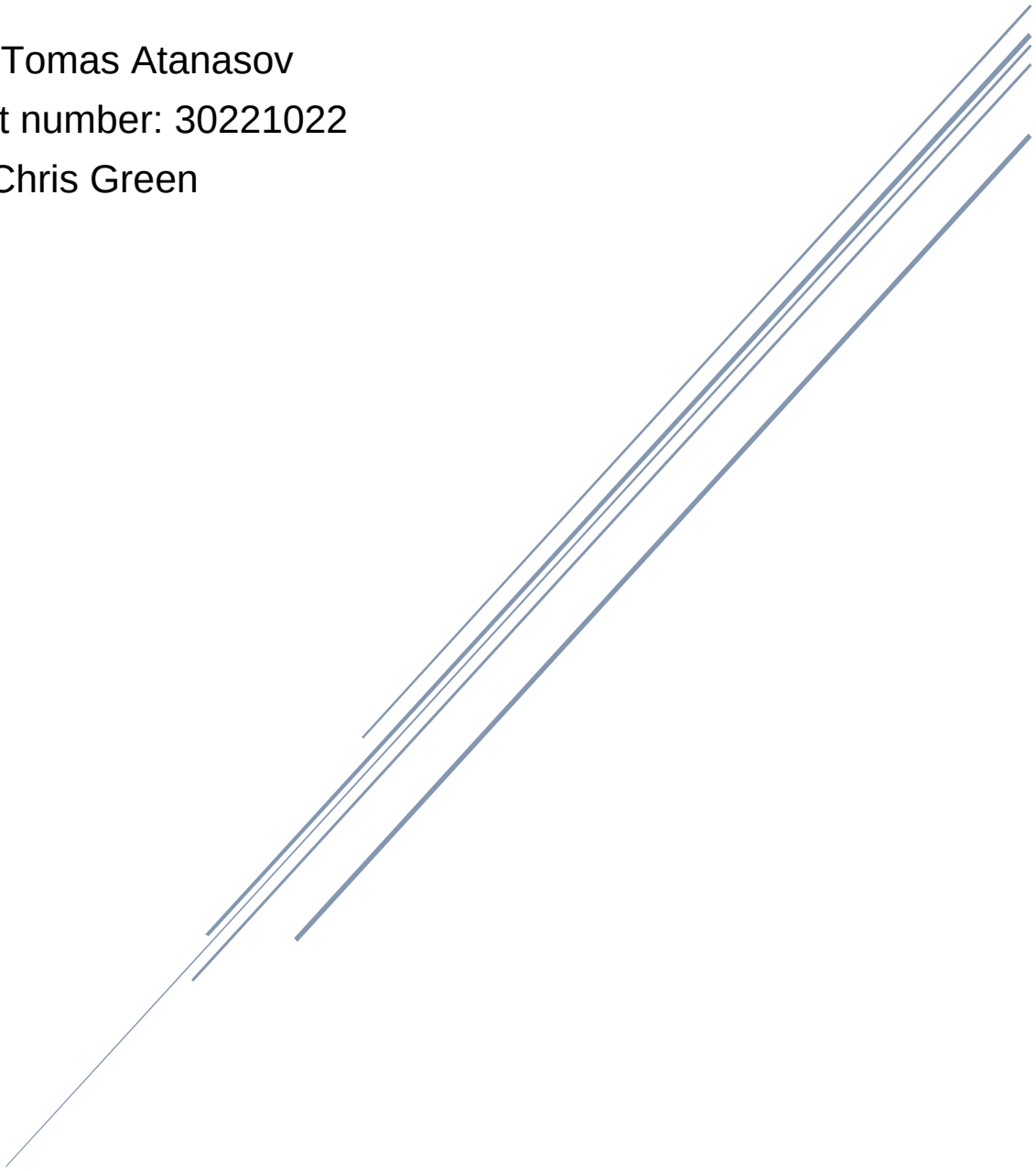
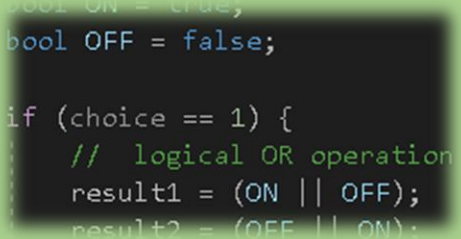
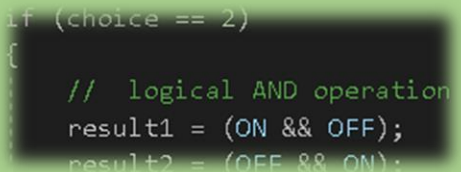
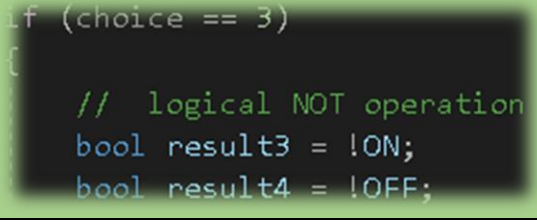
MATHEMATICAL CONCEPTS IN PROGRAMMING

Assignment 2 of 2

Name: Tomas Atanasov
Student number: 30221022
Tutor: Chris Green



Contents

TRUTH TABLE				
C++ Boolean Operators	A	B	c	
• OR (||)	0	0	0	
	0	1	1	
	1	0	1	
	1	1	1	
• AND (&&)	0	0	0	
	0	1	0	
	1	0	0	
	1	1	1	
• NOT (!)	0	1		
	1	0		

(The whole program is shown in FIG 1,2 and 3 in the Appendices section and link to it is provided at the end, FIG4 displays the additional XOR function)

Boolean operators

According to electronics-tutorials (2021), George Boole developed the Boolean Algebra during year 1854 to deal with logical functions with operators like AND, OR, and NOT represented by binary inputs (0 and 1). computerhope (2022) also suggest that those Boolean operators are commonly used in programming with conditional statements, in search engines, algorithms, or formulas.

As highlighted in the work of Layne (2019) The "OR" statement is a condition which is true if at least one out of multiple conditions is true. For example, represented in the context of a truth table, the "OR" operation retains a true output when either variable A or B hold a true value, or when both variables are true.

electronics-tutorials (2021) states that the concept of the “AND” function requires each one of the input events to occur simultaneously to trigger the output. This theory is fixed in the fundamental idea of the Boolean logic, in which the results are clearly assigned as either "TRUE" or "FALSE".

Learnatnoon (2023) explains that the ‘’NOT’’ logical Boolean operator simplifies Boolean expressions by negating a condition, usually represented by the ‘’!’’ symbol in most programming languages, it simply negates a Boolean value. If the operand is true, the ‘’NOT’’ operator’s result will be false, and vice versa. The work of Hollier C (2020) advice that the ‘’NOT’’ Boolean operator can be difficult and must be used with caution as it has the potential to delete and alter the results completely, and figuring out the reason behind it can be tricky. However, Learnatnoon (2023) emphasises the versatility of this logical operator when synchronised with other Boolean operators, like the use of ‘’NOT’’ operator in conjunction with the ‘’OR’’ operator to produce an exclusive’’XOR’’ operation. Furthermore Learnatnoon (2023) outlines several more benefits of using this Boolean operator, like its application in debugging of code by overruling the value of expressions, and the overall simplification it offers, since knowing that one of the values is false decreases overall complexity.

As Lomror (2023) observed, when using “XOR” operator, the outcome is always false (0) unless the input is different, in which case it is true (1). Furthermore, Lomror (2023) explains when “XOR” is used in negative numbers, it can make them positive by flipping all the bits and adding 1. The first bit in the result shows if it's positive or negative, and the rest represent the actual number. Additionally, according to Lomror (2023) “XOR” represents a versatile bitwise operator that is commonly used in mathematical procedures, and it can be applied in many different scenarios such as manipulation of binary data, cryptography, and array operations.

References

Computerhope (2022) *Boolean.* Available at: https://www.computerhope.com/jargon/b/boolean.htm. (Accessed on 17th of December 2023).

electronics-tutorials (2021) *Logic AND Function.* Available at: https://www.electronics-tutorials.ws/boolean/bool_1.html. (Accessed on 9th December 2023).

Hollier, C (2020) Research *Basics: Using Boolean Operators to Build a Search.* Available at: https://www.ifis.org/en/research-skills-blog/research-basics-boolean-operators . (Accessed on 15th of December 2023).

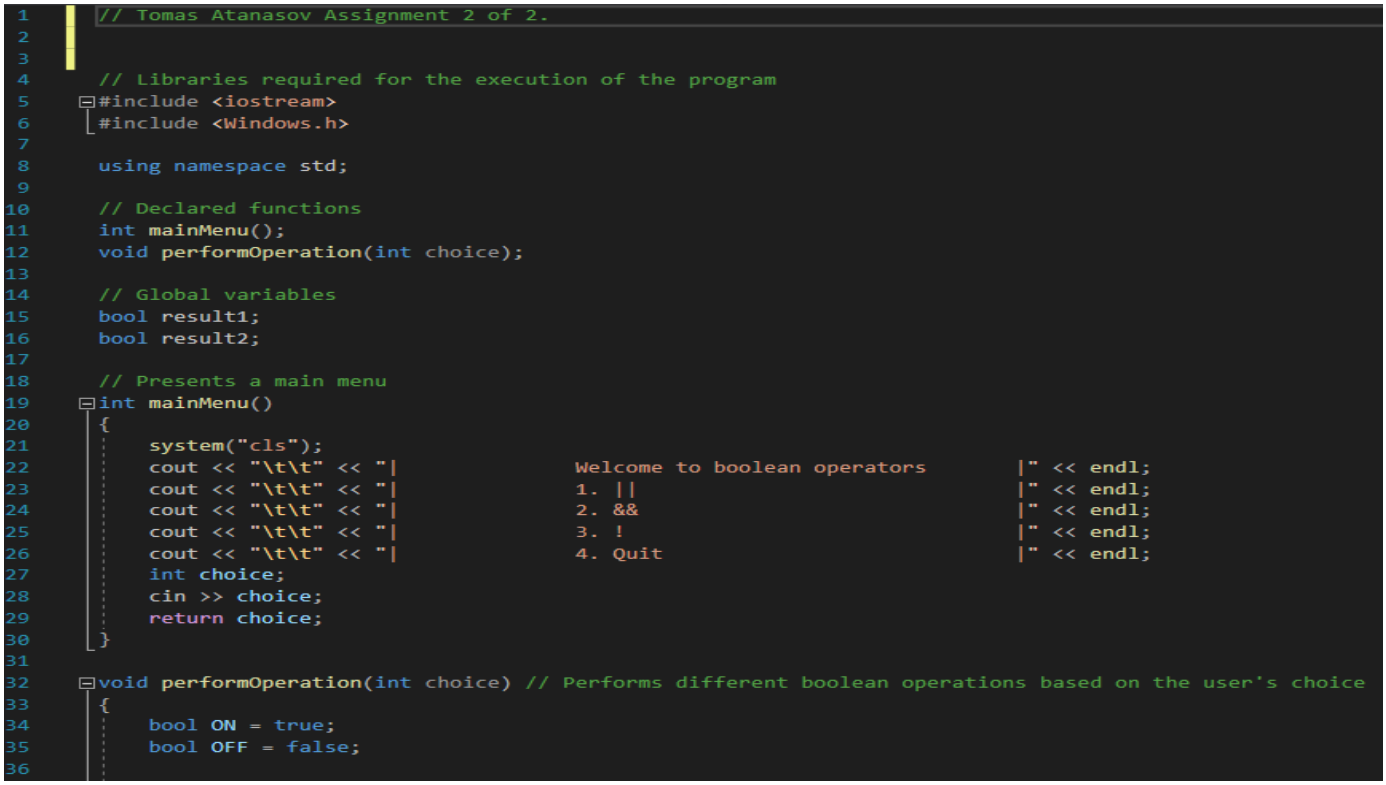
Layne, M (2019) *AND OR Boolean Logic.* Available at: https://dataschool.com/learn-sql/and-or-boolean-logic/. (Accessed on 9th December 2023).

Learnatnoon (2023) *What is a NOT operator in computer science?* Available at: https://www.learnatnoon.com/s/en-pk2/what-is-a-not-operator-in-computer-science/ . (Accessed on 15th of December 2023).

Lomror, K (2023) *How does bitwise ^ (XOR) work?* Available at: https://www.loginradius.com/blog/engineering/how-does-bitwise-xor-work/#:~:text=XOR%20is%20a%20bitwise%20operator,)%20else%20true(1 . (Accessed on 15th of December 2023).

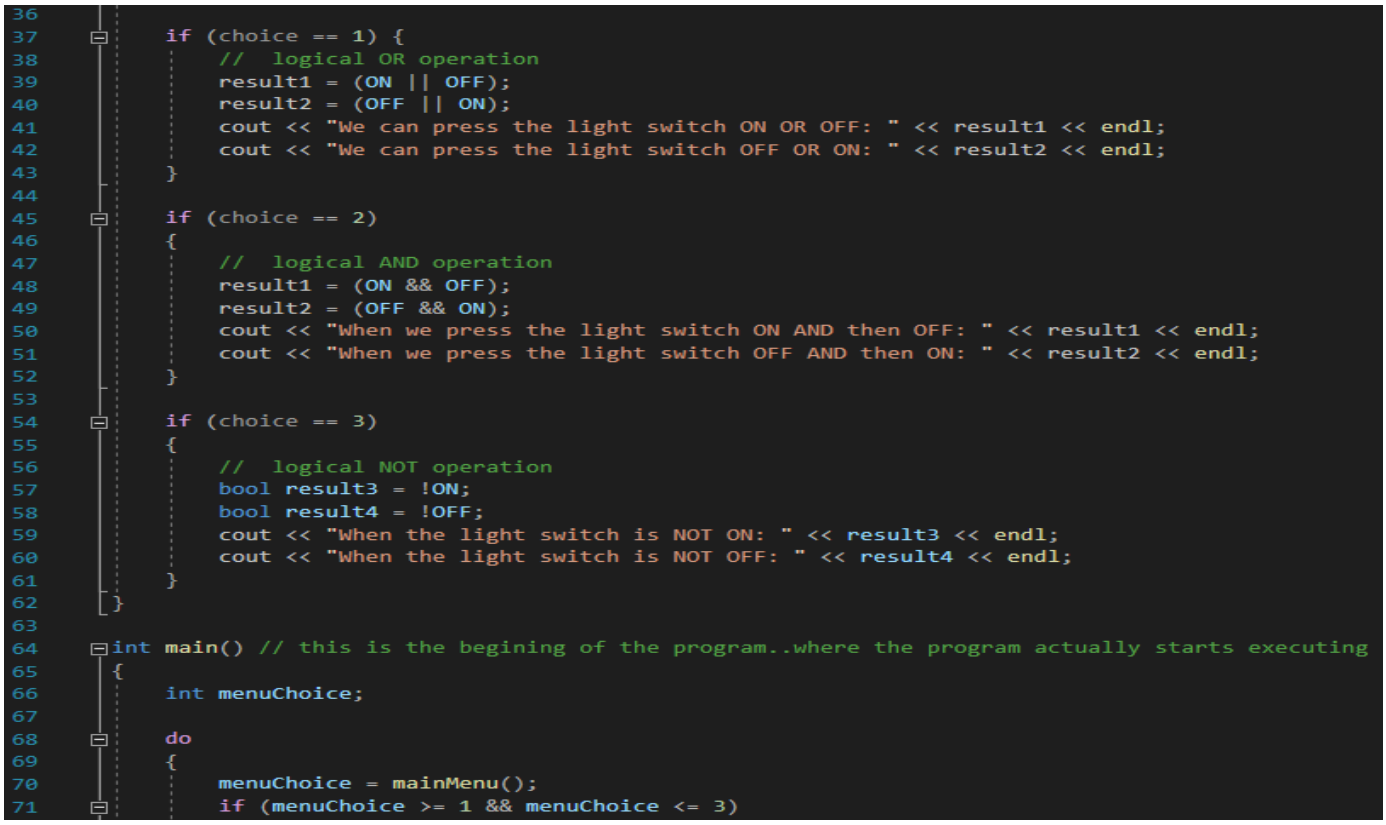
Appendices

FIG 1



```
// Tomas Atanasov Assignment 2 of 2.


// Libraries required for the execution of the program
#include <iostream>
#include <Windows.h>

using namespace std;

// Declared functions
int mainMenu();
void performOperation(int choice);

// Global variables
bool result1;
bool result2;

// Presents a main menu
int mainMenu()
{
    system("cls");
    cout << "\t\t" << "|              Welcome to boolean operators       |" << endl;
    cout << "\t\t" << "|              1. ||                              |" << endl;
    cout << "\t\t" << "|              2. &&                              |" << endl;
    cout << "\t\t" << "|              3. !                               |" << endl;
    cout << "\t\t" << "|              4. Quit                            |" << endl;
    int choice;
    cin >> choice;
    return choice;
}

void performOperation(int choice) // Performs different boolean operations based on the user's choice
{
    bool ON = true;
    bool OFF = false;

```

FIG 2



```
    if (choice == 1) {
        //  logical OR operation
        result1 = (ON || OFF);
        result2 = (OFF || ON);
        cout << "We can press the light switch ON OR OFF: " << result1 << endl;
        cout << "We can press the light switch OFF OR ON: " << result2 << endl;
    }

    if (choice == 2)
    {
        //  logical AND operation
        result1 = (ON && OFF);
        result2 = (OFF && ON);
        cout << "When we press the light switch ON AND then OFF: " << result1 << endl;
        cout << "When we press the light switch OFF AND then ON: " << result2 << endl;
    }

    if (choice == 3)
    {
        //  logical NOT operation
        bool result3 = !ON;
        bool result4 = !OFF;
        cout << "When the light switch is NOT ON: " << result3 << endl;
        cout << "When the light switch is NOT OFF: " << result4 << endl;
    }
}

int main() // this is the begining of the program..where the program actually starts executing
{
    int menuChoice;

    do
    {
        menuChoice = mainMenu();
        if (menuChoice >= 1 && menuChoice <= 3)
```

FIG 3

```
72          {
73              performOperation(menuChoice);
74          }
75          else if (menuChoice == 4)
76          {
77              cout << "BYE!" << endl;
78              break;
79          }
80          else
81          {
82              cout << "Sticky fingers? Please don't be cringe and enter a number between 1 and 4." << endl;
83              cout << "Error404" << endl;
84              cout << "You broke me..." << endl;
85              cout << "Well done..." << endl;
86
87          }
88
89          system("pause");
90      } while (menuChoice != 4);
91
92      return 0;
93  }
```

FIG4

C++ Boolean Operator	A	B	Output
• XOR (^)	0	0	0
	0	1	1
	1	0	1
	1	1	0

Link to program: [Tomas-Atanasov--18th-of-December---Boolean-Operators---Assignment-2-of-2-master.zip]