INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING

Object-Oriented Programming (OOP) is the term used to describe a programming approach based on **objects** and **classes**. The object-oriented paradigm allows us to organise software as a collection of objects that consist of both data and behaviour. This is in contrast to conventional functional programming practice that only loosely connects data and behaviour.

Since the 1980s the word 'object' has appeared in relation to programming languages, with almost all languages developed since 1990 having object-oriented features. Some languages have even had object-oriented features retro-fitted. It is widely accepted that object-oriented programming is the most important and powerful way of creating software.

The object-oriented programming approach encourages:

- Modularisation: where the application can be decomposed into modules.
- Software re-use: where an application can be composed from existing and new modules.

An object-oriented programming language generally supports five main features:

- Classes
- Objects
- Classification
- Polymorphism
- Inheritance

Need for Object oriented Programming

Object-oriented programming scales very well, from the most trivial of problems to the most complex tasks. It provides a form of abstraction that resonates with techniques people use to solve problems in their everyday life. Object-oriented programming was developed because limitations were discovered in earlier approaches to programming. There were two related problems. First, functions have unrestricted access to global data. Second, unrelated functions and data, the basis of the procedural paradigm, provide a poor model of the real world.

Benefits of Object oriented Programming

1. **Simplicity**: Software objects model real world objects, so the complexity is reduced and the program structure is very clear.

2. **Modularity**: Each object forms a separate entity whose internal workings are decoupled from other parts of the system.

3. **Modifiability**: It is easy to make minor changes in the data representation or the procedures in an OO program. Changes inside a class do not affect any other part of a program, since the only public interface that the external world has to a class is through the use of methods.

4. **Extensibility**: adding new features or responding to changing operating environments can be solved by introducing a few new objects and modifying some existing ones.

5. **Maintainability**: objects can be maintained separately, making locating and fixing problems easier.

6. **Re-usability**: objects can be reused in different programs.

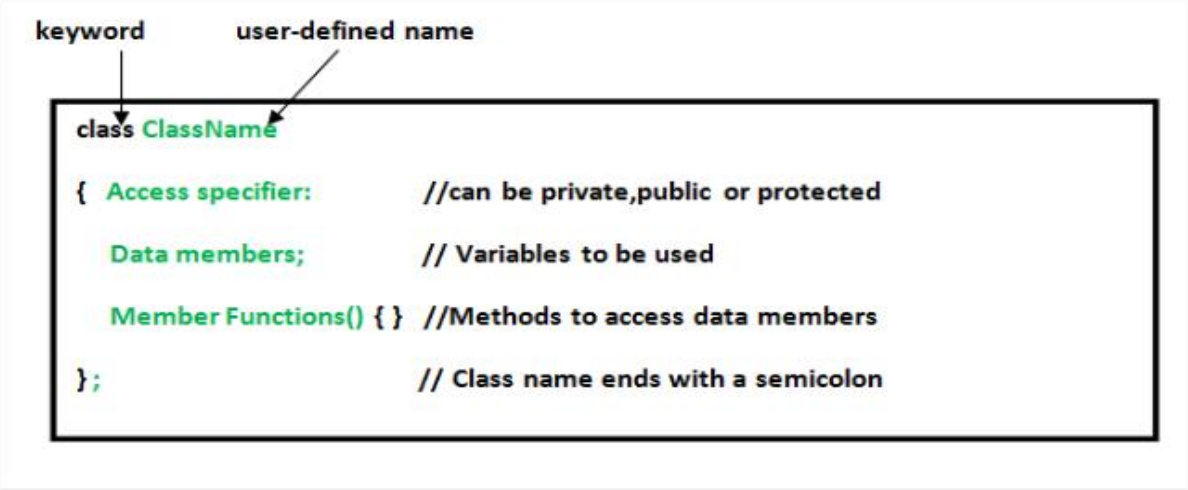
C++ Classes and Objects

Class: The building block of C++ that leads to Object Oriented programming is a **Class**. It is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.
For Example: Consider the Class of **Cars**. There may be many cars with different names and brand but all of them will share some common properties like all of them will have *4 wheels*, *Speed Limit*, *Mileage range* etc. So here, Car is the class and wheels, speed limits, mileage are their properties.

- A Class is a user defined data-type which has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions defines the properties and behavior of the objects in a Class.
- In the above example of class *Car*, the data member will be *speed limit*, *mileage* etc and member functions can be *apply brakes*, *increase speed* etc.

An **Object** is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

A class is defined in C++ using keyword class followed by the name of class. The body of class is defined inside the curly brackets and terminated by a semicolon at the end.



Declaring Objects: When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.

Syntax:

```
ClassName ObjectName;
```

Accessing data members and member functions: The data members and member functions of class can be accessed using the dot('.') operator with the object. For example if the name of object is *obj* and you want to access the member function with the name *printName()* then you will have to write *obj.printName()* .

Accessing Data Members

The public data members are also accessed in the same way given however the private data members are not allowed to be accessed directly by the object. Accessing a data member depends solely on the access control of that data member. This access control is given by Access modifiers in C++. There are three access modifiers: **public, private and protected**.

Example

```
// C++ program to demonstrate
// accessing of data members

#include <iostream>

using namespace std;

class Name{

public:           //Access specifier
    string myname;      //Data Member

    void printname()    // Member function
```

```
    {
    cout<<"My name is:" << myname <<endl;
    }
};

int main()
{
    Name nam1;          //Declaring an object
    nam1.myname = "Emmanuel";       //Accessing data member
    nam1.printname();               //Accessing member function
    return 0;
}
```

```
My name is:Emmanuel

Process returned 0 (0x0)   execution time : 0.149 s
Press any key to continue.
```

Example

A program to print book id

```
#include <iostream>

using namespace std;

class Book{

public:           //Access specifier

    int id;       //Data Member

    void BookMe()             // Member function
    {
    cout<<"The book ID is:" << id <<endl;
    }
};

int main()
{
    Book book1;         //Declaring an object
    book1.id = 200;           //Accessing data member
    book1.BookMe();           //Accessing member function

    return 0;
}
```

Output

```
The book ID is:200

Process returned 0 (0x0)   execution time : 27.855 s
Press any key to continue.
```

Example

Program to print book id and title

```
#include <iostream>

using namespace std;

class Book{

public:           //Access specifier

    int id;       //Data Member
    string title; //Data Member

    void BookMe()             // Member function
    {
    cout<<"The book ID is:" << id <<endl;
    cout<<"The book title is:" << title <<endl;
    }
};

int main()
{
    Book book1;         //Declaring an object
    book1.id = 200;           //Accessing data member
    book1.title = "Database";
    book1.BookMe();           //Accessing member function

    return 0;
}
```

Output

The book ID is: 200
The book title is: Database

Example

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

class Name{

public:               //Access specifier
    string myname;    //Data Member
    int id;           //Data Member
```

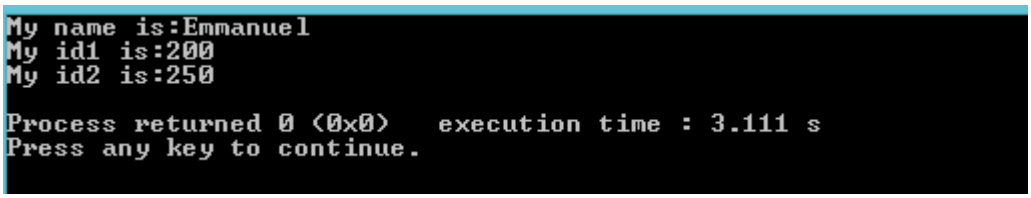
```
    void printname()  // Member function
    {
    cout<<"My name is:" << myname <<endl;
    }

    void printid()          // Member function
    {
    cout<<"My id is:" << id <<endl;
    }
};
int main()
{
    Name nam1;        //Declaring an object
    nam1.myname = "Emmanuel";    //Accessing data member
    nam1.printname();             //Accessing member function
     nam1.id = 200;        //Accessing data member
    nam1.printid();        //Accessing member function
    return 0;
}
```

```
My name is:Emmanuel
My id is:200

Process returned 0 (0x0)   execution time : 1.850 s
Press any key to continue.
```

Example

```
#include <iostream>

using namespace std;

class Name{

public:           //Access specifier

    string myname;      //Data Member

    int id1;       //Data Member

    int id2;       //Data Member

    void printname()    // Member function
    {
    cout<<"My name is:" << myname <<endl;
    }
    void printid1()     // Member function
    {
    cout<<"My id1 is:" << id1 <<endl;
```

```
    }
    void printid2()     // Member function
    {
    cout<<"My id2 is:" << id2 <<endl;
    }
};
int main()
{

    Name nam1;          //Declaring an object

    nam1.myname = "Emmanuel";       //Accessing data member

    nam1.id1 = 200;     //Accessing data member

    nam1.id2 = 250;     //Accessing data member

    nam1.printname();    //Accessing member function

    nam1.printid1();    //Accessing member function

    nam1.printid2();    //Accessing member function

    return 0;

}
```



Member Functions in Classes

There are 2 ways to define a member function:

- Inside class definition
- Outside class definition

To define a member function outside the class definition we have to use the scope resolution : : operator along with class name and function name.

Example

```
// C++ program to demonstrate function
```

```
// declaration outside class


#include <iostream>

using namespace std;

class Name{

public:  //Access specifier

    string myname; //Data Member

    int id1;  //Data Member

    int id2;  //Data Member

    // printname is not defined inside class definition

    void printname();// Member function

    void printid1()// Member function

    {
    cout<<"My id1 is:" << id1 <<endl;
    }
    void printid2()// Member function
    {
    cout<<"My id2 is:" << id2 <<endl;
    }
};
// Definition of printname using scope resolution operator ::
    void Name :: printname()

 {
    cout<<"My name is:" << myname <<endl;
    }
int main()
{
    Name nam1; //Declaring an object
    nam1.myname = "Emmanuel"; //Accessing data member

    nam1.id1 = 200; //Accessing data member

    nam1.id2 = 250; //Accessing data member

    nam1.printname();   //Accessing member function

    nam1.printid1();   //Accessing member function

    nam1.printid2();   //Accessing member function

    return 0;
```

```
}
```

```
My name is:Emmanuel
My id1 is:200
My id2 is:250

Process returned 0 (0x0)   execution time : 3.111 s
Press any key to continue.
```