [](bool a, bool b, bool c) {return If(Not(c), Xor(Not(a), Not(b)));};
    auto ten = [](bool a, bool b, bool c) {return Or(If(a, b), If(Not(c), Xor(Not(a), Not(b))));};
    function<bool(bool a, bool b, bool c)> funcs[] = {one, two, three, four, five, six, seven, eight, nine, ten};

    string header[] = {"a", "b", "c", "a→b", "¬a", "¬b", "¬c", "¬a⊕¬b", "¬c→(¬a⊕¬b)", "(a→b)∨(¬c→(¬a⊕¬b))"};

    int spaces[] = {1, 1, 1, 3, 2, 2, 2, 5, 10, 18};

    string var_num = "0";

    int sizeofheader = sizeof(header)/24;

    solve_variant(header, sizeofheader, spaces, funcs, var_num);
}
```

- Компілюємо проект і запускаємо його

Варіант 0										
a	b	c	a→b	¬a	¬b	¬c	¬a⊕¬b	¬c→(¬a⊕¬b)	(a→b)∨(¬c→(¬a⊕¬b))	
1	1	1	1	0	0	0	0	1	1	
0	1	1	0	1	0	0	1	0	0	
1	0	1	0	0	1	0	1	0	0	
0	0	1	1	1	1	0	0	1	1	
1	1	0	1	0	0	1	0	0	1	
0	1	0	0	1	0	1	1	1	1	
1	0	0	0	0	1	1	1	1	1	
0	0	0	1	1	1	1	0	0	1	

- Весь проект набув вигляду:

Варіант 0										
a	b	c	$a \rightarrow b$	$\neg a$	$\neg b$	$\neg c$	$\neg a \oplus \neg b$	$\neg c \rightarrow (\neg a \oplus \neg b)$	$(a \rightarrow b) \vee (\neg c \rightarrow (\neg a \oplus \neg b))$	
1	1	1	1	0	0	0	0	1	1	
0	1	1	0	1	0	0	1	0	0	
1	0	1	0	0	1	0	1	0	0	
0	0	1	1	1	1	0	0	1	1	
1	1	0	1	0	0	1	0	0	1	
0	1	0	0	1	0	1	1	1	1	
1	0	0	0	0	1	1	1	1	1	
0	0	0	1	1	1	1	0	0	1	

  

Варіант Логічні Операції								
a	b	$\neg a$	$a \wedge b$	$a \vee b$	$a \rightarrow b$	$a \sim b$	$a \oplus b$	
1	1	0	1	1	1	1	0	
0	1	1	0	1	0	0	1	
1	0	0	0	1	0	0	1	
0	0	1	0	0	1	1	0	
1	1	0	1	1	1	1	0	
0	1	1	0	1	0	0	1	
1	0	0	0	1	0	0	1	
0	0	1	0	0	1	1	0	

- Таблиця істиності заданої функції розвязана мною

a	b	c	$a \rightarrow b$	$\neg a$	$\neg b$	$\neg c$	$\neg a \oplus \neg b$	$\neg c \rightarrow (\neg a \oplus \neg b)$	$(a \rightarrow b) \vee (\neg c \rightarrow (\neg a \oplus \neg b))$
1	1	1	1	0	0	0	0	1	1
0	1	1	0	1	0	0	1	0	0
1	0	1	0	0	1	0	1	0	0
0	0	1	1	1	1	0	0	1	1
1	1	0	1	0	0	1	0	0	1
0	1	0	0	1	0	1	1	1	1
1	0	0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	0	0	1

**Висновок:** У цій лабораторній роботі я навчився працювати з логічними оперціями, створювати таблиці істиності. А також знаходити істиність складного виразу.

- Додаткові варіанти з лекції (5, 8, 10)

```
void variant8() {
    auto one = [](bool a, bool b, bool c) {return a;};
    auto two = [](bool a, bool b, bool c) {return b;};
    auto three = [](bool a, bool b, bool c) {return c;};
    auto four = [](bool a, bool b, bool c) {return Not(a);};
    auto five = [](bool a, bool b, bool c) {return Or(c, Not(a));};
    auto six = [](bool a, bool b, bool c) {return Or(a, Or(c, Not(a)));};
    auto seven = [](bool a, bool b, bool c) {return Not(b);};
    auto eight = [](bool a, bool b, bool c) {return And(Not(b), a);};
    auto nine = [](bool a, bool b, bool c) {return Or(Not(b), c);};
    auto ten = [](bool a, bool b, bool c) {return Xor(And(Not(b), a), Or(Not(b), c));};
    auto eleven = [](bool a, bool b, bool c) {return Xnor(Or(a, Or(c, Not(a))), Xor(And(Not(b), a), Or(Not(b), c)));};
    function<bool(bool a, bool b, bool c)> funcs[] = {one, two, three, four, five, six, seven, eight, nine, ten, eleven};

    string header[] = {"a", "b", "c", "~a", "c∨~a", "a∨(c∨~a)", "~b", "~b∧a", "~b∨c", "~b∧a⊕~b∨c", "(a∨(c∨~a))~((~b∧a)⊕(~b∨c))"};

    int spaces[] = {1, 1, 1, 2, 4, 8, 2, 4, 4, 9, 26};

    int var_num = 8;

    int sizeofheader = sizeof(header)/24;

    solve_variant(header, sizeofheader, spaces, funcs, var_num);
}
```

```
void variant5() {
    auto one = [](bool a, bool b, bool c) {return a;};
    auto two = [](bool a, bool b, bool c) {return b;};
    auto three = [](bool a, bool b, bool c) {return c;};
    auto four = [](bool a, bool b, bool c) {return And(a, b);};
    auto five = [](bool a, bool b, bool c) {return Or(And(a, b), c);};
    auto six = [](bool a, bool b, bool c) {return Not(a);};
    auto seven = [](bool a, bool b, bool c) {return And(c, Not(a));};
    auto eight = [](bool a, bool b, bool c) {return Xnor(Or(And(a, b), c), And(c, Not(a)));};
    function<bool(bool a, bool b, bool c)> funcs[] = {one, two, three, four, five, six, seven, eight};

    string header[] = {"a", "b", "c", "a∧b", "(a∧b)∨c", "~a", "c∧~a", "((a∧b)∨c)~(c∧~a)"};

    int spaces[] = {1, 1, 1, 3, 7, 2, 4, 16};

    int var_num = 5;

    int sizeofheader = sizeof(header)/24;

    solve_variant(header, sizeofheader, spaces, funcs, var_num);
}
```

```
void variant10() {
    auto one = [](bool a, bool b, bool c) {return a;};
    auto two = [](bool a, bool b, bool c) {return b;};
    auto three = [](bool a, bool b, bool c) {return c;};
    auto four = [](bool a, bool b, bool c) {return Not(c);};
    auto five = [](bool a, bool b, bool c) {return And(b, c);};
    auto six = [](bool a, bool b, bool c) {return Or(a, c);};
    auto seven = [](bool a, bool b, bool c) {return If(And(b, c), Or(a, c));};
    auto eight = [](bool a, bool b, bool c) {return And(b, Not(c));};
    auto nine = [](bool a, bool b, bool c) {return And(c, a);};
    auto ten = [](bool a, bool b, bool c) {return If(And(b, Not(c)), And(c, a));};
    auto eleven = [](bool a, bool b, bool c) {return Xnor(If(And(b, c), Or(a, c)), If(And(b, Not(c)), And(c, a)));};
    function<bool(bool a, bool b, bool c)> funcs[] = {one, two, three, four, five, six, seven, eight, nine, ten, eleven};

    string header[] = {"a", "b", "c", "~c", "b∧c", "a∨c", "(b∧c)→(a∨c)", "b∧~c", "c∧a", "(b∧~c)→(c∧a)", "((b∧c)→(a∨c))~((b∧~c)→(c∧a))"};

    int spaces[] = {1, 1, 1, 2, 3, 3, 11, 4, 3, 12, 28};

    string var_num = "10";

    int sizeofheader = sizeof(header)/24;

    solve_variant(header, sizeofheader, spaces, funcs, var_num);
}
```

- Компілюємо проект та запускаємо програму на виконання.

Варіант 5								
a	b	c	$a \wedge b$	$(a \wedge b) \vee c$	$\neg a$	$c \wedge \neg a$	$((a \wedge b) \vee c) \sim (c \wedge \neg a)$	
1	1	1	1	1	0	0	0	
0	1	1	0	1	1	1	1	
1	0	1	0	1	0	0	0	
0	0	1	0	1	1	1	1	
1	1	0	1	1	0	0	0	
0	1	0	0	0	1	0	1	
1	0	0	0	0	0	0	1	
0	0	0	0	0	1	0	1	

  

Варіант 8											
a	b	c	$\neg a$	$c \vee \neg a$	$a \vee (c \vee \neg a)$	$\neg b$	$\neg b \wedge a$	$\neg b \vee c$	$\neg b \wedge a \oplus \neg b \vee c$	$(a \vee (c \vee \neg a)) \sim ((\neg b \wedge a) \oplus (\neg b \vee c))$	
1	1	1	0	1	1	0	0	1	1	1	
0	1	1	1	1	1	0	0	1	1	1	
1	0	1	0	1	1	1	1	1	0	0	
0	0	1	1	1	1	1	0	1	1	1	
1	1	0	0	0	1	0	0	0	0	0	
0	1	0	1	1	1	0	0	0	0	0	
1	0	0	0	0	1	1	1	1	0	0	
0	0	0	1	1	1	1	0	1	1	1	

  

Варіант 10											
a	b	c	$\neg c$	$b \wedge c$	$a \vee c$	$(b \wedge c) \rightarrow (a \vee c)$	$b \wedge \neg c$	$c \wedge a$	$(b \wedge \neg c) \rightarrow (c \wedge a)$	$((b \wedge c) \rightarrow (a \vee c)) \sim ((b \wedge \neg c) \rightarrow (c \wedge a))$	
1	1	1	0	1	1	1	0	1	0	0	
0	1	1	0	1	1	1	0	0	1	1	
1	0	1	0	0	1	0	0	1	0	1	
0	0	1	0	0	1	0	0	0	1	0	
1	1	0	1	0	1	0	1	0	0	1	
0	1	0	1	0	0	1	1	0	0	0	
1	0	0	1	0	1	0	0	0	1	0	
0	0	0	1	0	0	1	0	0	1	1	

Код програми

LogCon.h

```

C LogCon.h M X  G LogCon.cpp M  G Lab_1.cpp M  compile & run.bat
lab1> C LogCon.h > ...
1  #include <iostream>
2  #include <windows.h>
3  #include <functional>
4
5  using namespace std;
6
7  bool Not(bool a);
8  bool And(bool a, bool b);
9  bool Or(bool a, bool b);
10 bool If(bool a, bool b);
11 bool Xnor(bool a, bool b);
12 bool Xor(bool a, bool b);
13 void solve_variant(string header[], int sizeofheader, int spaces[], function<bool(bool a, bool b, bool c)> funcs[], string var_num);
14 void variant5();
15 void variant8();
16 void variant10();
17 void additional_variant();
18 void logic_operators_variant();
19
20
21 |

```

LogCon.cpp



```
C LogCon.h M  LogCon.cpp M X  Lab_1.cpp M  compile & run.bat
lab1 > C LogCon.cpp > ...
1  #include "LogCon.h"
2  #include <functional>
3
4  bool Not(bool a){ // return !a
5      if (a){
6          return false;
7      }
8      return true;
9  }
10
11 bool And(bool a, bool b){ // return a && b;
12     if (a){
13         if (b){
14             return true;
15         }
16     }
17     return false;
18 }
19
20 bool Or(bool a, bool b){ // return a || b;
21     if (a){
22         return true;
23     } else if (b){
24         return true;
25     }
26     return false;
27 }
28
29 bool If(bool a, bool b){ // if (a) {return b;} else {return !b;}
30     if (a){
31         return b;
32     }
33     return Not(b);
34 }
35
C LogCon.h M X  LogCon.cpp M X  Lab_1.cpp M
lab1 > C LogCon.cpp > ...
35
36 bool Xnor(bool a, bool b){ // return a==b;
37     if (a) {
38         if (b) {
39             return true;
40         }
41         return false;
42     } else if (b) {
43         return false;
44     }
45     return true;
46 }
47 bool Xor(bool a, bool b){ // return a!=b;
48     if (a) {
49         if (b) {
50             return false;
51         }
52         return true;
53     } else if (b) {
54         return true;
55     }
56     return false;
57 }
58 }
```

```
C LogCon.h M  G LogCon.cpp M X  G Lab_1.cpp M  compile & run.bat
lab1 > G LogCon.cpp > ...
59 void variant5() {
60     auto one = [](bool a, bool b, bool c) {return a;};
61     auto two = [](bool a, bool b, bool c) {return b;};
62     auto three = [](bool a, bool b, bool c) {return c;};
63     auto four = [](bool a, bool b, bool c) {return And(a, b);};
64     auto five = [](bool a, bool b, bool c) {return Or(And(a, b), c);};
65     auto six = [](bool a, bool b, bool c) {return Not(a);};
66     auto seven = [](bool a, bool b, bool c) {return And(c, Not(a));};
67     auto eight = [](bool a, bool b, bool c) {return Xnor(Or(And(a, b), c), And(c, Not(a)));};
68     function<bool(bool a, bool b, bool c)> funcs[] = {one, two, three, four, five, six, seven, eight};
69
70     string header[] = {"a", "b", "c", "a^b", "(a^b)^c", "~a", "c^~a", "((a^b)^c)^(c^~a)"};
71
72     int spaces[] = {1, 1, 1, 3, 7, 2, 4, 16};
73
74     string var_num = "5";
75
76     int sizeofheader = sizeof(header)/24;
77
78     solve_variant(header, sizeofheader, spaces, funcs, var_num);
79 }
```

```
C LogCon.h M  G LogCon.cpp M X  G Lab_1.cpp M  compile & run.bat
lab1 > G LogCon.cpp > ...
59 void variant5() {
78     solve_variant(header, sizeofheader, spaces, funcs, var_num);
79 }
80
81 void variant8() {
82     auto one = [](bool a, bool b, bool c) {return a;};
83     auto two = [](bool a, bool b, bool c) {return b;};
84     auto three = [](bool a, bool b, bool c) {return c;};
85     auto four = [](bool a, bool b, bool c) {return Not(a);};
86     auto five = [](bool a, bool b, bool c) {return Or(c, Not(a));};
87     auto six = [](bool a, bool b, bool c) {return Or(a, Or(c, Not(a)));};
88     auto seven = [](bool a, bool b, bool c) {return Not(b);};
89     auto eight = [](bool a, bool b, bool c) {return And(Not(b), a);};
90     auto nine = [](bool a, bool b, bool c) {return Or(Not(b), c);};
91     auto ten = [](bool a, bool b, bool c) {return Xor(And(Not(b), a), Or(Not(b), c))};
92     auto eleven = [](bool a, bool b, bool c) {return Xnor(Or(a, Or(c, Not(a))), Xor(And(Not(b), a), Or(Not(b), c)));};
93     function<bool(bool a, bool b, bool c)> funcs[] = {one, two, three, four, five, six, seven, eight, nine, ten, eleven};
94
95     string header[] = {"a", "b", "c", "~a", "c^~a", "a^(c^~a)", "~b", "~b^a", "~b^c", "~b^a^~b^c", "(a^(c^~a))^(~b^a)^(~b^c)"};
96
97     int spaces[] = {1, 1, 1, 2, 4, 8, 2, 4, 4, 9, 26};
98
99     string var_num = "8";
100
101     int sizeofheader = sizeof(header)/24;
102
103     solve_variant(header, sizeofheader, spaces, funcs, var_num);
104 }
105 }
```

```
C LogCon.h M X LogCon.cpp M X Lab_1.cpp M compile & run.bat
lab1 > LogCon.cpp > ...
104 }
105
106 void variant10() {
107     auto one = [](bool a, bool b, bool c) {return a;};
108     auto two = [](bool a, bool b, bool c) {return b;};
109     auto three = [](bool a, bool b, bool c) {return c;};
110     auto four = [](bool a, bool b, bool c) {return Not(c);};
111     auto five = [](bool a, bool b, bool c) {return And(b, c);};
112     auto six = [](bool a, bool b, bool c) {return Or(a, c);};
113     auto seven = [](bool a, bool b, bool c) {return If(And(b, c), Or(a, c));};
114     auto eight = [](bool a, bool b, bool c) {return And(b, Not(c));};
115     auto nine = [](bool a, bool b, bool c) {return And(c, a);};
116     auto ten = [](bool a, bool b, bool c) {return If(And(b, Not(c)), And(c, a));};
117     auto eleven = [](bool a, bool b, bool c) {return Xnor(If(And(b, c), Or(a, c)), If(And(b, Not(c)), And(c, a)));};
118     function<bool(bool a, bool b, bool c)> funcs[] = {one, two, three, four, five, six, seven, eight, nine, ten, eleven};
119
120     string header[] = {"a", "b", "c", "-c", "b^c", "a^c", "(b^c)->(a^c)", "b^~c", "c^a", "(b^~c)->(c^a)", "((b^c)->(a^c))-~((b^~c)->(c^a))"};
121
122     int spaces[] = {1, 1, 1, 2, 3, 3, 11, 4, 3, 12, 28};
123
124     string var_num = "10";
125
126     int sizeofheader = sizeof(header)/24;
127
128     solve_variant(header, sizeofheader, spaces, funcs, var_num);
129 }
130 }
```

```
C LogCon.h M LogCon.cpp M X Lab_1.cpp M compile & run.bat
lab1 > LogCon.cpp > ...
130
131 void additional_variant() { // (a^b)v(-c->(-a^~b))
132     auto one = [](bool a, bool b, bool c) {return a;};
133     auto two = [](bool a, bool b, bool c) {return b;};
134     auto three = [](bool a, bool b, bool c) {return c;};
135     auto four = [](bool a, bool b, bool c) {return If(a, b);};
136     auto five = [](bool a, bool b, bool c) {return Not(a);};
137     auto six = [](bool a, bool b, bool c) {return Not(b);};
138     auto seven = [](bool a, bool b, bool c) {return Not(c);};
139     auto eight = [](bool a, bool b, bool c) {return Xor(Not(a), Not(b));};
140     auto nine = [](bool a, bool b, bool c) {return If(Not(c), Xor(Not(a), Not(b)));};
141     auto ten = [](bool a, bool b, bool c) {return Or(If(a, b), If(Not(c), Xor(Not(a), Not(b))))};
142     function<bool(bool a, bool b, bool c)> funcs[] = {one, two, three, four, five, six, seven, eight, nine, ten};
143
144     string header[] = {"a", "b", "c", "a^b", "~a", "~b", "~c", "~a^~b", "-c->(-a^~b)", "(a^b)v(-c->(-a^~b))"};
145
146     int spaces[] = {1, 1, 1, 3, 2, 2, 2, 5, 10, 18};
147
148     string var_num = "0";
149
150     int sizeofheader = sizeof(header)/24;
151
152     solve_variant(header, sizeofheader, spaces, funcs, var_num);
153 }
154 }
```

```

C LogCon.h M  G+ LogCon.cpp M X  G+ Lab_1.cpp M  compile & run.bat
lab1 > G+ LogCon.cpp > ...
154
155 void logic_operators_variant() { // (a→b)∨(¬c→(¬a⊕¬b))
156     auto one = [](bool a, bool b, bool c) {return a;};
157     auto two = [](bool a, bool b, bool c) {return b;};
158     auto three = [](bool a, bool b, bool c) {return Not(a);};
159     auto four = [](bool a, bool b, bool c) {return And(a, b);};
160     auto five = [](bool a, bool b, bool c) {return Or(a, b);};
161     auto six = [](bool a, bool b, bool c) {return If(a, b);};
162     auto seven = [](bool a, bool b, bool c) {return Xnor(a, b);};
163     auto eight = [](bool a, bool b, bool c) {return Xor(a, b);};
164     function<bool(bool a, bool b, bool c)> funcs[] = {one, two, three, four, five, six, seven, eight};
165
166     string header[] = {"a", "b", "¬a", "a∧b", "a∨b", "a→b", "a~b", "a⊕b"};
167
168     int spaces[] = {1, 1, 2, 3, 3, 3, 3, 3};
169
170     string var_num = "Логічні Операції";
171
172     int sizeofheader = sizeof(header)/24;
173
174     solve_variant(header, sizeofheader, spaces, funcs, var_num);
175 }
176
177 void solve_variant(string header[], int sizeofheader, int spaces[], function<bool(bool a, bool b, bool c)> funcs[], string var_num) {
178     bool elements[] = {true, false};
179
180     cout << " " << "Варіант " << var_num << "\n";
181
182     cout << " | ";
183     for (int i=0; i<sizeofheader; i++) {
184         cout<< header[i] << " | ";
185     }
186     cout << "\n";
187
188     //sort(elements.begin(), elements.end());
189     for (bool c : elements) {
190         for (bool b : elements) {
191             for (bool a : elements) {
192                 cout << " | ";
193                 for (int iv=0; iv<sizeofheader; iv++) {
194                     bool can_skip = true;
195                     for (int v=0; v<spaces[iv]/2; v++) {
196                         if (can_skip && spaces[iv]%2==0) {
197                             can_skip = false;
198                             continue;
199                         }
200                     }
201                     cout << " ";
202                     cout << (funcs[iv](a, b, c) ? "1" : "0");
203                     for (int v=0; v<spaces[iv]/2; v++) { cout << " ";}
204                     cout << " | ";
205                 }
206                 cout << endl;
207             }
208         }
209     }
210     cout << endl;
211 }
212

```

Lab\_1.cpp

C LogCon.h M

LogCon.cpp M

Lab\_1.cpp M X

lab1 > Lab\_1.cpp > main()

```
1  #include <iostream>
2  #include <windows.h>
3  #include <functional>
4  #include "LogCon.h"
5
6  using namespace std;
7
8  int main(){
9      SetConsoleOutputCP(CP_UTF8);
10     // variant5();
11     // variant8();
12     // variant10();
13     // additional_variant();
14     logic_operators_variant();
15
16     return 0;
17 }
18
```