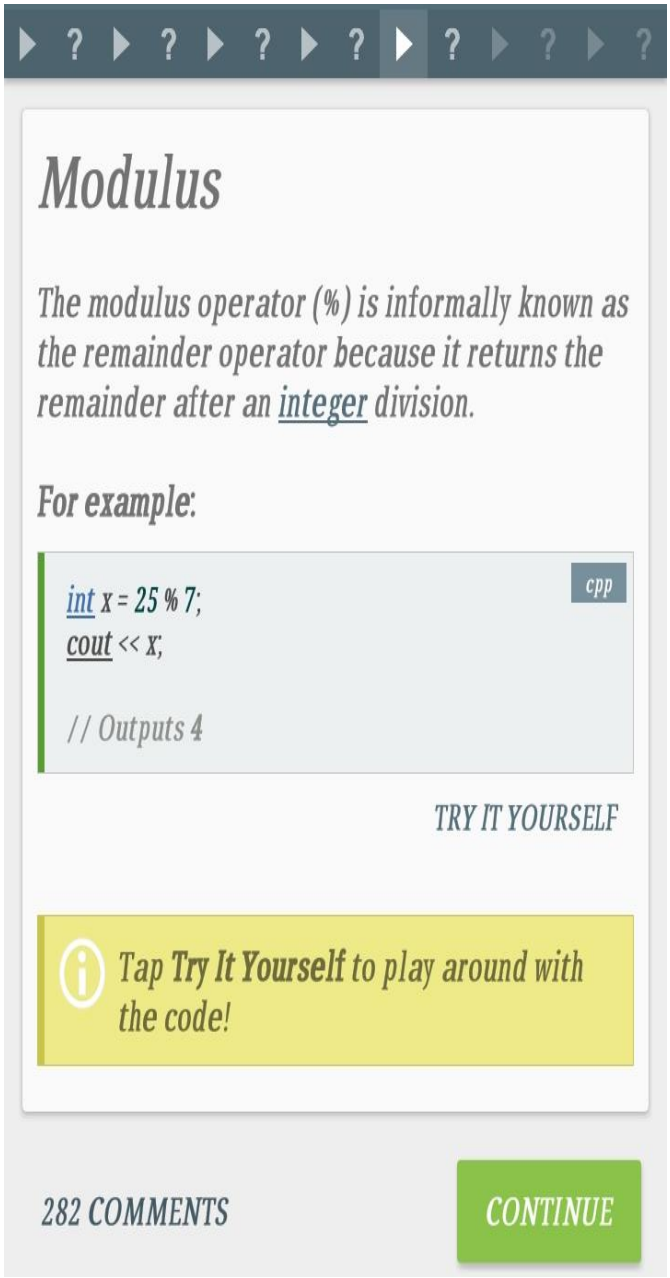
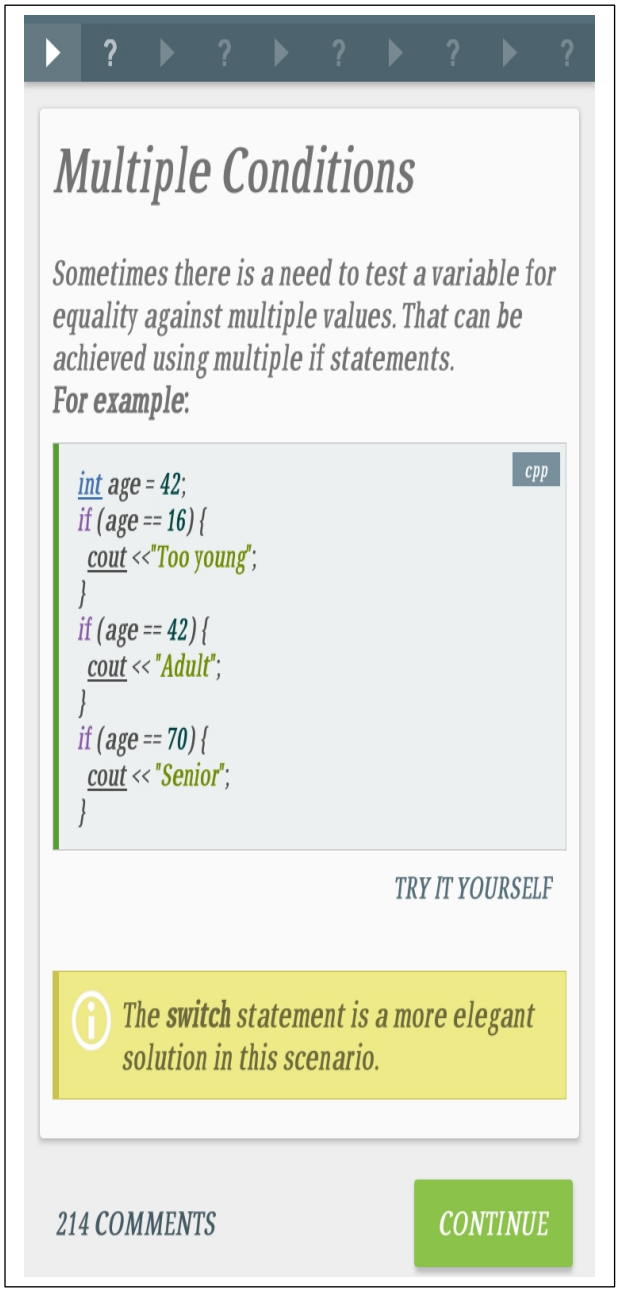
DAILY ASSESSMENT FORMAT

Date:	22nd June 2020	**Name:**	Soundarya NA
Course:	C++ programming	**USN:**	4AL16EC077
Topic:	Basic concepts Conditional and loops	**Semester & Section:**	8th - B

FORENOON SESSION DETAILS

Image of session





Report:

Object oriented programming is a type of programming which uses objects and classes its functioning. The object-oriented programming is based on real world entities like inheritance, polymorphism, data hiding, etc. It aims at binding together data and function work on these data sets into a single entity to restrict their usage.

Some basic concepts of object-oriented programming are −

- CLASS
- OBJECTS
- ENCAPSULATION
- POLYMORPHISM
- INHERITANCE
- ABSTRACTION

Class:

A class is a data-type that has its own members i.e. data members and member functions. It is the blueprint for an object in object-oriented programming language. It is the basic building block of object-oriented programming in c++. The members of a class are accessed in programming language by creating an instance of the class.

Some important properties of class are −

- Class is a user-defined data-type
- A class contains members like data members and member functions
- Data members are variables of the class
- Member functions are the methods that are used to manipulate data members
- Data members define the properties of the class whereas the member functions define the behavior of the class
- A class can have multiple objects which have properties and behavior that in common for all of them

Syntax:

```
class class_name {
  data_tpye data_name;
```

```
  return_type method_name(parameters);
}
```

Object:

An object is an instance of a class. It is an entity with characteristics and behavior that are used in the object-oriented programming. An object is the entity that is created to allocate memory. A class when defined does not have memory chunk itself which will be allocated as soon as objects are created.

Syntax:

```
class_name object_name;
```

Example:

```
#include<iostream>
using namespace std;
class calculator {
  int number1;
  int number2;
  char symbol;
  public :
  void add() {
    cout<<"The sum is "<<number1 + number2 ;
  }
  void subtract() {
    cout<<"The subtraction is "<<number1 - number2 ;
  }
  void multiply() {
    cout<<"The multiplication is "<<number1 * number2 ;
  }
  void divide() {
    cout<<"The division is "<<number1 / number2 ;
  }
```

```
  calculator (int a , int b , char sym) {
    number1 = a;
    number2 = b;
    symbol = sym;
    switch(symbol){
      case '+' : add();
        break;
      case '-' : add();
        break;
      case '*' : add();
        break;
      case '/' : add();
        break;
      default : cout<<"Wrong operator";
    }
  }
};
int main() {
  calculator c1(12 , 34 , '+');
}
```

Output:

The sum is 46

Encapsulation:

In object-oriented programming, encapsulation is the concept of wrapping together of data and information in a single unit. A formal definition of encapsulation would be: encapsulation is binding together the data and related function that can manipulate the data.

Let’s understand the topic with an easy real-life example,

In our colleges, we have departments for each course like computer science, information tech., electronics, etc. each of these departments have their own students and subjects that are kept track of and being taught. let's think of each department as a class that encapsulates the data about

students of that department and the subjects that are to be taught. Also a department has some fixed rules and guidelines that are to be followed by the students that course like timings, methods used while learning, etc. this is encapsulation in real life, there are data and there are ways to manipulate data.

Due to the concept of encapsulation in object-oriented programming another very important concept is possible; it is data abstraction or Data Hiding. it is possible as encapsulating hides the data at show only the information that is required to be displayed.

Polymorphism:

The name defines polymorphism is multiple forms. which means polymorphism is the ability of object-oriented programming to do some work using multiple forms. The behavior of the method is dependent on the type or the situation in which the method is called.

Let's take a real-life example, A person can have more than one behavior depending upon the situation. like a woman a mother, manager and a daughter and this define her behavior. This is from where the concept of polymorphism came from.

In c++ programming language, polymorphism is achieved using two ways. They are operator overloading and function overloading.

Operator overloading:

In operator overloading and operator can have multiple behavior in different instances of usage.

Function overloading:

Functions with the same name that can-do multiple types based on some condition.

Inheritance: It is the capability of a class to inherit or derive properties or characteristics other class. it is very important and object-oriented program as it allows reusability i.e. using a method defined in another class by using inheritance. The class that derives properties from other class is known as child class or subclass and the class from which the properties are inherited is base class or parent class.

C++ programming language supports the following types of inheritance

- single inheritance

- multiple inheritance
- multi-level inheritance
- Hierarchical inheritance
- hybrid inheritance

Abstraction:

Data abstraction or Data Hiding is the concept of hiding data and showing only relevant data to the final user. It is also an important part object-oriented programing.

let's take real life example to understand concept better, when we ride a bike we only know that pressing the brake will stop the bike and rotating the throttle will accelerate but you don't know how it works and it is also not think we should know that's why this is done from the same as a concept data abstraction.

In C++ programming language write two ways using which we can accomplish data abstraction:

- using class
- using header file

Conditionals and loop:

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

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

Code:

```
#include <iostream>
using namespace std;

int main () {
  // Local variable declaration:
  int x, y = 10;

  x = (y < 10) ? 30 : 40;
  cout << "value of x: " << x << endl;

  return 0;
}
```

Output:

value of x: 40

There are 3 types of loops in C++:

1. **While loop:** while loop can be address as an entry control loop. It is completed in 3 steps.

Variable initialization.(e.g int x=0;)

condition(e.g while(x<=10))

Variable increment or decrement (x++ or x-- or x=x+2)

Syntax:

```
variable initialization;
while (condition)
{
   statements;
   variable increment or decrement;
}
```

2. **For loop:** for loop is used to execute a set of statement repeatedly until a particular condition is satisfied. we can say it an open ended loop. General format is,

Syntax:

```
for(initialization; condition; increment/decrement)
{
   statement-block;
}
```

In for loop we have exactly two semicolons, one after initialization and second after condition. In this loop we can have more than one initialization or increment/decrement, separated using comma operator. for loop can have only one condition.

Nested for loop: We can also have nested for loop, i.e one for loop inside another for loop. Basic syntax is,

```
for(initialization; condition; increment/decrement)
{
   for(initialization; condition; increment/decrement)
   {
      statement;
   }
}
```

3. **Do while loop:**

In some situations it is necessary to execute body of the loop before testing the condition. Such situations can be handled with the help of do-while loop. do statement evaluates the body of the loop first and at the end, the condition is checked using while statement. General format of do-while loop is,

```
do
{
   // a couple of statements
}
while(condition);
```