Sandesh Prasai
210328
1/22

ASSINGMENT-1

(1) Define oop as a new paradigm in your own words with suitable example.

Answer

The meaning of paradigm is an example or model. The programing paradigm is a way of conceptualizing what it means to perform computation and how task to be carried out on a computer should be structured and organized. The techniaue used in oop gives an easy way to address problem in everyday life. Object oriented programing (OOP) is a programing paradigm based upon objects (having both data and methods) that aims to incorporate the advantage of modularity and reusablity. In object oriented programing the objects which are instances of classes are used to interact with one another to design application and computer program.. The object oriented program are divided into simple elements refered to as object. Some of the features of object oriented paradigm are:-

(a) Focouses on properties and function rathar than procedure.
(b) Data is hidden from external function.
(c) Functions operate on the properties of an object.
(d) Object may communicate with each other through a function called messaging.
(e) Follow the buttom-up approach in oop design.

Some of the example of oop language are:- c++ Java, c#, Python, Ruby, PHP etc.

CamScanner

Sandesh prasai
210328
2/22

(2) Why oop dominates pop? Mention the characterstics of oop?

Answer:-

There are many ~~ways~~ number of reasons why object oriented programing (oop) dominates procedural oriented programing. One of the reason is that oop is ~~suited~~ better suited for creating large and complex program and application. oop also makes it easier to reuse code which can save time and money. Additionally, oop can help to make code more ~~maintai~~ maintable and understable. The object oriented programing provides data hiding wheres in procedural oriented programing language a global data can be acessed from any where.
- The object oriented programing allows to reuse the code. In object oriented programing the main emphasis is on data rather than procedure (function). which ~~makes it advan~~ is one of the advantage over pop.

The characterstics of object oriented programing are:-

- Emphasis on data rather than procedure (function).
- Program are divided into objects
- function that operate on the data of an objects are tile togenter in data structure.
- Data is hidden and cannot be acessed by external function.
- objects may communicate with each other through functio
- New data and function can be eacily added whenever necessery.

Sandesh prasal
210328
3/22

(3) write short notes ~~not~~ on:-

(a) copping with complexity:-

When computing was in its infacy most programs were written in assemply language by single indivusial and would not be considered large by today's standard. Program become more complex, programer found it difficult to remember all the information they need to know in order to deulop or debug their software which value contains in which register? did a new identifier name conflict with any previously defined name? what variable need to be ~~need~~ initialized before control could be transfred to another section of code.

(b) computation as simulation.

In object oriented programing we never mention memory address, variable, assignments or any of the conventional programing terms. Insteed we speak of objects, message and responsibilities for some action. object oriented programing in Dan Ingall's memoriable phrase:-

Insteed of bit-grinding processor, plundering data structures, we have a universe of well-behaved objects that courteously ask each other to carry out their various desires"

Sandesh prasai
210328
4/22

This is a view of programing as creating a universe in many ways similar to style of computer simulation called "discrete even-driven simulation". In brief a discrete driven simulation the user create computer model of various elements of simulation, describes how they will interact with one another and sets them moving. This is identical to the average object oriented program, In which the user describes what the various entities in the universe for program are and how they will interact with each other and finally sets them in motion. Thus in object oriented programing we have the view that computation is simulation.

(c) Nonlinear behaviour of complexity:-

As programing projects become larger an intresting phenomenan was observed. A ask that will take one programer two month to perform could not be acomplished by two programer working for one month.

In fred Book's memoriable phrase "The bearing of child takes ~~two months~~ nine months, no matter how many women are assigned to ~~work~~ the task.

The reason for this non linear behaviour was complexity in paticular interconnection between software component were complicated and large quantities of information had to be communicated among the various member of programing team.

Sandesh prasad
210328

5| 22 Class and methords:-

(1) Explain various access specifier (or acess mode) used in c++ with an example.

Answer:-

Acess modifier are used to implement an important aspect of object oriented programing known as data hiding. Acess modifier or acess specifier in a class are used to assign the ~~as~~ accessibility to the class member ie they set some restiction on the class member so that they cannot be directly accessed by the outside function. There are three types of access modifier in c++.

(1) Public
(2) Private
(3) Protected.

~~Private~~

Public: All the class member declared under the Public specifier will be available to everyone. The data member and member function declared as public can be accessed by another classes and another function too. The public member of a class can be acessed from anywhare in the program using the direct member acess operator (.) with the object of class.

Sandesh prasai
210328
9/22

Example of public access specifier:-

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

class circle
{
  public:
    double radius;
  double compute_area()
  {
    return 3.14*radius*radius;
  }
};
  int main (void)
  circle cr;
  cr.radius = 7;
      cout<<"Radius is:" <<cr.radius<<endl;
      cout<<"Area is:" <<cr.compute_area();
      return 0;
  }.
```

(2) Private:

The class member declared as private can be accessed only by the member function inside the class. They are not allowed to be accessed directly by any object or function outside the class. Only member function or the friend function are allowed to access the private data member of the class.

Sandesh prasad
210328
7/22

Example to demonstrate private access specifier:-

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

class circle {
  private:
    double radius;
  Public:
    double compute_area() void compute_area(double r)
    {
     radius=r;
     double area= 3.14*radius*radius;
     cout<<"Radius is:"<<radius<<endl;
     cout<<"Area is:"<<area<<endl;
    }
  };
  int main()
  {
   circle c;
   c.compute_area(7.0);
   return 0;
  }
```

Sundesh Prasai
210328
8/22

(3) protected

The protected access modifier is similar to the private access modifier in the same sense that it can't be acessed outside of the class unless with the help of friend class. The difference is that the class member declared as protected can be accessed by any subclass (derived class) of that class as well.

Example to demonstrate protected access modifier.

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

class parent
{
    protected:
    int id_protected;
};
class child : public parent
{
 public:
  void ~~seti~~ setid(int id)
   {
    id_protected=id;
   }
void displayid ()
{
cout<<" id-protected is:" <<id_protected;
```

Sandesh prasai
210328
9/22

```
}
};
int main(void)
{
  Child obj1;
  Obj1.Setid(210328);
  Obj1.displayId();
  return 0;
}
```

Note:- If we donot specify any access modifier for the members inside the class then by default access modifier for the member will be private.

CamScanner

Sandesh prasai
210328
10/22

(2) What is function? Discuss the use of friend function taking in to consideration the concept of data hiding and object oriented programing?

Answer A function is a block of code that perform some operation. A function can optionally define input paramater that can enable caller to pass argument into the function. A function can optionally return a value ~~to~~ as output. Function are ~~use for~~ useful for encapsulating common operations in a single reusable block, ideally with a name that clearly describe what the function does. ~~Exan~~

Example:- The following function accepts two integer from caller and returns their sum; a and b are parameters of type int.

```
int Sum (int a , int b)
{
  return a+b;
}
```

Friend function:

A friend function is a function that is declared as a friend of class. Friend function are not member function of the class. However, they have access to the private and protected member of the class.

CamScanner

Sandesh prasai
210328
11/22

The use of friend function is to acess the private and protected member of the class. Friend function are not the part of classes and they are not inline. This means that they have access to the private and protected member of class. This violates the concept of data hiding which is one of the main principle of object oriented programing. This means the friend function can change the data member without the class knowing.

CamScanner

Sandesh prasai
210328
12/22

(3) What is the difference between defining a member function inside and outside the class? Give suitable example?

Answer:

Defining a member function inside the class:-

→ whenever a member function is defined inside the, the class name ~~are not~~ and scope resolution operato are not ~~required~~ specified in the function ~~body~~ heade
→ More over the member function defined inside the class defination are by default inline functions.
→ They are generally used for small code functions

Example of defining the member function inside the class:

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

class defininginside
{
    int value;
public:
    void input (int n) {
      value=n;
    }
void printvalue()
{
```

Sandesh prasal
210328
13/22

```
cout<<"value is"<<value<< endl;
}.
};
int main(void)
{
    defininginside di;
    di.input(20);
    di.printvalue();
}.
```

Defining the member function outside class:

Defining the member function outside a class require the function decleration (function prototype) to be provided inside the class defination. The member function decleration informs the compiler that the function is the member of the class and that it has been defined outside the class. After a member function is declered inside the class, it must be defined outside the class in the program. The defination of member function outside the class differ from normal function, as the function name in the function header is preceaded by the class name and scope resolution operator (::). The scope resolution operator informs the compailer what class the member belong to.

Syntax;

```
return_type class_name::function_name(paramater list)
{
    //body of member function
}
```

Sandesh prasai
210328
14/22

Example to demonstrate the defination of member function outside the class;

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

class example
{
  int value;
  public:
  void setdata(int n);
  void displaydata();
};
void example:: setdata(int n)
{
  value=n;
}
void example:: displaydata()
{
  cout<<"value is:"<<value<<endl;
}
int main(void)
{
example ex;
ex.setdata(20);
ex.displaydata();
}
```

Sandesh prasai
210328
15/22

(4) Write a c++ program to calculate factorial of a number using recursion.

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

class recursionfactorial
{
  Public:
  int fact(int n)
  {
    int if(n>0)
     {
     return n*(fact(n-1));
     }
    else
      {
       return(1);
       }
  }
};
 int main(void)
 {
  int n;
 cout<<"Enter number to calculate factorial:"<<endl;
  cin>>n;
  recursionfactorial rf;
  cout<<"factorial of number is:"<<rf.fact(n);
 }
```

~~Sandesh 210328~~ Sandesh prasai
210328
16/22

(5) Write the program to add two complex number using the concept of friend function.

```
~~#include<iostream>~~
~~using namespace std;~~
~~class complex1;~~
~~class complex2;~~
~~{~~
#include<iostream>
using namespace std;
class complex2;
class complex1
{
  int real, imaginary;
  Public:
  void setcomplex1(int r, int imag)
  {
    real=r;
    imaginary=imag;
  }
  void displaycomplex1()
  {
  cout <<" Your first complex number in a+ib form
        is"<<real <<" +i"<<imaginary<<endl;
  }
  friend void complexadd(complex1 ; complex2);
};
```

Sandesh prasai
210328
17/22

```
class complex2
{
int real2, imaginary2;
Public:
 void setcomplex(int r, int imag)
 {
 real2 = r;
 imaginary2 = imag;
 }
void displaycomplex2()
{
 cout<<"your second complex number in a+ib form
      is"<< real2<<" +i"<< imaginary2 <<endl;
 }
 friend void complexadd(complex1 complex2);
};
 void complexadd(complex1 c1, complex2 c2)
 {
 int realsum, imagsum;
 realsum = c1.real + c2.real2;
 imagsum = c1.imaginary + c2.imaginary2;
 cout<<"sum of two complex number in a+ib form is"
      <<realsum<<" +i "<<imagsum<<endl;
 }
 int main(void)
 {
  int r, i;
  complex cp1;
```

Sandesh prasai
210328
18/22

```
cout<<"Enter first complex number"<<endl;
cout<<"Enter real part:"<<endl;
cin>>r;
cout<<"Enter imaginary part"<<endl;
cin>>i;
cp1.setcomplex(r,i);
complex cp2;
cout<<"Enter second complex number"<<endl;
cout<<"Enter real part"<<endl;
cin>>r;
cout<<"Enter Imaginary part"<<endl;
cin>>i;
cp2.setcomplex2(r,i);
cp1.displaycomplex1();
cp2.displaycomplex2();
complexadd(cp1, cp2);
}
```

Sandesh prasai
210328
19/22

(6) Write short notes on:

(a) Default arguments:

→ Argument is the data passed in the function. Default argument is useful in situation where some argument have same value. For example bank intrest for all costumer is same.

→ Unlike c, c++ allows us to call a function without specifying all arguments. In such case, the function assignes a default value to parameter which doesn't has the matching argument in function call.

→ Default value are specified when the function is declared

Example program of default argument:-

```
#include<iostream>
using namespace std;
class weight{
  float w;
  Public:
  void process( float m, float g= 9.8)
    {
     w=m*g;
     cout<<"weight="<<w;
    }
};
int main()
```

CamScanner

Sandesh prasai
210328
20/22

```
weight ob;
  ob.process(10, 1.6);
  return 0;
}
```

(b) Inline function:

→ To ~~es~~ eliminate the cost of call of small function, C++ process a new feature called inline function

→ An inline function is expanded inline when the function is called or invoked. i.e the compiler replaces the function call with the corrosponding function code.

→ All inline function must be defined before they are called.

Syntax:

```
inline return-type function-name(argument-list)
{
 function body;
}
```

- Some of the situation where inlined expression may not work are;

→ For function returning value, if a 'loop', 'switch' or goto statement.

→ If the function consist static variable.

→ If the inline function are recursive.

Sandesh prasal
210328
21/22

(c) Static Member functions:

→ A static function can have access to only other static members (function or variables) declared in the same class.

→ A static function can be called using the class name (instead of objects) as follows.

~~Class name::object name~~

Class name :: function name;

Example:-

```
#include <iostream>
using namespace std;
Class test
{
int code;
static int count; // static member variable
Public:
void setcode(void)
{
code =++ count;
}
void showcode(void)
{
cout<<"object member :"<<code<<endl;
}
static void showcount()
{
cout <<" count=" << count<<endl;
```

Sandesh prasai
210328
22/22

```
}
};
int test::count;

int main(void)
{
    test t1, t2;
  t1.setcode();
  t2.setcode();
  test::showcount();
  test t3;
  t3.setcode();
  test::showcount();
  t1.showcode();
  ~~t2.setcode();~~ showcode();
  t3.showcode();
  return 0;
}
```