Lecture 3
Introduction to OOP

Khola Naseem
khola.naseem@uet.edu.pk

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

Constructor

- Parameterized constructor

```
class Box{
    private:
        float height;
        float length;
    public:
    Box(float a,float b)
    {
        height=a;
        length=b;
            }

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

- Error

		E:\UET\Spring 23\OOP\Class\contructor.cpp	In function 'int main(int, char**)':
17	6	E:\UET\Spring 23\OOP\Class\contructor.cpp	[Error] no matching function for call to 'Box::Box()'
17	6	E:\UET\Spring 23\OOP\Class\contructor.cpp	[Note] candidates are:
9	2	E:\UET\Spring 23\OOP\Class\contructor.cpp	[Note] Box::Box(float, float)
9	2	E:\UET\Spring 23\OOP\Class\contructor.cpp	[Note] candidate expects 2 arguments, 0 provided

- Therefore, to avoid such pitfalls, if a class has constructor(s), the class should also include the default constructor

Constructor

- Parameterized constructor
  - Default parameters values

```
class Box{
    private:
        float height;
        float breadth;
    public:

    Box(float a=5,float b=5)
    {
        height=a;
        breadth=b;
    }

    double Area()
    {
        return height*breadth;
    }
};
int main(int argc, char** argv) {
    Box b1;
    cout<<"Area is "<<b1.Area();
}
```

- Output:

```
E:\UET\Spring 23\OOP\Class\class_construtor.exe
Area is 25
--------------------------------
Process exited after 0.1034 seconds with return value 0
Press any key to continue . . .
```

Credit: Khola Naseem

Constructor

- Parameterized constructor
  - Default parameters values

```
class Box{
    private:
        float height;
        float breadth;
    public:

    Box(float a=5,float b=5)
    {
        height=a;
        breadth=b;
    }

    double Area()
    {
        return height*breadth;
    }
};
int main(int argc, char** argv) {
    Box b1;
    Box b2(3);
    cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area();
}
```

- Output:

```
E:\UET\Spring 23\OOP\Class\class_construtor.exe
Area of b1 is 25
Area of b2 is 15
--------------------------------
Process exited after 0.1569 seconds with return value 0
Press any key to continue . . .
```

Credit: Khola Naseem

Constructor

- Parameterized constructor

```
class Box{
    private:
        float height;
        float breadth;
    public:
    Box()
    {
        height=2;
        breadth=2;
    }
    Box(float a=5,float b=5) // call of overloaded 'Box()' is ambiguous
   {
        height=a;
        breadth=b;

    }


    double Area()
    {
        return height*breadth;
    }
};
int main(int argc, char** argv) {
    Box b1;
    Box b2(3);
    cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area();
}
```

issue

	E:\data_UET\Spring 23\OOP\Class\contructor_default ...	**In function 'int main(int, char**)':**
6	E:\data_UET\Spring 23\OOP\Class\contructor_default val...	[Error] call of overloaded 'Box()' is ambiguous
6	E:\data_UET\Spring 23\OOP\Class\contructor_default val...	[Note] candidates are:
2	E:\data_UET\Spring 23\OOP\Class\contructor_default val...	[Note] Box::Box(float, float)
3	E:\data_UET\Spring 23\OOP\Class\contructor_default val...	[Note] Box::Box()
	E:\data_UET\Spring 23\OOP\Class\Makefile.win	recipe for target '"contructor_default values.o"' failed

Credit: Khola Naseem

Constructor

- Parameterized constructor

```
class Box{
    private:
        float height;
        float breadth;
    public:
    Box()
    {
        height=2;
        breadth=2;
    }
    Box(float a=5,float b=5)
   {
        height=a;
        breadth=b;
   }

    Box(float c,float d=5.5)
    {
        height=c;
        breadth=d;

    }
    double Area()
    {
        return height*breadth;
    }
};
int main(int argc, char** argv) {
    //Box b1;
    Box b2(3);
    //cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area();
}
```

issue

21	2	E:\data_UET\Spring 23\OOP\Class\contructor_default val...	[Error] 'Box::Box(float, float)' cannot be overloaded
14	2	E:\data_UET\Spring 23\OOP\Class\contructor_default val...	[Error] with 'Box::Box(float, float)'
28		E:\data_UET\Spring 23\OOP\Class\Makefile.win	recipe for target '"contructor_default values.o"' failed

Credit: Khola Naseem

Copy constructor

- The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.
- If a copy constructor is not defined in a class, the compiler itself defines one.
- If the class has pointer variables and has some dynamic memory allocations, then it is a must to have a copy constructor.
- Definition:
  - classname (classname &obj) { // body of constructor }
- Here, obj is a reference to an object that is being used to initialize another object.
- Types
  - Default copy constructor
  - User defined copy constructor

Credit: Khola Naseem

Default Copy constructor

- Example:

```
class Box{
    private:
        float length;
        float breadth;
    public:
    Box()
    {
        length=2.0;
        breadth=2.0;
    }
    Box(float a,float b)
    {
        length=a;
        breadth=b;
    }

    double Area()
    {
        return length*breadth;
    }

};
int main(int argc, char** argv) {

    Box b1(3.1,8.1);
    //Box b2=b1; //one way
    Box b2(b1); //second way
    //Box b3;
    //b3=b1; //through assignment operator    //Third way

    cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area();
}
```

Output:

```
Area of b1 is 25.11
Area of b2 is 25.11
--------------------------------
```

Credit: Khola Naseem

User Defined Copy constructor

- A user-defined copy constructor is generally needed when an object owns pointers or non-shareable references, such as to a file, in which case a destructor should be written.
- The default constructor does only shallow copy
- Deep copy is possible only with a user-defined copy constructor.
- In a user-defined copy constructor, we make sure that pointers (or references) of copied objects point to new memory locations
- Problem with shallow copy:
  - if variables of the object are defined in the heap section of memory. If variable(s) are dynamically allocated memory , then the copied object variable will also reference the same memory location.
  - Since both objects will reference to the same memory location, then change made by one will reflect those change in another object as well. Since we wanted to create a replica of the object, this purpose will not be filled by Shallow copy.

Default Copy constructor

- Issue:

```
#include <iostream>
using namespace std;
class Box{
    private:
        float length;
        float *breadth;
    public:
    Box(float a,float b)
    {
        length=a;
        breadth=new float;
        *breadth=b;
        cout<<"address of b1 breadth"<<breadth<<endl;
    }
    double Area()
    {
        return length*(*breadth);
    }
    void changevalue()
    {
        *breadth=5.5;

    }
    ~Box()
    {
        delete breadth;
    }
};
int main(int argc, char** argv) {
    Box b1(3.1,8.1);
    Box b2(b1);
    cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area()<<endl;
    b1.changevalue();
    cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area()<<endl;
}
```

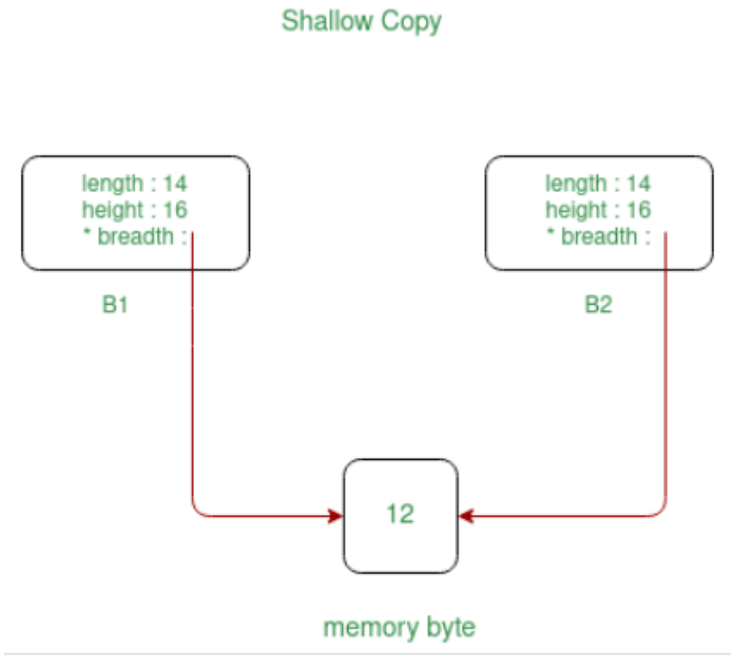
Output:

```
address of b1 breadth0xab1530
Area of b1 is 25.11
Area of b2 is 25.11
Area of b1 is 17.05
Area of b2 is 17.05
```

Credit: Khola Naseem

User Defined Copy constructor

- Problem with shallow copy:
  - if variables of the object are defined in the heap section of memory. If variable(s) are dynamically allocated memory , then the copied object variable will also reference the same memory location.



Credit: Khola Naseem

User Defined Copy constructor

- Example

```
#include <iostream>
using namespace std;
class Box{
    private:
        float length;
        float *breadth;
    public:
    Box(float a,float b)
    {
        length=a;
        breadth=new float;
        *breadth=b;
        cout<<"address of b1 breadth"<<breadth<<endl;
    }
    double Area()
    {
        return length*(*breadth);
    }
    void changevalue()
    {
        length=5.5;

    }
    ~Box()
    {
        delete breadth;
    }
};
int main(int argc, char** argv) {
    Box b1(3.1,8.1);
    Box b2(b1);
    cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area()<<endl;
    b1.changevalue();
    cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area()<<endl;
}
```

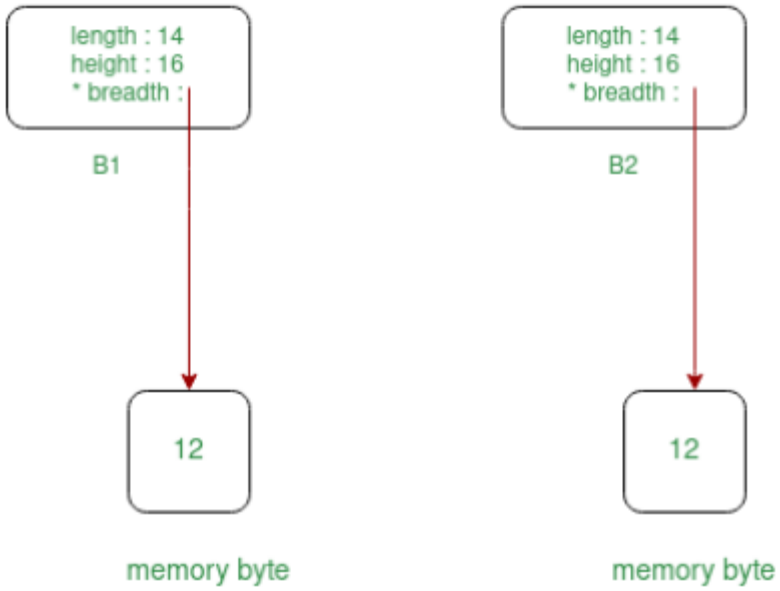
```
E:\data_UET\Spring 2024\OOP A\Class\Default copy.exe
address of b1 breadth0xa71530
Area of b1 is 25.11
Area of b2 is 25.11
Area of b1 is 44.55
Area of b2 is 25.11

--------------------------------
Process exited after 1.238 seconds with return value 3221226356
Press any key to continue . . .
```

Credit: Khola Naseem

User Defined Copy constructor

- **Deep copy**
- In Deep copy, an object is created by copying data of all variables, and it also allocates similar memory resources with the same value to the object. In order to perform Deep copy, we need to explicitly define the copy constructor and assign dynamic memory as well, if required.



Credit: Khola Naseem

User Defined Copy constructor

- Example:

```
class Box{
    private:
        float length;
        float breadth;
    public:
    Box()
    {
        length=2.0;
        breadth=2.0;
    }
    Box(float a,float b)
    {
        length=a;
        breadth=b;
    }
    Box(Box &obj)
    {
        length=obj.length;
        breadth = obj.breadth;

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

};
int main(int argc, char** argv) {

    Box b1(3.1,8.1);
    //Box b2=b1; //one way
    Box b2(b1); //second way

    cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area();
}
```

Output:

```
Area of b1 is 25.11
Area of b2 is 25.11
--------------------------------
```

Credit: Khola Naseem

User Defined Copy constructor

- Example:

```
class Box{
    private:
        float length;
        float *breadth;
    public:
    Box(float a,float b)
    {
        length=a;
        breadth=new float;
        *breadth=b;
        cout<<"address of b1 breadth"<<breadth<<endl;
    }
    Box(Box &obj1)
    {
        length=obj1.length;
        breadth=new float;
        *breadth=*(obj1.breadth);
        cout<<"address of b2 breadth"<<breadth<<endl;
    }
    double Area()
    {
        return length*(*breadth);
    }
    ~Box()
    {
        delete breadth;
    }
};
int main(int argc, char** argv) {
    Box b1(3.1,8.1);
    Box b2(b1);
    cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area();
}
```

Output:

```
address of b1 breadth0xb61920
address of b2 breadth0xb61940
Area of b1 is 25.11
Area of b2 is 25.11
```

Credit: Khola Naseem

User Defined Copy constructor

- Example:

```
class Box{
    private:
        float length;
        float *breadth;
    public:
    Box(float a,float b)
    {
        length=a;
        breadth=new float;
        *breadth=b;
        cout<<"address of b1 breadth"<<breadth<<endl;
    }
    /* Box(Box &obj1)
    {
        length=obj1.length;
        breadth=new float;
        *breadth=*(obj1.breadth);
        cout<<"address of b2 breadth"<<breadth<<endl;
    }*/
    double Area()
    {
        return length*(*breadth);
    }
    void b2_breadth()
    {
    cout<<"address of b2 breadth"<<breadth;
    }
    ~Box()
    {
        delete breadth;
    }
};
int main(int argc, char** argv) {
    Box b1(3.1,8.1);
    Box b2(b1);
    cout<<"Area of b1 is "<<b1.Area()<<endl;
    cout<<"Area of b2 is "<<b2.Area()<<endl;
    b2.b2_breadth();
}
```

Output:

```
address of b1 breadth0xa76a60
Area of b1 is 25.11
Area of b2 is 25.11
address of b2 breadth0xa76a60
--------------------------------
```

Issues

Credit: Khola Naseem