BATCH

RECURSION

C AND C++

PROGRAMMING MASTERCLASS

AWESH ISLAM
BUET, CSE

C++ Class-01

SHAROARE HOSAN EMON
BME , BUET

আমাদের সবগুলো ক্লাস দেখার জন্য ভিজিট করো

https://www.hsccrackers.com/

SCAN ME

Introduction to C++ & OOP

Basic Structure of a C++ Program

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

int main(){

    return 0;
}
```

Input in C++

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

int main(){
    int a;
    cin>>a;
    int b,c,d;
    cin>>b>>c>>d;
    return 0;
}

```

Output in C++

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

int main(){
    int a;
    cin>>a;
    int b,c,d;
    cin>>b>>c>>d;
    cout<<"A is : "<<a<<endl;
    cout<<"B is : "<<b<<endl;
    cout<<"C is : "<<c<<endl;
    cout<<"D is : "<<d<<endl;
    return 0;
}

```

New Datatypes in C++

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

int main(){
    bool a = true;
    bool b = false;
    if(a) cout<<"a is: "<<a<<endl;
    if(!b) cout<<"b is: "<<b<<endl;

    string s = "Hello World";
    cout<<s<<endl;
}
```

C vs C++

Metrics	C	C++	Java
Programming Paradigm	Procedural language	Object-Oriented Programming (OOP)	Pure Object Oriented
Origin	Based on assembly language	Based on C language	Based on C and C++
Developer	Dennis Ritchie in 1972	Bjarne Stroustrup in 1979	James Gosling in 1991
Translator	Compiler only	Compiler only	Interpreted language (Compiler + interpreter)
Platform Dependency	Platform Dependent	Platform Dependent	Platform Independent
Code execution	Direct	Direct	Executed by JVM (Java Virtual Machine)
Approach	Top-down approach	Bottom-up approach	Bottom-up approach
File generation	.exe files	.exe files	.class files
Pre-processor directives	Support header files (#include, #define)	Supported (#header, #define)	Use Packages (import)
keywords	Support 32 keywords	Supports 63 keywords	50 defined keywords
Datatypes (union, structure)	Supported	Supported	Not supported
Inheritance	No inheritance	Supported	Supported except Multiple inheritance
Overloading	No overloading	Support Function overloading (Polymorphism)	Operator overloading is not supported
Pointers	Supported	Supported	Not supported
Allocation	Use malloc, calloc	Use new, delete	Garbage collector
Exception Handling	Not supported	Supported	Supported
Templates	Not supported	Supported	Not supported
Destructors	No constructor neither destructor	Supported	Not supported
Multithreading/ Interfaces	Not supported	Not supported	Supported
Database connectivity	Not supported	Not supported	Supported
Storage Classes	Supported (auto, extern)	Supported (auto, extern)	Not supported

What is OOP

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

Basically it is a set of rules to write a program that we use for solving a specific problem and the problem is to represent real life objects in code with their attributes and behaviours.

What is OOP

For Example : A Car
What attributes does it have?
1) Color
2) Price
3) Max Speed
4) Max acceleration
5) Model
Etc.

What is OOP

For Example : A Car
What behaviors does it have?
1) Accelerate
2) Open Door
3) Put fuel
4) Show how much fuel left
5) Rotate left and right
Etc.

Car Class Attributes

```
4  class Car{
5      string color;
6      int price;
7      int max_speed;
8  };
```

Car Class Behaviours

```
4  class Car{
5      string color;
6      int price;
7      int max_speed;
8      void accelerate(){
9          cout<<"The car is accelerating"<<endl;
10     }
11     void horn(){
12         cout<<"Beep Beep"<<endl;
13     }
14 };
```

Student Class Attributes

```
16  class Student{
17        int roll;
18        string name;
19        double cgpa;
20  };
```

Student Class Behaviours

```
16  class Student{
17      int roll;
18      string name;
19      double cgpa;
20      void Introduce_Yourself(){
21          cout<<"I am "<<name<<endl;
22          cout<<"My roll no is: "<<roll<<endl;
23      }
24      double show_result(){
25          cout<<"My cgpa is : "<<cgpa<<endl;
26          return cgpa;
27      }
28  };
```

Creating an Instance of a Class

```
16  class Student{
17      int roll;
18      string name;
19      double cgpa;
20      void Introduce_Yourself(){
21          cout<<"I am "<<name<<endl;
22          cout<<"My roll no is: "<<roll<<endl;
23      }
24      double show_result(){
25          cout<<"My cgpa is : "<<cgpa<<endl;
26          return cgpa;
27      }
28  };
29
30  int main(){
31      Student a;
32  }
```

Access Modifiers

There are 3types of access modifiers:

1) Public
2) Private
3) Protected

By default every member is private.

Access Modifiers

There are 3types of access modifiers:

1) Public
2) Private
3) Protected

By default every member is private.

```
30  int main(){
31      Student a;
32      a.name = "Awesh Islam";        'name' is a private member of 'Student'
33  }
```

Access Modifiers

There are 3types of access modifiers:

1) Public
2) Private
3) Protected

By default every member is private.

```
30  int main(){
31      Student a;
32      a.name = "Awesh Islam";        'name' is a private member of 'Student'
33  }
```

Let's Make it Public

```
16  class Student{
17  public:
18      int roll;
19      string name;
20      double cgpa;
21      void Introduce_Yourself(){
22          cout<<"I am "<<name<<endl;
23          cout<<"My roll no is: "<<roll<<endl;
24      }
25      double show_result(){
26          cout<<"My cgpa is : "<<cgpa<<endl;
27          return cgpa;
28      }
29  };
```

Now access the members

```
31  int main(){
32      Student a;
33      a.name = "Awesh Islam";
34      a.roll = 2005054;
35      a.cgpa = 3.75;
36  }
```

Let's Use the functions

```
37          a.Introduce_Yourself();
38          a.show_result();
39      }
40
```

```
I am Awesh Islam
My roll no is: 2005054
My cgpa is : 3.75
Program ended with exit code: 0
```

Default Constructors

```
31  int main(){
32      Student a;
33      a.Introduce_Yourself();
34      a.show_result();
35  }
36
```

```
I am
My roll no is: -2031131802
My cgpa is : 2.122e-314
Program ended with exit code: 0
```

Constructors are functions

1) Must be the same name as the class
2) No return type
3) Public

Constructors

```
22      Student(){
23          roll = 2005054;
24          name = "Awesh Islam";
25          cgpa = 3.75;
26      }
```

```
38  int main(){
39      Student a;
40      a.Introduce_Yourself();
41      a.show_result();
42  }
```

```
I am Awesh Islam
My roll no is: 2005054
My cgpa is : 3.75
Program ended with exit code: 0
```

Parameterised Constructors

```
22      Student(string Name,int Roll,double CGPA){
23          roll = Roll;
24          name = Name;
25          cgpa = CGPA;
26      }
```

How to Declare an Object

```
38  int main(){
39      Student a("Awesh Islam",2005054,3.75);
40      a.Introduce_Yourself();
41      a.show_result();
42
43      Student b("Ibnul Islam Nabil",2005060,3.85);
44      b.Introduce_Yourself();
45      b.show_result();
46  }
```

Player Class for snake and ladder

```cpp
class Player{
public:
    //Attributes
    int id;
    int current_position;
    //Constructor
    Player(int Id){
        id = Id;
        current_position = 0;
    }
    //Methods
    int throw_dice(){
        srand((unsigned)time(0));
        int random = rand() % 6 + 1;
        cout<<"Player: "<<id<<" has thrown: "<<random<<endl;
        return random;
    }
    void move_player(int threw){
        current_position += threw;
        if(current_position > 100){
            current_position -= threw;
            cout<<"Cannot move"<<endl;
        }
    }
    void show_position(){
        cout<<"Current position of player: "<<id<<"is :"<<current_position<<endl;
    }
    bool win(){
        if(current_position == 100){
            cout<<"Player: "<<id<<" has won."<<endl;
            return true;
        }
        return false;
    }
};
```