FAST NATIONAL UNIVERSTIY OF COMPUTER AND EMERGING SCIENCES, PESHAWAR

DEPARTMENT OF COMPUTER SCIENCE

CL217 – OBJECT ORIENTED PROGRAMMING LAB

INHERITANCE - III IN C++

LAB MANUAL # 14

INSTRUCTOR: MUHAMMAD ABDULLAH

SEMESTER SPRING 2021

OBJECT ORIENTED PROGRAMMING LANGUAGE

Table of Contents

Inheritance - III in C++

Derived Class Constructors

In inheritance, a constructor of the derived class as well as the constructor functions of the base class are automatically executed when an object of the derived class is created.

Constructor in Single Inheritance without Arguments

The following example explains the concept of execution of constructor functions in single inheritance.

Example:

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

class Parent
{
    public:
    Parent(void)
    {
        cout<<"Constructor of the Parent Class Called"<<endl;
    }
}; // end of Parent class

class Child : public Parent
{
    public:
    Child(void)
    {
        cout<<"Constructor of the Child Class Called"<<endl;
    }
};  // end of child class

int main() {

    Child obj;   // object of the child class


    return 0;
}

/*Sample Output

Constructor of the Parent Class Called
Constructor of the Child Class Called   */
```

In the above program, when an object of the derived class Child is created, the constructor of both the derived class as well the base class are executed.

Constructor in Single Inheritance with Arguments

In case of constructor functions with arguments, the syntax to define constructor of the derived class is different. To execute a constructor of the base class that has arguments through the derived class constructor, the derived class constructor is defined as:

- Write the name of the constructor function of the derived class with parameters.
- Place a colon (:) immediately after this and then write the name of the constructor function of the base class with parameters.

The following example explains the concept of the constructor functions with arguments in single inheritance.

```
class Student
{
    private:
    char name[15];
    char address[25];

    public:
    Student(char nm[], char add[]);
    void show();

}; // end of Student class

class Marks : public Student
{
    private:
    int sub1, sub2, sub3, total;

    public:
    Marks(char nm[], char add[], int a, int b, int c):Student(nm, add);
    void show_detail();

};  // end of Marks class
```

```
#include <iostream>
using namespace std;
#include<string.h>

class Student
{
    private:
    char name[15], address[25];

    public:
    Student(char nm[], char add[])
    {
        strcpy(name,nm);
        strcpy(address, add);

    }
    void show()
    {
        cout<<"Name is: "<<name<<endl;
        cout<<"Address is: "<<address<<endl;
    }
}; // end of Student class

class Marks : public Student
{
    private:
    int sub1, sub2, sub3, total;

    public:
    Marks(char nm[], char add[], int a, int b, int c):Student(nm, add)
    {
        sub1=a;
        sub2=b;
        sub3=c;
        total=a+b+c;
    }
    void show_detail();
};  // end of Marks class

int main() {

    Marks marks("Aiman", "Hangu", 66,77,88);
    marks.show_detail();
    return 0;
}  // end of main() function
```

```
//defining
void Marks:: show_detail()
{
    show();
    cout<<"Marks of 1st subject: "<<sub1<<endl;
    cout<<"Marks of 1st subject: "<<sub2<<endl;
    cout<<"Marks of 1st subject: "<<sub2<<endl;
    cout<<"Total Marks         : "<<total<<endl;
}

/*Sample Output
Name is: Aiman
Address is: Hangu
Marks of 1st subject: 66
Marks of 1st subject: 77
Marks of 1st subject: 88
Total Marks         : 231
*/
```

The compiler automatically calls a base class constructor before executing the derived class constructor. The compiler's default action is to call the default constructor in the base class. You can specify which of several base class constructors should be called during the creation of a derived class object.

```
#include<iostream>
using namespace std;
class Rectangle
{
private :
    float length;
    float width;
public:
    Rectangle ()
    {
        length = 0;
        width = 0;
    }

    Rectangle (float len, float wid)
    {
        length = len;
        width = wid;
    }

    float area()
    {
        return length * width ;
    }
```

```
};

class Box : public Rectangle
{
private :
    float height;
public:
    Box ()
    {
        height = 0;
    }

    Box (float len, float wid, float ht) : Rectangle(len, wid)
    {
        height = ht;
    }

    float volume()
    {
        return area() * height;
    }
};

int main ()
{
    Box bx;
    Box cx(4,8,5);
    cout << bx.volume() << endl;
    cout << cx.volume() << endl;
    return 0;
}
/*
output
0
160
*/
```

Constructors in Multiple Inheritance Without Arguments

When a class is derived from more than one base classes, the constructor of the derived class as well as of all its base classes are executed when an object of the derived class is created.

If the constructor functions have no parameters then first the constructor functions of the base classes are executed and then the constructor functions of the derived class are executed.

A program is given below to explain the execution of the constructor without parameters.

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

class A
{
    public:
    A(void)
    {
        cout<<"Constructor of class A"<<endl;
    }
}; // end of A class

class B
{
    public:
    B(void)
    {
        cout<<"Constructor of class B"<<endl;
    }
}; // end of B class

class C : public A, public B
{
    public:
    C(void)
    {
        cout<<"Constructor of class C"<<endl;
    }
}; // end of C class

int main() {

    C objC;

    return 0;
}
/*
Constructor of class A
Constructor of class B
Constructor of class C
*/
```

In the above program, three classes are defined. The **C** class is publicly derived from two classes **A** and **B**. All the classes have the constructor functions. When an object of the derived class **C** is created, the constructor functions are executed in the following sequence:

- Constructor of the class **A** is executed first.
- Constructor of the class **B** is executed second.
- Constructor of the derived class **C** is executed in the end.

Constructors in Multiple Inheritance with Arguments

To execute constructors of base classes that have arguments through the derived constructor functions, the derived class constructor is defined as:

- Write the constructor function of the derived class with parameters.
- Place a colon (**:**) immediately after this and then write the constructor functions names of the base classes with parameters separated by commas.

An example is given below to explain the constructor functions arguments in multiple inheritance.

```
#include <iostream>
using namespace std;
#include<string.h>

class Student
{
    private:
    char name[15], address[25];

    public:
    Student(char nm[], char add[])
    {
        strcpy(name,nm);
        strcpy(address, add);
    }
    void show()
    {
        cout<<"Name is: "<<name<<endl;
        cout<<"Address is: "<<address<<endl;
    }
}; // end of Student class

class Marks
{
    private:
    int sub1, sub2, sub3, total;

    public:
    Marks(int a, int b, int c)
    {
```

```
        sub1=a;
        sub2=b;
        sub3=c;
        total=a+b+c;
    }

      void show_marks()
      {
      cout<<"Marks of 1st subject: "<<sub1<<endl;
      cout<<"Marks of 1st subject: "<<sub2<<endl;
      cout<<"Marks of 1st subject: "<<sub3<<endl;
      cout<<"Total Marks         : "<<total<<endl;
      }
};  // end of Marks class

class Show : public Student, public Marks
{
      public:
      Show(char nm[], char add[], int s1, int s2, int s3) : Student(nm,
add), Marks(s1, s2, s3)
            {
             show();          // calling show() method of student class
             show_marks();   // calling show_detail() of marks class
            }


};  // end of Show class

int main() {

    Show marks("Amir", "Kohat", 66,99,88);
    return 0;
}  // end of main() function

/*
output
Name is: Amir
Address is: Kohat
Marks of 1st subject: 66
Marks of 1st subject: 99
Marks of 1st subject: 88
Total Marks         : 253 */
```

References

https://beginnersbook.com/2017/08/cpp-data-types/

http://www.cplusplus.com/doc/tutorial/basic_io/

https://www.w3schools.com/cpp/default.asp

https://www.javatpoint.com/cpp-tutorial

https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/?ref=lbp

https://www.programiz.com/

https://ecomputernotes.com/cpp/