Faculty of Computer Science and Information Technology
Department of Business Information Technology
Sana'a - Yemen

Lab Manual # 4&5

Prepared by Eng. Enas Aldahbali

Student Name			
Department			
Roll #			
Section			
Lab Date	-10-2024		
Submission Date			
Lab Grade:	10	**Obtained Grade**	
Instructor's Signature:			

A. ***Title:*** **More Flow of Control**
B. ***Objectives of this lab:***
- More Examples of C++ Basic input/output
- More examples of C++ programs
- Learn about simple flow of control (*if* and *if ... else* statements)

Introduction :-

we will learn to use the cin object to take input from the user, and the cout object to display output to the user with the help of examples.

in C++, cout sends formatted output to standard output devices, such as the screen. We use the cout object along with the inserting << operator for displaying output.

Activity 3.1: Output statement

Code Here:-

```
#include <iostream>
using namespace std;
int main() {
// prints the string enclosed in double quotes
 cout << "This is C++ Programming";
return 0;
}
```

result displayed on the screen

```
This is C++ Programming
```

How does this program work?

- We first include the iostream header file that allows us to display output.
- The cout object is defined inside the std namespace. To use the std namespace, we used the using namespace std; statement.
- Every C++ program starts with the main() function. The code execution begins from the start of the main() function.
- cout is an object that prints the string inside quotation marks " ". It is followed by the << operator.
- return 0; is the "exit status" of the main() function. The program ends with this statement, however, this statement is not mandatory.

Example 2: Numbers and Characters Output

To print the numbers and character variables, we use the same cout object but without using quotation marks

Code:-

```
#include <iostream>
using namespace std;
int main() {
int num1 = 70;
double num2 = 256.783;
char ch = 'A';
    cout << num1 << endl;     // print integer
    cout << num2 << endl;     // print double
    cout << "character: " << ch << endl;     // print char
    return 0;
}
```

result displayed on the screen

```
70
256.783
character: A
```

Notes:

- The endl manipulator is used to insert a new line. That's why each output is displayed in a new line.
- The << operator can be used more than once if we want to print different variables, strings and so on in a single statement. For example:

```
cout << "character: " << ch << endl;
```

Activity 3.2: input statement

In C++, cin takes formatted input from standard input devices such as the keyboard. We use the cin object along with the Extracting >> operator for taking input.

Example 3: Integer Input/Output

Code:-

```
#include <iostream>
using namespace std;
int main() {
int num;
  cout << "Enter an integer: ";
  cin >> num;  // Taking input
  cout << "The number is: " << num;
  return 0;
}
```

result displayed on the screen

```
Enter an integer: 70
The number is: 70
```

In the program, we used:-
cin >> num;
to take input from the user. The input is stored in the variable *num*. We use the >> operator with cin to take input.

Activity 3.2.2:C++ Taking Multiple Inputs:-

Code:-

```
#include <iostream>
using namespace std;
int main() {
char a;
int num;
   cout << "Enter a character and an integer: ";
    cin >> a >> num;
    cout << "Character: " << a << endl;
    cout << "Number: " << num;
 return 0;
}
```

result displayed on the screen

```
Enter a character and an integer: F
23
Character: F
Number: 23
```

Activity 3.3 :C++ if, if...else and Nested if...else:-

we will learn about the if...else statement to create decision making programs with the help of examples
In computer programming, we use the if statement to run a block code only when a certain condition is met.

For example, assigning grades (A, B, C) based on marks obtained by a student.

- if the percentage is above **90**, assign grade **A**
- if the percentage is above **75**, assign grade **B**
- if the percentage is above **65**, assign grade **C**

There are three forms of if...else statements in C++.

1. if statement
2. if...else statement
3. if...else if...else statement

Activity 3.4.1 C++ if statement:-

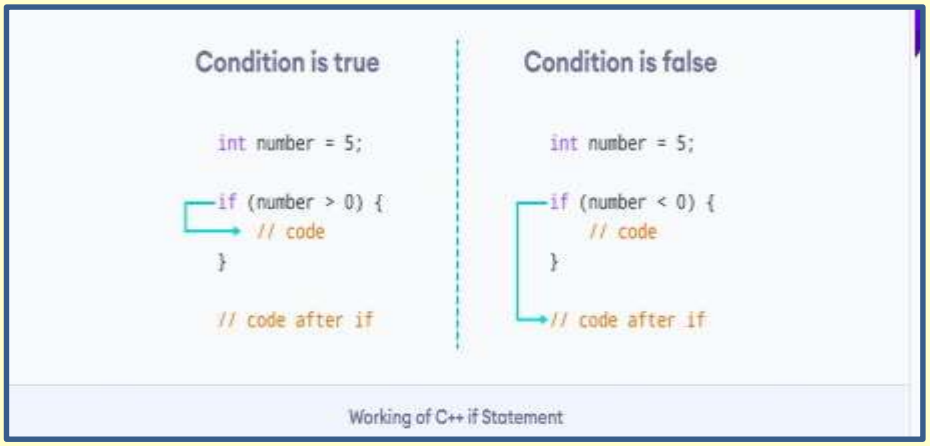
The syntax of the if statement is:

```
if (condition) {
   // body of if statement
}
```

The if statement evaluates the condition inside the parentheses ( ).

- If the condition evaluates to true, the code inside the body of if is executed.
- If the condition evaluates to false, the code inside the body of if is skipped.

Note: The code inside { } is the body of the if statement.



Example 1: C++ if Statement:-

Code:-

```
// Program to print positive number entered by the user
// If the user enters a negative number, it is skipped
#include <iostream>
using namespace std;
int main() {
int number;
   cout << "Enter an integer: ";
   cin >> number;
   // checks if the number is positive
   if (number > 0) {
 cout << "You entered a positive integer: " << number << endl;
   }
   cout << "This statement is always executed.";
 return 0;
}
```

Result 1 displayed on the screen

```
Enter an integer: 5
You entered a positive number: 5
This statement is always executed.
```

When the user enters 5, the condition number > 0 is evaluated to true and the statement inside the body of if is executed.

Result 2 displayed on the screen

```
Enter a number: -5
This statement is always executed.
```

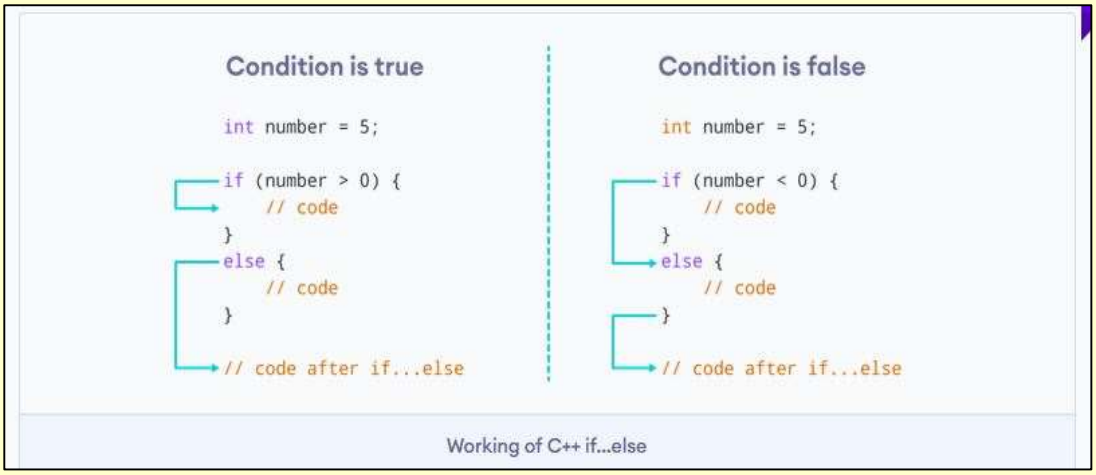
When the user enters -5, the condition number > 0 is evaluated to false and the statement inside the body of if is not executed

Activity 3.4.2 C++ if ..else statement:-

The if statement can have an optional else clause. Its syntax is:

```
if (condition) {
    // block of code if condition is true
}
else {
    // block of code if condition is false
}
```

The if..else statement evaluates the condition inside the parenthesis.



Working of C++ if...else

If the `condition` evaluates `true`,

- the code inside the body of `if` is executed
- the code inside the body of `else` is skipped from execution

If the `condition` evaluates `false`,

- the code inside the body of `else` is executed
- the code inside the body of `if` is skipped from execution

Example 2: C++ if...else Statement

Code:-

```
// Program to check whether an integer is positive or negative
// This program considers 0 as a positive number

#include <iostream>
using namespace std;
int main() {
int number;
  cout << "Enter an integer: ";
   cin >> number;
   if (number >= 0) {
      cout << "You entered a positive integer: " << number << endl;
   }
   else {
      cout << "You entered a negative integer: " << number << endl;
   }
   cout << "This line is always printed.";
return 0;
}
```

Result 1 displayed on the screen

```
Enter an integer: 4
You entered a positive integer: 4.
This line is always printed.
```

In the above program, we have the condition number >= 0. If we enter the number greater or equal to *0*, then the condition evaluates true.

Here, we enter *4*. So, the condition is true. Hence, the statement inside the body of if is executed.

Result 2 displayed on the screen

```
Enter an integer: -4
You entered a negative integer: -4.
This line is always printed.
```

Here, we enter *-4*. So, the condition is false. Hence, the statement inside the body of else is executed.

Activity 3.4.3 C++ if...else...else if statement

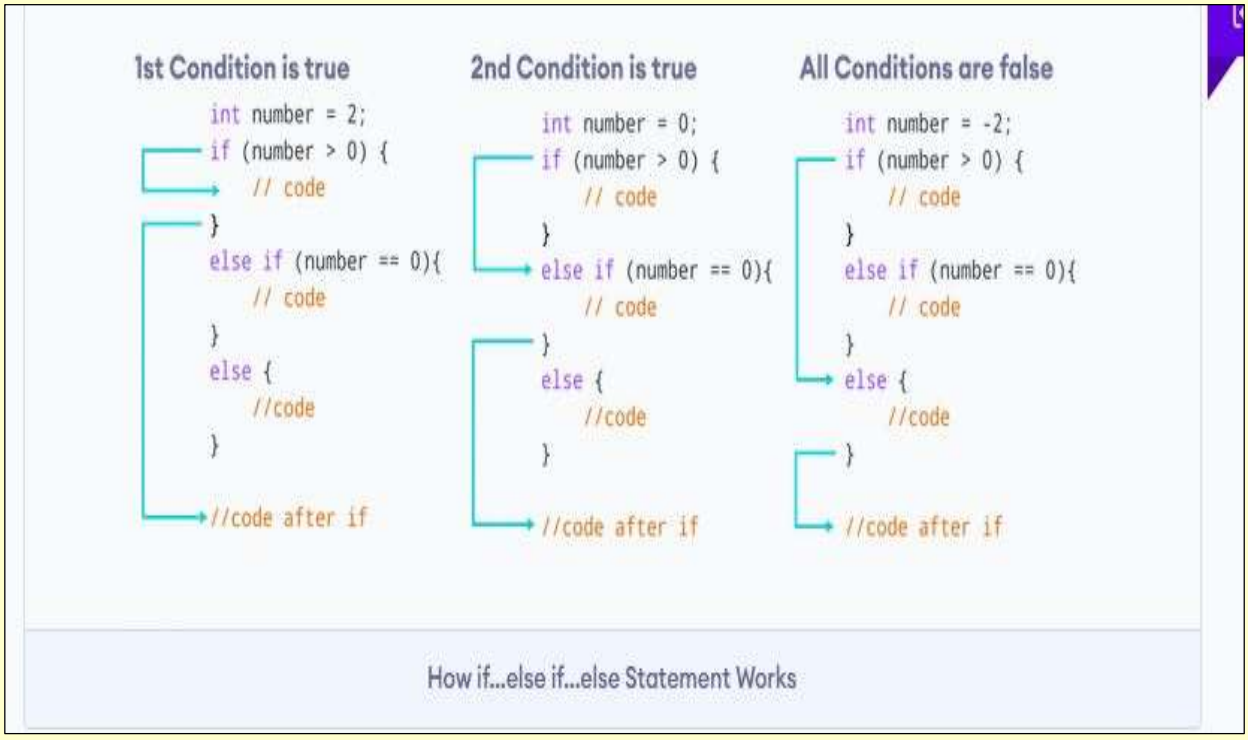
The if...else statement is used to execute a block of code among two alternatives. However, if we need to make a choice between more than two alternatives, we use the if...else if...else statement.

The syntax of the if...else if...else statement is:

```
if (condition1) {
    // code block 1
}
else if (condition2){
    // code block 2
}
else {
    // code block 3
}
```

Here,

- If `condition1` evaluates to `true`, the `code block 1` is executed.
- If `condition1` evaluates to `false`, then `condition2` is evaluated.
- If `condition2` is `true`, the `code block 2` is executed.
- If `condition2` is `false`, the `code block 3` is executed.



How if...else if...else Statement Works

Note: There can be more than one `else if` statement but only one `if` and `else` statements.

Example 3: C++ if...else...else if

Code:-

```
// Program to check whether an integer is positive, negative or zero
#include <iostream>
using namespace std;
int main() {
int number;
   cout << "Enter an integer: ";
   cin >> number;
   if (number > 0) {
      cout << "You entered a positive integer: " << number << endl;
   }
else if (number < 0) {
     cout << "You entered a negative integer: " << number << endl;
    }
else {
       cout << "You entered 0." << endl;
   }
    cout << "This line is always printed.";
   return 0;   }
```

Result 1 displayed on the screen

```
Enter an integer: 1
You entered a positive integer: 1.
This line is always printed.
```

Result 2 displayed on the screen

```
Enter an integer: -2
You entered a negative integer: -2.
This line is always printed.
```

Result 2 displayed on the screen

```
Enter an integer: 0
You entered 0.
This line is always printed.
```

In this program, we take a number from the user. We then use the `if...else if...else` ladder to check whether the number is positive, negative, or zero.

If the number is greater than `0`, the code inside the `if` block is executed. If the number is less than `0`, the code inside the `else if` block is executed. Otherwise, the code inside the `else` block is executed.

Exercise #1:-

Write a C++ program that compute the Area of square?

Write Your code Here

```
1
2
3
4
5
6
7
8
9
11
12
13
14
15
16
17
18
19
```

Exercise #2:-

write a C++ program that compute the Area of triangle?

Write Your code Here

```
1
2
3
4
5
6
7
8
9
11
12
13
14
15
16
17
18
19
```

Activity 3.4.4

In the programs you have seen so far, we have had a list of statements, which were executed in the order that they were written in your program (the .cpp file). In more complicated programs, you may need to change the order in which statements are executed. The order of execution for statements in your program is referred to as **flow of control**.

Let's look at one example:-

Suppose you are organizing an event that costs $12 for everyone older than 8 and $6 for any one 8 years or younger. One way to do this is to say the ticket is $12, unless you are 8 or younger, then it is $6. In this case, you can write:

Pseudo Code (English like statement)	C++ equivalent

ticket = 12;
if(age is 8 or younger)
  ticket = 6;
	double ticket = 12.00, age;
	if(age <= 8)
	  ticket = 6;

- There is another way to do this.

Pseudo Code (English like statement)	C++ equivalent

if(age is 8 or younger)
  ticket = 6;
else
  ticket = 12;
	if(age <= 8)
	  ticket = 6;
	else
	  ticket = 12;

- Both of these do the same thing. In both cases, you will change the flow of execution when you reach the statement; "age is 8 or younger". If that statement happens to be true, i.e., age is 8 or younger, then the value of ticket will change to $6, otherwise, you will go with its initialized value of $12.
- In general, the statement in the parentheses is either TRUE or FALSE. Depending on that being true or false, you will change the flow of execution of the statements in the program. In order for your program to decide about the flow of the execution, it uses a comparison operator.

Examples of comparison operators are:

1) *equal to* , = , which will be written as == (2 ='s) in C++, with a general form of: statement1== statement2. Example: y == x + 1

2) *not equal to* , ≠, which will be written as != in C++, with a general form of: statement1!= statement2. Example: y != x + 1

3) *less than* , <, which will be written as < in C++, with a general form of: statement1< statement2. Example: y < x + 1

4) *less than or equal to* , ≤, which will be written as <= in C++, with a general form of: statement1<= statement2. Example: y <= x + 1

5) *greater than* , >, which will be written as > in C++, with a general form of: statement1> statement2. Example: y > x + 1

6) *greater than or equal to* , ≥, which will be written as >= in C++, with a general form of: statement1>= statement2. Example: y >= x + 1

7) OR, which will be written as || (2 of the |'s) in C++, with a general form of: statement1|| statement2. Which may be True when either one of the two statements are TRUE.

8) AND, which will be written as && (2 of the &'s) in C++, with a general form of: statement1 &&statement2.

Now that you have learned about changing the flow of control, let's write a program called ticket.cpp that asks users to enter an age and displays the cost of the ticket based on the criteria that was given above. Use both methods to make sure

2-A) Desktop Testing

Now that you have the algorithm, test it to see if it works

3) Implementing the Algorithm in C++

Write Your code Here

```
1  // ticket.cpp -  This program asks for an age and displays the cost of the
2  ticket
   #include<iostream.h>
3  using namespace std;
4  int main( )
   {
5  double age, ticket = 12;
   cout << "Please enter the age \n";
6.  cin >> age;
7  if(age <= 8)
    ticket = 6;
8  cout << "Your ticket costs " << ticket << endl;
9    return 0;
   }
10
```

Activity 3.4.5 C++ Nested if...else

Sometimes, we need to use an `if` statement inside another `if` statement. This is known as nested `if` statement.

Think of it as multiple layers of `if` statements. There is a first, outer `if` statement, and inside it is another, inner `if` statement. Its syntax is:

```
// outer if statement
if (condition1) {
    // statements

    // inner if statement
    if (condition2) {
        // statements
    }
}
```

Notes:

- We can add else and else if statements to the inner if statement as required.
- The inner if statement can also be inserted inside the outer else or else if statements (if they exist).
- We can nest multiple layers of if statements.

In the below example,

- We take an integer as an input from the user and store it in the variable *num*.
- We then use an `if...else` statement to check whether num is not equal to `0`.
  - If `true`, then the **inner** `if...else` statement is executed.
  - If `false`, the code inside the **outer** `else` condition is executed, which prints "The number is 0 and neither even nor odd."
- The **inner** `if...else` statement checks whether the input number is divisible by `2`.
  - If `true`, then we print a statement saying that the number is even.
  - If `false`, we print that the number is odd.

Notice that `0` is also divisible by `2`, but it is actually not an even number. This is why we first make sure that the input number is not `0` in the outer `if` condition.

Note: As you can see, nested `if...else` makes your logic complicated. If possible, you should always try to avoid nested `if...else`.

Code:-

```
// C++ program to find if an integer is even or odd or neither (0)
// using nested if statements
#include <iostream>
using namespace std;
int main() {
   int num;
   cout << "Enter an integer: ";
    cin >> num;
// outer if condition
   if (num != 0) {
      // inner if condition
      if ((num % 2) == 0) {
         cout << "The number is even." << endl;
      }
       // inner else condition
      else {
         cout << "The number is odd." << endl;
      }
   }
   // outer else condition
   else {
      cout << "The number is 0 and it is neither even nor odd." << endl;
   }
   cout << "This line is always printed." << endl;
return 0;
}
```

Result 1 displayed on the screen

```
Enter an integer: 34
The number is even.
This line is always printed.
```

Result 2 displayed on the screen

```
Enter an integer: 35
The number is odd.
This line is always printed.
```

Result 3 displayed on the screen

```
Enter an integer: 0
The number is 0 and it is neither even nor odd.
This line is always printed.
```

Assignment #1:-

Change the ticket.cpp program such that it displays $6 for people who are 8 years old or younger OR 65 years or older.

Write Your code Here

```
1
2
3
4
5
6
7
8
9
11
12
13
14
15
16
17
18
19
```

Assignment #2:-

write a C++ program that enter number by user and print if this number is positive even number or negative even number or positive odd number or negative odd number?

Write Your code Here

```
1
2
3
4
5
6
7
8
9
11
12
13
14
15
16
17
18
19
```

Assignment #3:-

Write C++ Program to Check Whether a Character is Vowel or Consonant.?

Write Your code Here

```
1
2
3
4
5
6
7
8
9
11
12
13
14
15
16
17
18
19
```