NOV-27, C++ Assignment 1

1. Explain the abstraction mechanism technique in C++ with an example.

- Data abstraction is one of the most essential and important feature of object oriented programming in C++. Data abstraction refers to providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details. Let's take a real life example of AC, which can be turned ON or OFF, change the temperature, change the mode, and so on. But, we don't know the internal details of the AC, i.e., how it works internally. Similarly, C++ provides different methods to the outside world without giving internal detail about those methods and data.

Some of the abstraction techniques are explain below **:**

- **Abstraction using header files**
  - Header files can also be used as abstraction tachnique. For example, sqrt() function available in <cmath> header file can be used to calculate the square root of a number without actually knowing which algorithm the function uses to calculate the square root. Thus, we can say that header files hides all the implementation details from the user.

Example : Data abstraction using header file

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

int main() {
  int n;
  cout<<"\nEnter a number : ";
  cin>>n;
```

```
  cout <<"\nThe square root of "<<n<<" is "<< sqrt(n) ;
}
```

- **Abstraction using class :**
    - An abstraction can be achieved using classes. A class is used to group all the data members and member functions into a single unit by using the access specifiers. A class has the responsibility to determine which data member is to be visible outside and which is not.

    - **Access Specifiers Implement Abstraction :**

    Public specifier : When the members are declared as public, members can be accessed anywhere from the program.
    Private specifier : When the members are declared as private, members can only be accessed only by the member functions of the class.

2. Differentiate between structure and class in C++. Why is class preferred over structure? Explain.

-The differences between structure and class are listed below :

STRUCTURE	CLASS
It is a value type.	It is a reference type.
Instance of 'structure' is called 'structure variable'.	Instance of a 'class' is called 'object'.
Memory is allocated on Stack.	Memory is allocated on Heap.
If access specifier is not declared, by default all member are 'public'.	If access specifier is not declared, by default all members are 'private'.
It can only have parameterized constructor.	It can have all type of constructor and destrutor.
It doesnot have encapsulation, overloading features.	It supports encapsulation, overloading and other features as well.
Member variables in structure cannot be initialized directly.	Member variables in class can be initialized directly.

Class is preferred over structure because it has overcome most of the limitations of the structure and some of them are given below **:**

a) The most important of point is security. A Structure is not secure and cannot hide its implementation details while a class is secure and can hide its programming and designing details.
b) If visibility (public, private and public) is not specified for the members of the structure, they will be public whereas in class they will be private. Moreover, for inheritance, if we donot specify anything then the struct will inherit publicly from its base class while the class will do private inheritance.

3. What are the various access specifiers used in class? Provide an example demonstrating these specifiers.

- In inheritance, it is important to know when a member function in the base class can be used by the objects of the derived class. This is called accessibility and the access specifiers are used to determine this. These 3 keywords **:** public, protected, and private, are known as access specifiers in C++ inheritance.

a) <u>Public</u> :
Use of Public specifier makes the public members of the base class public in the derived class, and the protected members of the base class remain protected in the derived class.The Private members are directly inaccessible.

Example :

```
#include <iostream>

using namespace std;


class Base {

   private:

    int a = 5;

   protected:

    int b = 10;

   public:

    int c = 15;

    // function to access private member

    int get_a() {    return a;    }

};
```

```
class Child : public Base {
   public:
    // function to access protected member from Base
    int get_b() {       return b;    }
};
int main() {
    Child c1;
    cout << "Private = " <<c1.get_a() << endl;
    cout << "Protected = " <<c1.get_b() << endl;
    cout << "Public = " <<c1.c << endl;
    return 0;
}
```

Output :

Private = 1

Protected = 2

Public = 3

- Here, Child is derived from Base in public mode. So, in Child, c and get_a() is inherited as public, b as protected while a is inaccessible since it is private in Base. Since private and protected members are not directly accessible in main(), we created public functions get_a() and get_b() to access them in main().

-Trying to directly access private and protected data in main() as below will be error :

```
cout << "Private = " << c1.a;
cout << "Protected = " << c1.bt;
```

b) Protected :

- Use of Protected specifier makes the public members of the base class protected in the derived class, and the protected members of the base class remain protected in the derived class.The Private members are directly inaccessible.

Example :

```
#include <iostream>

using namespace std;


class Base {

   private:

    int a = 5;

   protected:

    int b = 10;

   public:

    int c = 15;

    // function to access private member

    int get_a() {    return a;    }

};


class Child : protected Base {

   public:

    // function to access protected member from Base

    int  get_b() {       return b;    }

// function to access public member from Base

    int  get_public_c() {
```

```
        return  c;
    }
};
int main() {
    Child c1;
    cout << " Private cannot be accessed. " << endl;
    cout << "Protected = " <<c1.get_b() << endl;
    cout << "Public = " << get_public_a()<< endl;
    return 0;
}
```

Output :

Private cannot be accessed.

Protected = 2

Public = 3

- Here, Child is derived from Base in protected mode. So, in Child, c, get_a() and b is inherited as protected while a is inaccessible since it is private in Base. As we know, protected members cannot be accessed directly, we cannot use get_a() from Child. So, for the same reason, we created the get_public_c() and get_b() functions in Child in order to access the ‘c’ and ’b’ variable, respectively.

-Trying to directly access private and protected data in main() as below will be error :

```
cout << "Private = " << c1.get_a();
cout << "Public = " << c1.get_c;
```

c) Private

- Use of Private specifier makes the public members and the protected members of the base class private in the derived class.The Private members are directly inaccessible.

Example :

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


class Base {
   private:
    int a = 5;
   protected:
    int b = 10;
   public:
    int c = 15;
    // function to access private member
    int get_a() {    return a;    }
};


class Child : private Base {
   public:
    // function to access protected member from Base
    int  get_b() {       return b;    }
// function to access public member from Base
    int  get_public_c() {
        return  c;
```

```
    }
};
int main() {
    Child c1;
    cout << " Private cannot be accessed. " << endl;
    cout << "Protected = " <<c1.get_b() << endl;
    cout << "Public = " << get_public_a()<< endl;
    return 0;
}
```

Output :

```
Private cannot be accessed.
Protected = 2
Public = 3
```

- Here, Child is derived from Base in private mode. So, in Child, c, get_a() and b is inherited as private while a is inaccessible since it is private in Base. As we know, private members cannot be accessed directly, we cannot use get_a() from Child. So, for the same reason, we created the get_public_c() and get_b() functions in Child in order to access the ‘c’ and ‘b’ variable,respectively.

-Trying to directly access private and protected data in main() as below will be error :

```
cout << "Private = " << c1.get_a();
cout << "Public = " << c1.get_c;
```

4. Illustrate the role of the friend function in OOP with its pros and cons. Also, write a suitable program.

- Data hiding is a fundamental concept of object-oriented programming. It restricts the access of private members from outside of the class. Similarly, protected members can only be accessed by derived classes and are inaccessible from outside. So, this is where friend function plays its role. Friend functions that break this rule and allow us to access member functions from outside the class. A friend function in C++ is a function that is preceded by the keyword “friend”. When the function is declared as a friend, then it can access the private and protected data members of the class.

Pros :

1. It acts as the bridge between two classes by operating on their private data’s.
2. It is able to access members without need of inheriting the class.
3. It can be used to increase the versatility of overloading operator.
4. It provides functions that need data which isn’t normally used by the class.
5. Allows sharing private class information by a non-member function.

Cons :

1. It violates the law of data hiding by allowing access to private members of the class from outside the class.
2. Breach of data integrity.
3. Conceptually messy.
4. Runtime polymorphism in the member cannot be done.
5. Size of memory occupied by objects will be maximum.

Example :

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

class Distance {
   private:
      int meter;

      // friend function
      friend int addFive(Distance);

   public:
      Distance() : meter(0) {}

};

// friend function definition
int addFive(Distance d) {

   //accessing private members from the friend function
   d.meter += 5;
   return d.meter;
}
```

```
int main() {
    Distance D;
    cout << "Distance: " << addFive(D);
    return 0;
}
```

Output :

Distance: 5

5. **Create a class called Person with suitable data members to represent their name and age. Use member functions to initialize and display this information. Derive Student and Employee from the Person class with their unique features. Initialize objects of these classes using constructor and display the information.**

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

class Person{
public :
    string name;
    int age;

    void setValue()
    {
        name="Sangeet Poudel";
        age= 21;
    }
    void displayValue()
    {
        cout<<"\nDIsplaying the information about Person : "<<endl;
        cout<<"\n\tName = "<<name;
        cout<<"\n\tAge  = "<<age;
    }
```

```
};

class Student : public Person{
    int Class;
    string branch;
public :
    Student()
    {
        name="Sudip Bhandari";
        age=21;
        Class=12;
        branch ="Software Engineering";
        cout<<"\n\nDIsplaying the information about Student : "<<endl;
        cout<<"\n\tName   = "<<name;
        cout<<"\n\tAge    = "<<age;
        cout<<"\n\tClass  = "<<Class;
        cout<<"\n\tBranch = "<<branch;
    }
};

class Employee : public Person{
    float salary;
    int experience;
public :
    Employee()
```

```
    {
        name="Arpit Hamal";
        age=22;
        salary=50000;
        experience=2;
        cout<<"\n\nDIsplaying the information about Employee : "<<endl;
        cout<<"\n\tName   = "<<name;
        cout<<"\n\tAge    = "<<age;
        cout<<"\n\tSalary = "<<salary;
        cout<<"\n\tExperience in Work = "<<experience<<" Years ";
    }
};

int main()
{
    Person p1;
    p1.setValue();
    p1.displayValue();

    Student s1;
    Employee e1;
}
```

6. What is data hiding? How do you achieve data hiding in C++? Explain with a suitable program.

- Data hiding is an important feature of object-oriented programming which hides the internal object details (data members). Data hiding ensures exclusive data access to class members and protects the members of a class from an illegal or unauthorized access.

In C++, data hiding is achieved by using access modifiers which includes Private and Protected where Public is also used to indirectly access the hidden data when required.

i. Private : The private keyword is used to create private members (data and functions).Private members/methods can only be accessed within the class they are defined. It cannot be directly accessed outside the class scope. Data is most often defined as private to prevent direct outside access from other classes. Private members can be accessed by members of the class.

ii. Protected : The protected keyword is used to create protected members (data and function).The protected members can be accessed within the class and from the derived class.

iii. Public : The public keyword is used to create public members (data and functions). The public members are accessible from any part of the program.So, public functions are used to give indirect access of the private and protected data members and functions.

Example :

```
#include <iostream>

using namespace std;

class Base {

  private:

   int a = 5;
```

```
   protected:
    int b = 10;
   public:
    int c = 15;
    // function to access private member
    int get_a() {    return a;    }
};


class Child : public Base {
   public:
    // function to access protected member from Base
    int get_b() {       return b;    }
};
int main() {
    Child c1;
    cout << "Private = " <<c1.get_a() << endl;
    cout << "Protected = " <<c1.get_b() << endl;
    cout << "Public = " <<c1.c << endl;
    return 0;
}
```

Here, a is private, b is protected and c is public.Since, Private and protected members cannot be accessed directly, we created public functions get_a() to access the private the data and get_b() to access the protected data. Public data is directly accessed.

7. **Using a class write a program that receives inputs principal amount, time and rate. Keeping rate 8% as the default argument, calculate the simple interest for three customers.**

```
#include<iostream>
using namespace std;
class Interest{

   float principal;
   float rate;
   float time;

public :

   void getData()
   {
      cout<<"\nEnter the Principal Amount : ";
      cin>>principal;
      cout<<"\nEnter the Time in Years : ";
      cin>>time;
      cout<<"\nEnter the Interest Rate : ";
      cin>>rate;
   }
```

```
    void displayData(float rate=8)
    {
        cout<<"\n\n\tPrinciple Amount : "<<"Rs."<<principal<<endl;
        cout<<"\tTime            : "<<time<<" Years "<<endl;
        cout<<"\tInterest Rate    : "<<rate<<" %"<<endl;
        cout<<"\tSimple Interest  : "<<"Rs."<<(principal*time*rate)/100
<<endl;
    }
};

int main()
{
    Interest c1,c2,c3;
    cout<<"\nFor First Customer : "<<endl;
    c1.getData();
    c1.displayData();
    cout<<"\nFor Second Customer : "<<endl;
    c2.getData();
    c2.displayData();
    cout<<"\nFor Third Customer : "<<endl;
    c3.getData();
    c3.displayData();
}
```

8. **Explain how private data and functions are possible to be accessed, clarify with a proper example.**

- Data hiding is a fundamental concept of object-oriented programming. It restricts the access of private members from outside of the class, which means the data members and functions declared as private cannot be directly accessed outside of the class in which they are declared. So one of the way to indirectly access the private members would be by defining a public function within the same class through which the private data members can be accessed by the objects of the class.

Example :

```
#include<iostream>

using namespace std;

class Demo{

private :

   int num=10;

public :

   void displayNum()

{

       cout<<"Num = "<<num;     }

};


int main()  {

   Demo d1;

   d1.displayNum();

 }
```

Here, the data member of class 'Demo' Is declared as private. Due to the private access specifier, the data member cannot be accessed directly by the object of the class in main function. In order to allow the object of the class to access that data member, we defined a method (public function) having the name "displayNum " in the class. This function is used by the object of the class in main function through which the private data member is accessed.

Also, friend functions can also be used to access the private data members and functions.

9. Differentiate between Constructor and Destructor. Can there be more than one destructor in the same class? Explain.

CONSTRUCTOR	DESTRUCTOR
A constructor is a special member in the class that is used to allocate memory to an object.	A destructor is a special member of the class that is used to deallocate memory of an object
A constructor is invoked when the object is created.	A destructor is called when the object is destroyed or deleted.
There can be multiple constructors with a different number of parameters and different types of parameters.	There can be single destructor in the class.
A constructor has the same name as the class name.	A destructor has the same name as the class name with a tilde (~) symbol.
A constructor can be overloaded.	A destructor cannot be overloaded.

- We can have more than 1 constructor since we can overload the constructor which is not possible with Destructors. A destructor doesn't have parameters, so there can be only one. Also, a destructor is used to terminate the instance of the class and release all resources which it is using. So, the instance will not exist when destructor will be called afterwards. Hence, there cannot be more than one destructor in the class.

10. Explain different features of OOP

- Object-oriented programming (OOP) is a high-level computer programming language that implements objects and their associated procedures within the programming context to create software programs. The object-oriented language uses an object-oriented programming technique that binds related data and functions into an object and encourages reuse of these objects within the same and other programs. Some of the features are explained below :

Class and Objects : In object-oriented programming, a class can be defined as a common name to describe the objects of similar characteristics, behaviour and properties.It is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods).

An object is an identifiable entity with some characteristics and behaviour.It is an instance of class.It is an unique entity with unique definition to introduce it.It contains data member and member functions.

Encapsulation : It means that the unnecessary details of an object are hidden from the user but the user can access the essential details which are required at a specific time. So, encapsulation is also known as hiding of information. In object-oriented concepts, encapsulation is required because classes share data and methods between themselves when we design classes. Encapsulation is the most important characteristic that secures the data and the methods of a class.

Abstraction: The concept of abstraction is implemented in object-oriented programming by creating classes. In the class, all the attributes of the objects of the classes are defined. Therefore data cannot be stored in a class because when we create a class it does not allow any space to the class. so to store data, we have to create objects of the class, which have memory allocated as soon as it is created. Similar to encapsulation, abstraction allows you to provide restricted access to the data. Abstraction means to ignore the unnecessary details of an object and accessing essential details. To implement abstraction, we have to use encapsulation features. Encapsulation hides the inappropriate details of an object and abstraction makes only the appropriate details of an object visible.

Inheritance: Inheritance as the name suggests is the concept of inheriting or deriving properties of an existing class to get new class or classes. In other words we may have common features or characteristics that may be needed by number of classes. So those features can be placed in a common class called base class and the other classes which have these characteristics can take the base class and define only the new things that they have on their own in their classes. These classes are called derived class. The main advantage of using this concept of inheritance in Object oriented programming is it helps in reducing the code size, reduces duplication and reusability.

Polymorphism : In object-oriented methodology, polymorphism is the characteristic that facilitates you to assign a special meaning or usage to an entity in different circumstances. The entity can be a variable, method, or an object. In other words, a programmer can use of an entity in a number of different forms without affecting the existing identity of the entity.

Message Passing : Objects can communicate with each other by passing message with each other through arguments. Actually, the arguments are the message passed by one object to another.

Overloading : An very important feature of object oriented programming methodology which extended the handling of data type and operations. Operations overloading, function overloading, constructor overloading etc. are basic concepts of overloading in C++.

Some key features of the Object Oriented programming are:

- Emphasis on data rather than procedure
- Programs are divided into entities known as objects
- Data Structures are designed such that they characterize objects
- Data is hidden and cannot be accessed by external functions