POLYMORPHISM

Topic 2 –term2

Polymorphism- definition

- Polymorphism is the ability of two or more objects to respond differently when they are sent the same message.
- The word polymorphism means having many forms
- Also defined as the ability of a message to be displayed in more than one form
- In real life example of polymorphism, a person at the same time can have different characteristics. Like a man at the same time is a son, a brother, an employee. So this same person posses different behaviors in different situations. This is called polymorphism and is considered as one of the important features of object oriented programming.

In C++ polymorphism is divided into two

1. **Compile time polymorphism**

A type of polymorphism that is achieved by two options of overloading- function overloading or Operator overloading

2. **Run time polymorphism**

A type of polymorphism that only works with one option called virtual functions. This is done through functions overriding

Function Overloading and Operator Overloading.

C++ supports OOPs concept Polymorphism which means “many forms”.In C++ we have two types of polymorphism, i.e. Compile-time polymorphism, and Run-time polymorphism.

Compile-time polymorphism is achieved by using an Overloading technique. <u>Overloading simply means giving additional meaning to an entity by keeping its base meaning intact.</u>

C++ supports two types of overloading:

1. Function overloading

2. Operator overloading

Question 1

- With the aid of a hierarchical diagram illustrate polymorphism as applied in C++ (6 Marks)

Function overloading

- When there are multiple functions with the same name but different parameters then these functions are said to be overloaded .
- Parameter list means the data type and sequence of parameter list.
- Functions are overloaded by change in number of arguments for example a variable X can be set to be a single integer parameter, a single double parameter or even have two parameters as illustrated below
- The main advantage of function overloading is to improve code readability and allows code reusability

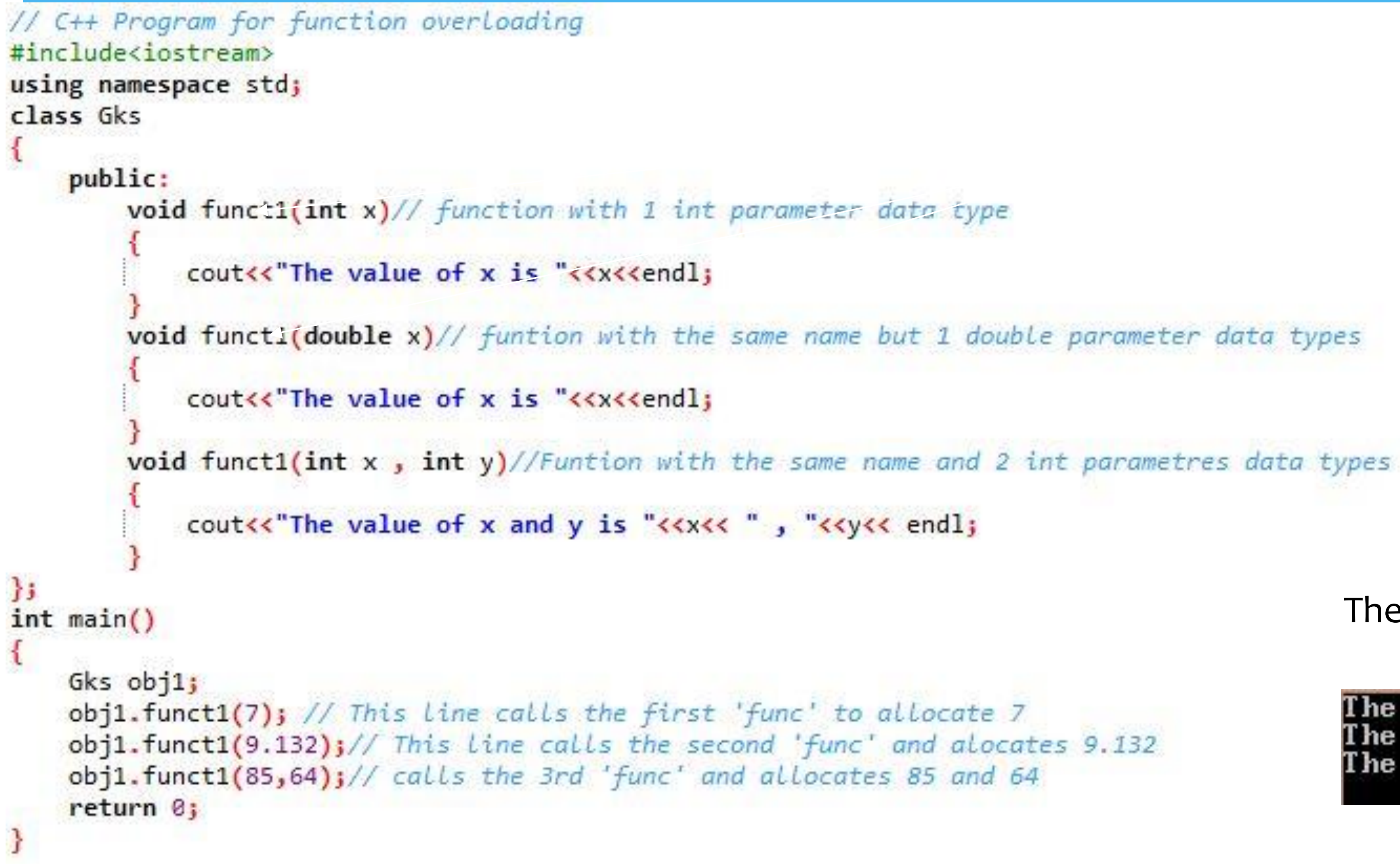
Cont.. **Function Overloading:**

Function overloading is a technique that allows the programmer to have more than one function with the same name but different parameter list.

In other words, we overload the function with different arguments i.e. be it the type of arguments, number of arguments or the order of arguments.

Function overloading is never achieved on its return type.

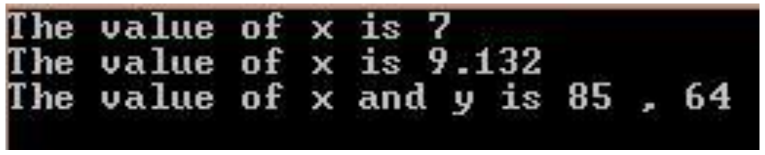
Topic :function overloading example 1



```
// C++ Program for function overloading
#include<iostream>
using namespace std;
class Gks
{
    public:
        void funct1(int x)// function with 1 int parameter data type
        {
            cout<<"The value of x is "<<x<<endl;
        }
        void funct1(double x)// funtion with the same name but 1 double parameter data types
        {
            cout<<"The value of x is "<<x<<endl;
        }
        void funct1(int x , int y)//Funtion with the same name and 2 int parametres data types
        {
            cout<<"The value of x and y is "<<x<< " , "<<y<< endl;
        }
};
int main()
{
    Gks obj1;
    obj1.funct1(7); // This line calls the first 'func' to allocate 7
    obj1.funct1(9.132);// This line calls the second 'func' and alocates 9.132
    obj1.funct1(85,64);// calls the 3rd 'func' and allocates 85 and 64
    return 0;
}
```

The function named Funct1 takes 3 forms here, as integer that's singular, a double and integer of two parameter list

The output of the program



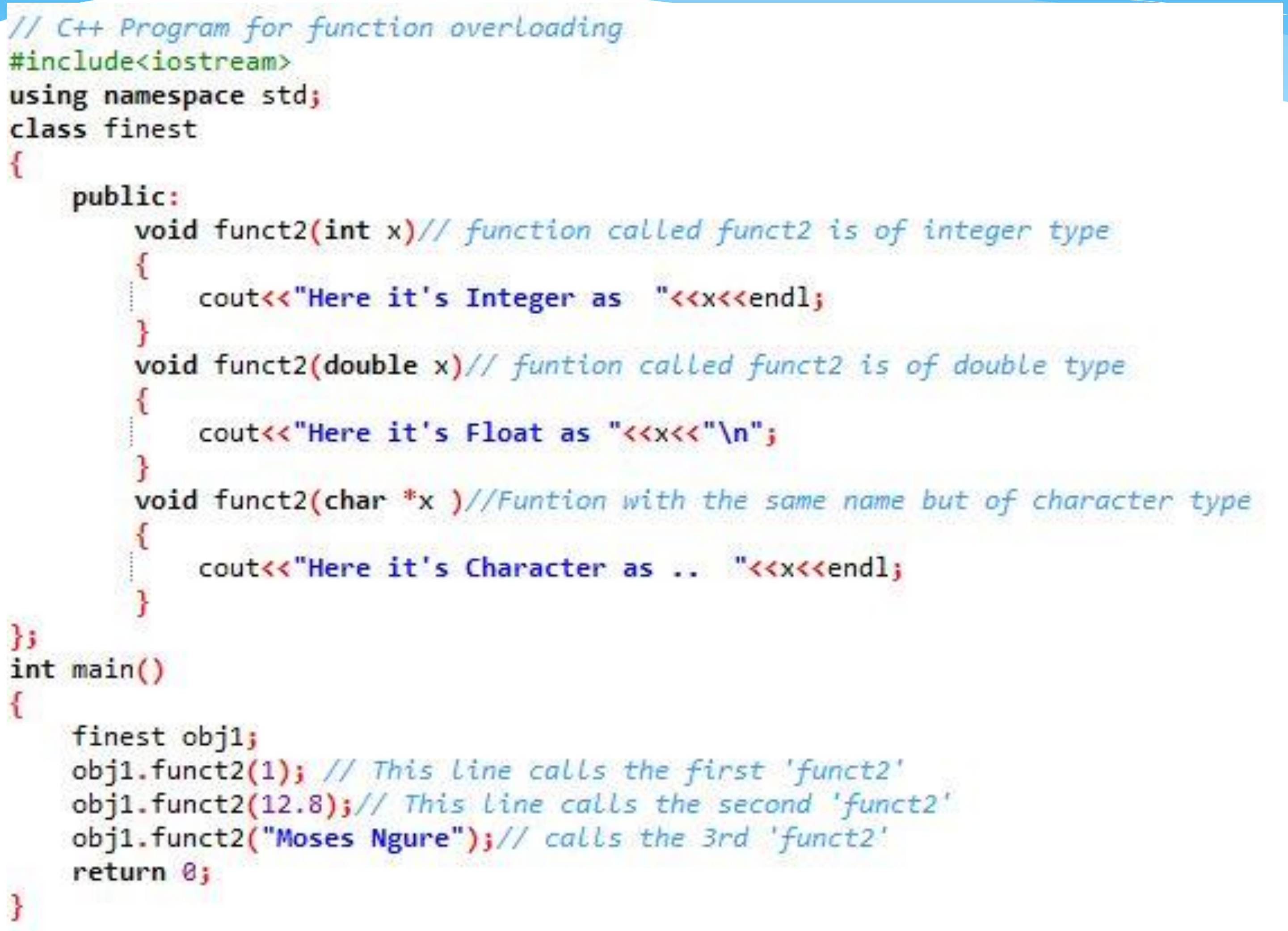
```
The value of x is 7
The value of x is 9.132
The value of x and y is 85 , 64
```

Question 2

Write a C++ program of class "poy" that has a function named "Functionk" that takes three forms as follows

i. As an integer of 73
ii. As a double of 73.936
iii. As an integer of three values 57 ,93 and 22 respectively (6 Marks)

Function overloading example 2



```
// C++ Program for function overloading
#include<iostream>
using namespace std;
class finest
{
    public:
        void funct2(int x)// function called funct2 is of integer type
        {
            cout<<"Here it's Integer as  "<<x<<endl;
        }
        void funct2(double x)// funtion called funct2 is of double type
        {
            cout<<"Here it's Float as "<<x<<"\n";
        }
        void funct2(char *x )//Funtion with the same name but of character type
        {
            cout<<"Here it's Character as ..  "<<x<<endl;
        }
};
int main()
{
    finest obj1;
    obj1.funct2(1); // This line calls the first 'funct2'
    obj1.funct2(12.8);// This line calls the second 'funct2'
    obj1.funct2("Moses Ngure");// calls the 3rd 'funct2'
    return 0;
}
```

The program has The function named Funct2 which takes 3 forms here namely:

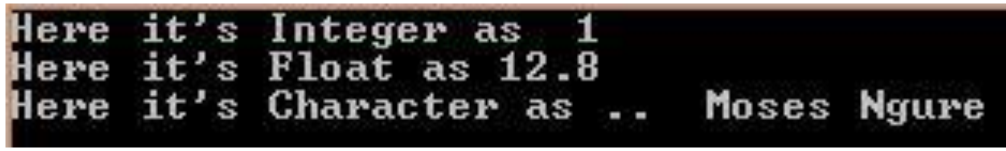
-as single integer x

-as double x

-as character *x

Note: remember to place * for character

The Output of the program



```
Here it's Integer as  1
Here it's Float as 12.8
Here it's Character as ..  Moses Ngure
```

Question 3

Write a C++ program of class “poy2” that has function named “overload2” that takes three forms as follows

i. As an integer of 87
ii. As a double of 87.9
iii. As character ‘Positive Thinking’ (8 Marks)

Reasons that make functions not get overloaded

1. If Function declarations differ in the return type.
2. If any of the functions has a static member function, then it should be declared.
3. If two parameter declarations differ in their default arguments.
4. If parameter declaration differ in that some have a constant and others don’t have or one has a volatile/unstable parameter while the other doesn't.
5. If the Parameter declarations differ in that one becomes a pointer while the other is an array [] with different parameter list
6. If the Parameter declarations that differ in that one is a function type and the other is a pointer to the same function type.

Question 4

- Explain four rules of functions overloading (8 Marks)

Operator Overloading

C++ also provides an option to overload operators, e.g. we can make the operator (+) for joining two strings, this simply adds the 2 and display them as one by having one space in between. We know that this is the addition operator whose task is to add two operands, so a single + when placed between integer operands, it adds them and when placed between two strings it links/joins them to make them one phrase or series

Cont.. **Operator Overloading:**

This is yet another type of compile-time polymorphism that is supported by C++. In operator overloading, an operator is overloaded, so that it can operate on the user-defined types as well with the operands of the standard data type. But while doing this, the standard definition of that operator is kept intact. E.g. an Addition operator (+) that operates on numerical data types can be overloaded to operate on two objects just like an object of complex number class.

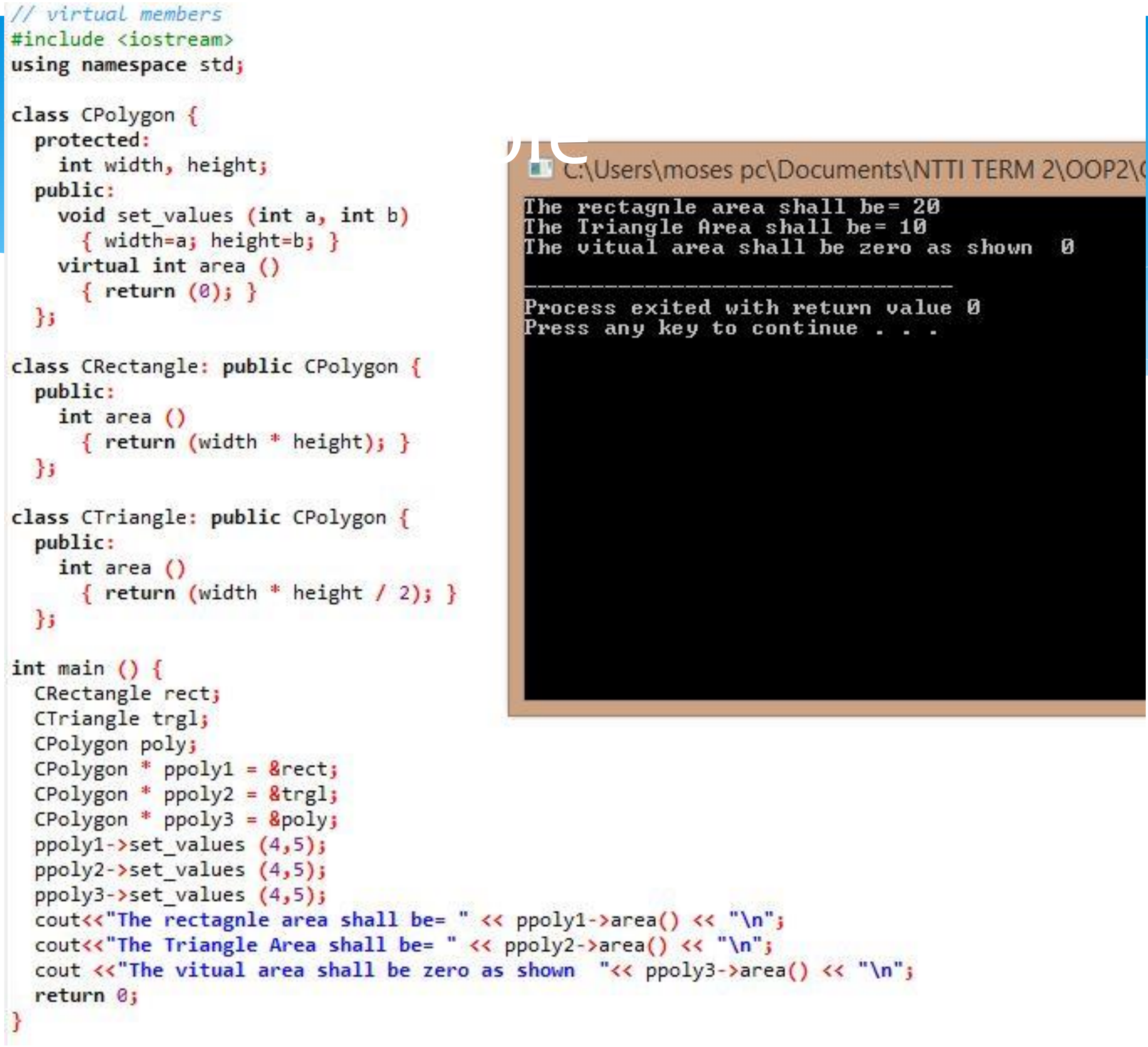
Benefits of Operator Overloading

- By overloading standard operators on a class, we can extend the meaning of these operators, so that they can also operate on the other user-defined objects.
- Function overloading allows us to reduce the complexity of the code and make it more clear and readable as we can have the same function names with different argument lists.

Runtime polymorphism

- **Virtual members**

This is a member of a class that can be redefined in its derived classes . In order to declare a member of a class as virtual, we must do the declaration with the keyword virtual:



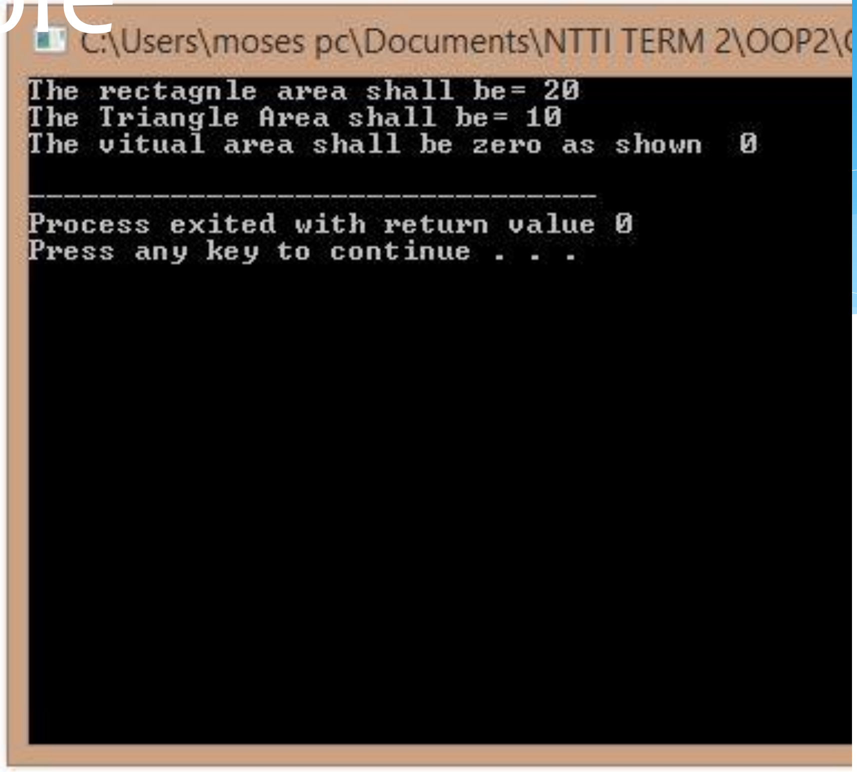
```cpp
// virtual members
#include <iostream>
using namespace std;

class CPolygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      { width=a; height=b; }
    virtual int area ()
      { return (0); }
  };

class CRectangle: public CPolygon {
  public:
    int area ()
      { return (width * height); }
  };

class CTriangle: public CPolygon {
  public:
    int area ()
      { return (width * height / 2); }
  };

int main () {
  CRectangle rect;
  CTriangle trgl;
  CPolygon poly;
  CPolygon * ppoly1 = &rect;
  CPolygon * ppoly2 = &trgl;
  CPolygon * ppoly3 = &poly;
  ppoly1->set_values (4,5);
  ppoly2->set_values (4,5);
  ppoly3->set_values (4,5);
  cout<<"The rectagnle area shall be= " << ppoly1->area() << "\n";
  cout<<"The Triangle Area shall be= " << ppoly2->area() << "\n";
  cout <<"The vitual area shall be zero as shown  "<< ppoly3->area() << "\n";
  return 0;
}
```



C:\Users\moses pc\Documents\NTTI TERM 2\OOP2\

```
The rectagnle area shall be= 20
The Triangle Area shall be= 10
The vitual area shall be zero as shown  0

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

Question 5

- A group of trainers at a TVET institute work in both Registry and ICT department at an allowance of 5,000 per month and bonus of 1500 per term(3 months) respectively. Write a C++ program that creates a base class named Amicus with protected access specifiers for these group of trainers, calculates their three months allowance included in the subclass Registry department and also calculates the end term bonus included in the subclass ICT department (8 Marks)

Coercion and parametric types of polymorphism

Coercion is the automatic application of built in or user defined type conversion e.g when adding an integral and a floating point number, the integral number is promoted automatically to a floating point number by the compiler.

Parametric polymorphism that has an implicit or explicit type parameter which determines the type of the argument for each application of that function

POINTERS

* A pointer is a variable that holds the memory address of another variable.

Advantages of pointers

1. Allows to pass variable, array, functions strings and structures as function arguments
2. A pointer allows to return structured variable from functions
3. Provides functions which can modify their calling arguments
4. It supports dynamic allocation and deallocation of memory segments
   It allows to establish links between data elements for complex data structures such as linked stacks, binary trees and graphs

Cont..

- **Pointer operator** A pointer operator can be represented by a combination of *(asterisk) with a variable
- Example
- Int *ptr; // Declaration
  *Data_type *pointer_variable;*
- Where: Data_type is a type of pointer variable such as integer, character and floating point number variable.
- Pointer_variable is any valid identifier and should be proceeded by the pointer variable.
- Example

- *Float *fpointer;*
  *Char *mpoint1;*

Pointers to base class

- A pointer to a derived class is type-compatible to a pointer to its base class.
- Polymorphism is the art of taking advantage of pointers work, that brings Object Oriented Methodologies to their full potential.

Types of functions in C++

1. Pure virtual functions
2. Recursive functions
3. Inline functions
4. Virtual functions

1. Pure virtual functions

1. The function is assigned zero(0) and generates an abstract base class .A Pure Virtual Member Function is a member function in which the base class forces the derived classes to override.

This class cannot be instantiated and it usually acts as a blueprint that has several sub-classes with further implementation.

e.g.

```
class Shape
 {
public: virtual
void draw() = 0;
 };
```

2. Recursive function

- Recursive function- A function that calls itself

3. Inline functions

- Inline functions enhance execution speed of the program though they demand a more active memory
- It improves the application's performance in exchange for increased compile time
- It is a function which is declared by prefixing it with the keyword inline to the function prototype
- It leads to an increase in the size of the generated binary executables

4. Virtual functions

Virtual function are declared with a body that can <u>be redefined </u>by the derived classes and generates a polymorphic class

It is a function declared by the user to allow algorithms in the base class to be replaced in the derived class, even if users don’t know about the derived class

It is effected by the compiler which ensures the replacement is always called whenever the object in question is actually of the derived class

Reduces the size of the number of program instructions by allowing reusability

Types of operations that can be carried out on a class

1. Selectors (Accessors)- Operations that do not modify the state of an object
2. Modifiers(setters) – Operations that modify the state of an object
3. Conversion Operations – Operations that produce an object of another type based on the value (state) of the object to which they are applied
4. Iterators – Operations that processes data members of objects or objects containing collections of objects

Difference between Call by value & call by reference

Call by Value

* The value of the actual parameters in the calling function do not get affected when the arguments are passed using call by value method
* Actual and formal parameters have different memory locations

Call by reference

* The values of the formal parameters affect the values of actual parameters in the calling function when the arguments are passed using call by reference method.
* The formal parameters are not allocated any memory but they refer to the memory locations of their corresponding actual parameters

Difference between Unary & BINARY Operators

- Unary operators

These are operators are associated with only one operand.

- Binary operators

These are those operators that requires at least two operands.
For example:
+, -, and *

Operators can not be overloaded:

- sizeof – sizeof operator
- . – Dot operator
- .* – dereferencing operator
- -> – member dereferencing operator
- :: – scope resolution operator
- ?: – conditional operator

<u>State the main reason overloading functions are first set as friend function</u>

- So that they can access data members of other classes ,Non-member operator overloading functions are made to be friend function

Restrictions when overloading operators

1. The precedence(preference) of an operator cannot be altered or changed
2. The number of operands that an operator takes cannot be changed
3. Operator functions cannot have default arguments except for the function call
4. A derived class is free to overload any operator
5. Except for the (=) operator, operator functions are inherited by any derived class

Difference between Method Overloading and Method Overriding in C++

Method overloading is having functions with the same name but different argument lists. This is a form of compile-time polymorphism.

Method overriding comes into picture when we rewrite the method that is derived from a base class. Method overriding is used while dealing with run-time polymorphism or virtual functions.

Reasons why arrays are processed with ‘for loop’

Array uses the index to negotiate each of its elements.

If A is an array then each of its elements is accessed as A[i]. Programmatically, all that is required for this to work is an iterative block with a loop variable [i] that serves as an index (counter) incrementing from zero (0) to A.length, cases of decrementing are also possible with the ‘for loop’

Difference between Method Overloading and Method Overriding

Method overloading

- Method overloading is having functions with the same name but different argument lists. This is a form of compile-time polymorphism.

Method overriding

- In Method overriding , we rewrite the method that is derived from a base class. It is used while dealing with run-time polymorphism or virtual functions