Lecture 2
Introduction to OOP

Khola Naseem
khola.naseem@uet.edu.pk

Access specifier or Access modifier:

- The access modifiers of C++ are
  - Public
  - Private
  - Protected
- Data hiding is one of the key features of object-oriented programming languages such as C++.
- Data hiding refers to restricting access to data members of a class. This makes it impossible for other functions and classes to manipulate class data.
- However, it is also important to make some member functions and member data accessible so that the hidden data can be manipulated indirectly.
- The access modifiers of C++ allows us to determine which class members are accessible to other classes and functions, and which are not.

Constructor

- A constructor is a special type of member function that is called automatically when an object is created.
- A class constructor provides the opportunity to initialize the new object as it is created and to ensure that member variables contain valid values.
- It is a special kind of function in a class that differs in a few significant respects from an ordinary member function.
  - A class constructor always has the same name as the class.
  - A constructor also does not return a value and therefore has no return type. It is an error to specify a return type for a constructor.
  - Example:

```
class Box {
public:
Box(){
//code
}

};
```

Constructor

- C++ Default Constructor:
- A constructor with no parameters is known as a **default constructor**. In the example, Box() is a default constructor.
- Example:

```
class Box {
    private:
        float length;
        float breadth;
        float height;
    public:
        Box()
        {
         length=5.5;
         breadth=5.7;
         height =9.7;
        }
        double calculateArea(){
            return length * breadth;
        }

        double calculateVolume(){
            return length * breadth * height;
        }

};

int main(){
    // create objects
    Box box1;  // one object

    cout<<"Area of the box is "<<box1.calculateArea()<<endl;
    cout<<"Volume of the box is "<<box1.calculateVolume();

}
```

Output:

```
Area of the box is 31.35
Volume of the box is 304.095
--------------------------------
Process exited after 0.197 seconds with return value 0
Press any key to continue . . .
```

Credit: Khola Naseem

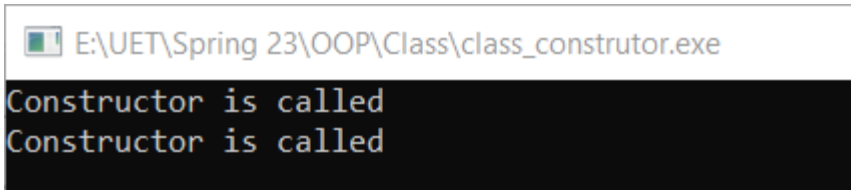
Constructor

- A constructor is a special type of member function that is called automatically when an object is created.
- Example:

```
class Box{
    public:
    Box()
    {
        cout<<"Constructor is called"<<endl;
            }

};
int main(int argc, char** argv) {
    Box b1,b2;
}
```

Output:



Constructor

- C++ Default Default Constructor:
- if you don't define a constructor for a class, the compiler supplies a default default constructor. And no, the two "defaults" is no typo.
- A default default constructor has no parameters and its sole purpose is to allow an object to be created.

```
class Box {
    private:
        float length;
        float breadth;
        float height;
    public:
        double calculateArea(){
            return length * breadth;
        }

        double calculateVolume(){
            return length * breadth * height;
        }

};

int main(){
    // create objects
    Box box1;  // one object
    cout<<"Area of the box is "<<box1.calculateArea()<<endl;
    cout<<"Volume of the box is "<<box1.calculateVolume();

}
```

```
E:\UET\Spring 23\OOP\Class\Classes.exe
Area of the box is 0
Volume of the box is 0
--------------------------------
Process exited after 0.1794 seconds with return value 0
Press any key to continue . . .
```

-

Credit: Khola Naseem

Constructor

- **C++ Parameterized Constructor**
- In C++, a constructor with parameters is known as a parameterized constructor.

```
class Box {
    private:
        float length;
        float breadth;
        float height;
    public:
        Box(float le,float br,float he)
        {
         length=le;
         breadth=br;
         height =he;
        }
        double calculateArea(){
            return length * breadth;
        }

        double calculateVolume(){
            return length * breadth * height;
        }

};

int main(){
    // create objects
    Box box1(4.3,2.2,1.1);  // one object
    cout<<"Area of the box is "<<box1.calculateArea()<<endl;
    cout<<"Volume of the box is "<<box1.calculateVolume();

}
```

- Output:

```
Area of the box is 9.46
Volume of the box is 10.406
--------------------------------
Process exited after 0.09914 seconds with return value 0
Press any key to continue . . .
```

Credit: Khola Naseem

- **C++ Parameterized Constructor**
- In C++, a constructor with parameters is known as a parameterized constructor.

```
class Box {
    private:
        float length;
        float breadth;
        float height;
    public:
        Box(float le,float br,float he)
        {
         length=le;
         breadth=br;
         height =he;
        }
            void print(){
            cout<<"length ="<<length<<endl;
            cout<<"breadth ="<<breadth<<endl;
            cout<<"height ="<<length<<endl;

        }
        double calculateArea(){
            return length * breadth;
        }

        double calculateVolume(){
            return length * breadth * height;
        }
};

int main(){
    // create objects
    Box box1(4.3,2.2,1.1);  // one object
    box1.print();

    cout<<"Area of the box is "<<box1.calculateArea()<<endl;
    cout<<"Volume of the box is "<<box1.calculateVolume();
```

Output:

```
length =4.3
breadth =2.2
height =4.3
Area of the box is 9.46
Volume of the box is 10.406
```

Credit: Khola Naseem

- **C++ Constructor Overloading**
- Overloaded constructors have the same name (name of the class) but the different number of arguments. Depending upon the number and type of arguments passed, the corresponding constructor is called.

```
class Box{
    private:
        float length;
        float breadth;
        float height;
    public:
        Box()
        {
            length=1.1;
            breadth=2.2;
            height=1.0;
        }
        Box(float le,float br)
        {
         length=le;
         breadth=br;
         height =1;
        }
        Box(float le,float br,float he)
        {
         length=le;
         breadth=br;
         height =he;
        }
        double calculateArea(){
            return length * breadth;
        }
};
```

```
int main(){
    // create objects
    Box box1(4.3,2.2,1.1);
    Box box2;
    Box box3(4.1,4.2);
    cout<<"Area of the box1 is "<<box1.calculateArea()<<endl;
    cout<<".............For Box2 .........."<<endl;
    cout<<"Area of the box2 is "<<box2.calculateArea()<<endl;
    cout<<".............For Box3 .........."<<endl;
    cout<<"Area of the box3 is "<<box3.calculateArea()<<endl;
}
```

Output:

```
Area of the box1 is 9.46
.............For Box2 ..........
Area of the box2 is 2.42
.............For Box3 ..........
Area of the box3 is 17.22

--------------------------------
```

Credit: Khola Naseem

destructor

- A destructor works opposite to constructor; it destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically.
- A destructor is defined like constructor. It must have same name as class. But it is prefixed with a tilde sign (~).
- Destructors free up the memory used by the objects the constructor generated.

```
class Box{
    public:
    Box()
    {
        cout<<"Constructor is called"<<endl;
            }
    ~Box()
    {
        cout<<"Destructor is called"<<endl;
            }

};
int main(int argc, char** argv) {
    Box b1,b2;
}
```

```
E:\UET\Spring 23\OOP\Class\class_construtor.exe
Constructor is called
Constructor is called
Destructor is called
Destructor is called

--------------------------------
Process exited after 0.02833 seconds with return value 0
Press any key to continue . . .
```

Credit: Khola Naseem