



## Summary CPP

Sessions No 8(20-10-2022)

- A **Boolean data type** is a data type that can only be *true* or *false*. Check the example shown below, when we try to print  $(x > y)$  it shows "0" which means when  $x$  is 10 and  $y$  is 20 then  $(x > y)$  is a **false** statement, but if we check  $(y > x)$  then it is true so the output will be 1.

```
Untitled1 boolean_if_else.cpp
1 #include "iostream"
2 using namespace std;
3 main(){
4
5     int x=10;
6     int y=20;
7     cout<<(x>y)<<endl;
8 }
9
```

```
C:\my_projects\CPP\boolean_if_else.exe
0
-----
Process exited after 0.1103 seconds with return value 0
Press any key to continue . . .
```

- Boolean datatype reserve **1 byte** of space in the MinGW compiler.

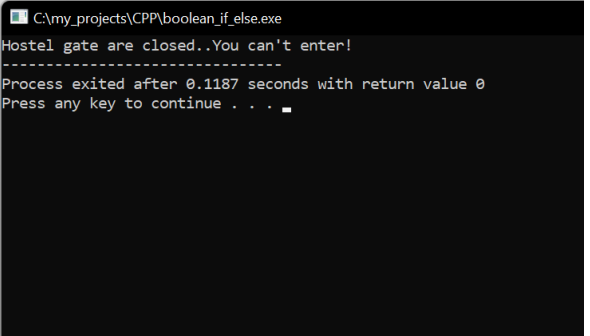
```
Untitled1 boolean_if_else.cpp
1 #include "iostream"
2 using namespace std;
3 main(){
4
5     int x=10;
6     int y=20;
7     bool z = x>y;
8     cout<<"value of z : "<<z<<endl;
9     cout<<"size of boolean datatype: "<<sizeof(z)<<endl;
10 }
11
```

```
C:\my_projects\CPP\boolean_if_else.exe
value of z : 0
size of boolean datatype: 1
-----
Process exited after 0.1015 seconds with return value 0
Press any key to continue . . .
```

- The **if/else statement** executes a block of code if a specified condition is **true**. If the condition is false, another block of code can be executed. Now in the given example, we have used if, else if, and else statements to create our algorithm. "**else if**"

statement will be executed when the “if” condition is “false”.

```
#include "iostream"
using namespace std;
main(){
    int current_time=1; //In 24 hour format-"1" here means 1AM(Night)
    /*Create a algorithm for hostel security guard. So that students will be not allowed to enter hostel after
    10PM("22" in 24 hour clock), And college gates will be open only between 6AM to 10PM.
    */
    if (current_time>22 || current_time<6)
    {
        cout<<"Hostel gate are closed..You can't enter!";
    }
    else if(current_time==22)
    {
        cout<<"Pay 10 rupees..and enter your hostel ";
    }
    else
    {
        cout << "gate is open..";
    }
}
```



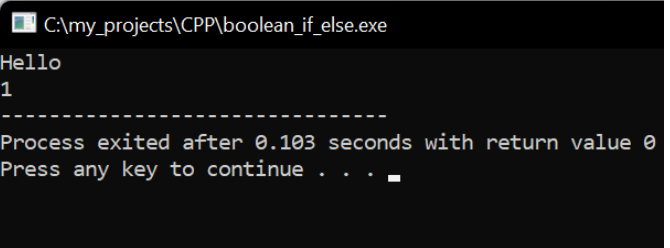
- All of these conditional statements always look for “True” i.e 1 if they are “true” then they will execute their block of code. So if we write:

```
if(True)
{
    cout<<"I am running..";
}
```

Now, this “if” statement will always run as we have passed directly True(1) in its condition.

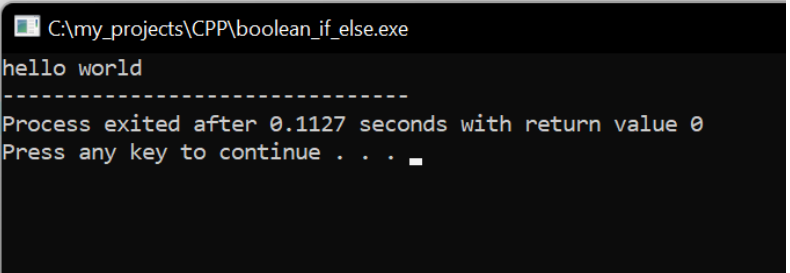
- When any statement runs successfully without any error then behind the scene it returns the “1” (exit code), otherwise, it will return “0” to our program. Now to prove this check the below example. Here “cout” is successfully executed, So it will return “1” to your program internally, *which here we are storing in a bool datatype variable “b”*

```
bool1 boolean_if_else.cpp
#include "iostream"
using namespace std;
main(){
    bool b= cout<<"Hello"<<endl;
    cout<< b;
}
```



- Here in the below example we have printed “Hello world” with the help of a conditional statement, As we know conditional statements always look for 0 or 1, so here in their condition we have provided a CPP statement, which if executed successfully then will return 1 internally to our “if” statement.

```
#include "iostream"
using namespace std;
main(){
    if(cout<<"hello "){
        cout<< "world";
    }
}
```



- **Logical Operators** - Logical operators are used to check whether an expression is true or false. They are used in decision-making.

Operator	Example	Meaning
&& (Logical AND)	expression1 && expression2	true only if both expression1 and expression2 are true
(Logical OR)	expression1    expression2	true if either expression1 or expression2 is true
! (Logical NOT)	! expression	true if the expression is false and vice versa

- Now as we know about **&& operator**, It will return *True* only when both the statement are *true*. And if we focus on its truth table:

Condition1	condition2	Result
0	0	0
0	1	0
1	0	0
1	1	1

Now we can see when the first condition is *false*( i.e 0) then obviously the result will be *false* too, as, in && operator, it returns *true* only when both conditions is *true*. Now with this knowledge, we can save our RAM space and computing power, As the first condition will be false then we will not check the second condition, and this concept here is called a **short circuit**. Check the below example for a better understanding.

```

boolean_if_else.cpp helloworld.cpp
1  #include "iostream"
2  using namespace std;
3  main(){
4  int x =10;
5  cout<<(x>20 && ++x);
6  cout<<endl;
7  cout<<"value of x: "<<x<<endl; //it will be still 10, as "++x" is not executed
8  cout<<(x>5 && ++x);
9  cout<<endl;
10 cout<<"value of x: "<<x<<endl; //it will be 11, as "x>5" was True, so "++x" was executed.
11 }
12
13
14
15
16 //
17 //int x=10;
18 //int x=20;

```

C:\my\_projects\CPP\boolean\_if\_else.exe

```

0
value of x: 10
1
value of x: 11
-----
Process exited after 0.1291 seconds with return value 0
Press any key to continue . . .

```

- Now as we know about the **|| operator**, It will return *True* when any of the conditions out of two is *true*. And if we focus on its truth table:

Condition1	condition2	Result
1	0	1
0	1	1
1	1	1
0	0	0

Now we can notice if the first condition is *true*, then we don't even need to check the second condition and the result will be *true*. So here also we have a **short circuit**. Check out the below example.

```

#include "iostream"
using namespace std;
main(){
int x =10;
cout<<(x>20 || ++x);
cout<<endl;
cout<<"value of x: "<<x<<endl; //it will be 11, as "++x" is executed, because (x>20) is false.
cout<<(x>5 || ++x);
cout<<endl;
cout<<"value of x: "<<x<<endl; //it will be still 11, as "x>5" was True, so "++x" was not needed to be executed.
}
//
//int x=10;

```

C:\my\_projects\CPP\boolean\_if\_else.exe

```

1
value of x: 11
1
value of x: 11

```

- Now with the help of *short-circuit*, we can create *if-else conditions* in one line, it is good in readability and does the same work. Check the below example here for the same question that is shown in starting of the Summary, We have written it with the help of the OR and AND operators.

boolean\_if\_else.cpp helloworld.cpp

```
1  #include "iostream"
2  using namespace std;
3  main(){
4  int current_time=1; //In 24 hour format-"1" here means 1AM(Night)
5
6  /*Create a algorithm for hostel security guard. So that students will be not allowed to enter hostel after
7  10PM("22" in 24 hour clock), And college gates will be open only between 6AM to 10PM.
8  */
9  (current_time>22 || current_time<6 )&& cout<<"gate is closed"<<endl || cout<<"gate is open"<<endl ; //inline code
10
11
12
13 }
14
15
16
```

C:\my\_projects\CPP\boolean\_if\_else.exe

gate is closed

-----  
Process exited after 0.1325 seconds with return value 0  
Press any key to continue . . .