KATHMANDU UNIVERSITY

DHULIKHEL, KAVRE

Subject: COMP116
{Object Oriented Programming}
Assignment No: 3

Submitted by:
Ashraya Kadel
CE/UNG
Batch 2022
Rollno: 25

Submitted to:
Rajani Chulyadyo
Department of Computer Science and Engineering

SUBMISSION DATE: 08/10/2023

<Q-1>: What is operator overloading? Why is it important?

Ans:

Operator overloading is a type of polymorphism in which an operator is overloaded to give user defined meaning and to perform operation on user-defined datatype.

The importance of operator overloading are as follows:

i) It helps us to enhanced readability. This helps us to overload an operator to make it meaningful for the class.
ii) It helps us to promote code reusability. The existing operators can be used to operator on custom/ user-defined datatypes.
iii) It helps us to customize our operators and enables to define specific behaviour for variables.
iv) It helps us to make our ~~co~~ custom datatypes compatible with libraries and framework.

<Q-2> What are the different ways through which operator can be overloaded?

Ans:

The different ways through which operator can be overloaded are as follows:

i) using member functions
ii) using friend functions
iii) using normal functions.

i) Using member functions:

During ~~a~~ use of members functions as functions to overload operators, we provide one less argument than the type of operator

Eg: for binary operator, we provide one argument.

Syntax: return_type operator * (parameters);

ii) Using friend functions:

friend functions are functions that can access private data members outside the class. We need to provide the ~~a~~ same numbers of operands as required.

~~Syntax~~ It is defined outside class.

Syntax: friend return-type operator * (parameters);

(iii) Using normal functions

The normal functions, when used for overloading operators, cannot access private members of classes. It uses similar method as friend functions.

<Q.3>: Is it possible to overload the I/O operator using member function? Explain.

Ans

We cannot directly use member functions to overload ~~men~~ the I/O operators. This is because they require an object of the class of istream and ostream to be on the left side of the operator.

Due to this, we use ~~F~~ friend functions for this purpose.

For cout:

~~F~~ → Declaration: friend std::class_name& operator << (std::class_name& outputstream, const datatype& parameter)

Eg: friend std::ostream& operator <<(std::ostream& output stream, const Vector& t);

std::ostream& operator <<(std::ostream& outputstream, const vector& t)
{ return outputstream << t.x << t.y << t.z; }

Similarly for input,

friend std::istream& operator>>(std::istream& inputstream, const Vector & t);

std::istream& operator >> (std::istream& ~~out~~inputstream, const Vector& t)
{ return inputstream >> t.x >> t.y >> t.z; }

<Q.4>: What is the difference between copy constructor and copy assignment operator?

Ans:

Copy constructor	Copy Assignment Operator
creates a new object as a copy of an existing object when it is initialized.	copies the contents of an existing object into another object that has already been initialized
Syntax: class_name (const class_name& other);	Syntax: class_name& operator = (const className & other)
It is called automatically when object is created	It is not called automatically.
It doesn't return a value.	It returns a reference to assigned object.
It doesn't required an existing object.	It requires an existing object.

⟨Q.~~4~~5⟩: What is inheritance? What are the advantages of inheritance?

Ans:

Inheritance is a fundamental concept in object-oriented programming that allows a new class to inherit properties and behaviours from an existing class. The new class is derived class and the existing class is called base or superclass.

The advantages of inheritance are as follows:

i) Code reuseability: This allows us to reduce the code length by letting us to use code from existing classes.

ii) This helps us to extend or specialize the functionality of program promoting modular and extensible code design.

iii) This helps us to organizing classes in our program into a hiesarchial structure making our code more intuitive.

iv) This helps us to make code easies for debugging.

⟨Q.6⟩: Why is the protected access specifier needed?

Ans:

The protected access specifies is needed for the following purpose:

(i): The protected members can be accessed using derived class promoting code reusability. and prevents unauthorized access.

(ii) The members are encapsulated with class and derived class, that helps in data security.

(iii): It helps friend functions access members of class.

(iv) It helps us to balance encapsulation and inheritance.

⟨Q.7⟩: In what order are the class destructors called when a derived class object is destroyed?

Ans:

The class destructors called when a derived class object is destroyed is in the order.

i) Most derived class first

ii) Middle derived class/classes

iii) Base class.

This is in opposite order of constructor.

Eg:
```
#include <iostream>
class A
{ public:
   A() {}
   ~A() {std::cout<< "D A"<<std::endl; };
class B
{ public:
   B() {}
   ~B() {std::cout<< "D B"<<std::endl; };
class C
{ public:
   C() {}
   ~C() {std::cout<<"D C"<<std::endl; };

int main()
{
   A a;
   B b;
   C c;    return 0;
}
```

Output
```
D C
D B
D A
D B
D A
D A
```

<Q.8>: What are the differences between function overloading and function overriding?

Ans:

Function overloading	Function overriding
→ Multiple functions in the same scope with same name but different parameters and return type.	→ occurs in inheritance hierarchy when a sub-class provides specific implementation
→ Compile-time polymorphism	Run-time polymorphism
→ occurs within a single class	involves based derived class
→ Have different returntype and numbers of parameters.	→ Have same return type and numbes of variables
keywords is not applicable.	'virtual' keyword is used
Overloaded methods have different signatures.	~~Ore~~ Over~~written~~ ridden methods has same signature
It doesn't require a base class method.	It requires a base class method.