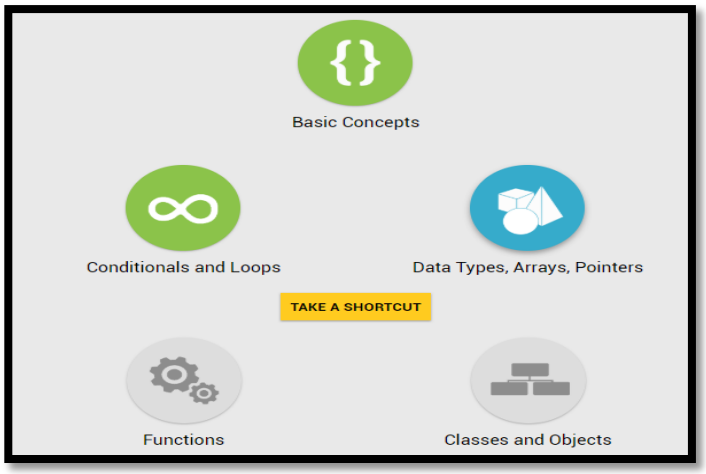
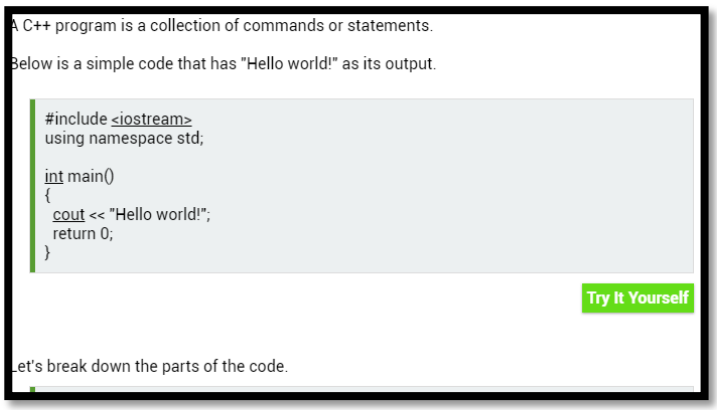
DAILY ASSESSMENT

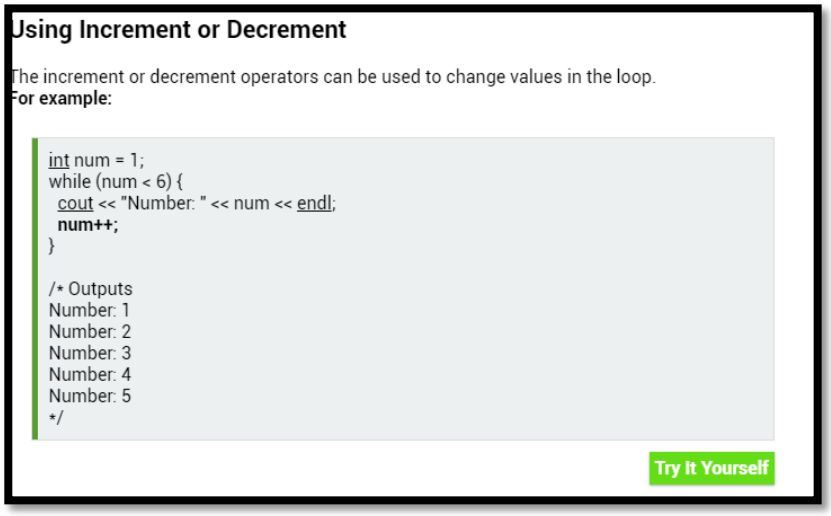
Date:	**22/06/2020**	**Name:**	**Davis S. Patel**
Course:	**C++ Programming**	**USN:**	**4AL16EC045**
Topic:	**Module 1** **Module 2**	**Semester & Section:**	**8th - A**
GitHub Repository:	**Davis**		

FORENOON & AFTERNOON SESSION DETAILS

Image of session







REPORT –

Module 1: Basic Concepts

The prime purpose of C++ programming was to add object orientation to the C programming language, which is in itself one of the most powerful programming languages.

The core of the pure object-oriented programming is to create an object, in code, that has certain properties and methods. While designing C++ modules, we try to see whole world in the form of objects. For example a car is an object which has certain properties such as color, number of doors, and the like. It also has certain methods such as accelerate, brake, and so on.

Object

This is the basic unit of object oriented programming. That is both data and function that operate on data are bundled as a unit called as object.

Class

When you define a class, you define a blueprint for an object. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

Abstraction

Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

For example, a database system hides certain details of how data is stored and created and maintained. Similar way, C++ classes provides different methods to the outside world without giving internal detail about those methods and data.

Encapsulation

Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object-oriented programming provides you framework to place the data and the relevant functions together in the same object.

Inheritance

One of the most useful aspects of object-oriented programming is code reusability. As the name suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.

This is a very important concept of object-oriented programming since this feature helps to reduce the code size.

Polymorphism

The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism. Poly refers to many. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism.

Overloading

The concept of overloading is also a branch of polymorphism. When the exiting operator or function is made to operate on new data type, it is said to be overloaded.

A variable provides us with named storage that our programs can manipulate. Each variable in C++ has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C++ is case-sensitive −

Module 2: Conditionals and Loops

C++ Conditions and If Statements

C++ supports the usual logical conditions from mathematics:

- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b

- Greater than or equal to: a >= b
- Equal to a == b
- Not Equal to: a != b

You can use these conditions to perform different actions for different decisions.

C++ has the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

C++ programming language provides the following type of loops to handle looping requirements.

Sr.No	Loop Type & Description
1	while loop Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2	for loop Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	do...while loop Like a 'while' statement, except that it tests the condition at the end of the loop body.
4	nested loops

	You can use one or more loop inside any another ‘while’, ‘for’ or ‘do..while’ loop.

While Loop example in C++

```
#include <iostream>

using namespace std;

int main(){

   int i=1;

   while(i<=6){

      cout<<"Value of variable i is: "<<i<<endl; i++;

   }

}
```

do-while loop example in C++

```
#include <iostream>

using namespace std;

int main(){

   int num=1;

   do

{

      cout<<"Value of num: "<<num<<endl;

      num++;

   }
```

```
while(num<=6);
   return 0;
}
```

Example of Switch Case

```
#include <iostream>
using namespace std;
int main(){
   int num=5;
   switch(num+2) {
      case 1:
        cout<<"Case1: Value is: "<<num<<endl;
      case 2:
        cout<<"Case2: Value is: "<<num<<endl;
      case 3:
        cout<<"Case3: Value is: "<<num<<endl;
      default:
        cout<<"Default: Value is: "<<num<<endl;
   }
   return 0;
}
```