Assignment -01

OOC

Shahul Hameed. S
1KN18CS097
CSE 'A' Sec

1. Differentiate between [1] POP and OOP.

POP	OOP
* Emphasis on procedure	* Emphasis on data.
* programming task is divided into collection of data structures and functions.	* programming task is divided into tasks.
* procedures are being separated from data being manipulated	* procedures are not separated from data, instead procedure and data are combined together.
* Data is not secure.	* Data is secure.
* A piece of code uses the data to perform the specific task.	* The data uses the piece of code to perform specific task.
* Top down approach is used in the program design.	* Bottom down approach is used in program design.
* debugging is the difficult as the code size increases.	* Debugging is easy even the code size increases.

(2). Structures

* It does not support inheritance.
* Memory is allocated on the stack.
* value type.
* It is used for smaller amount of data
* It may have only parameterized constructor.

classes.

* It supports inheritance.
* Memory is allocated on the heap.
* Reference type
* It is used for larger amount of data.
* It may have all the types of constructors & destructors.

Creating a New data type using structures

Step1:- put the structure definition and prototypes of the associated functions in a header file, as shown.

```
struct data
{
 int d, m, y;
};
void next_day (struct data*);
void get_sys_date (struct dat *);
```

Step 2:- As shown above, put the definition of the associated functions in a source code and create a library.

```
#include "date.h"
void next_day (struct date* p)
 {
   // Calculate the date that immidiately follows the one
   // represented by *p and set it to *p.
 }
```

```
void get_sys_date (struct date *p)
{
  //determine the current system date and set it to *p
}
```

Step 3: provide the header file and the library, in whatever media, to other programmers who want to use this new data type.

Creation of a structure and creation of its associated functions are two separate steps that together constitute one complete process.

(9) Elucidate about reference variables in C++ with appropriate Example. Also write a program in C++ to swap two numbers and display the values before and after swaping.

The OS maintains the address of each variable as it allocates memory for them during runtime. In order to access the value of a variable, the OS first finds the address of the variable and then transfers control to the byte whose address matches that of a variable.

Ex: If 'x' is an existing integer type variable and we want to declare iref as a reference to it,

```
int & iref = x;
```

iref is a reference to 'x'. This means that although 'iref' and 'x' have separate entries in the OS, Their addresses are actually the same!

Thus, a change in the value of 'x' will naturally reflect in iref and vice versa.

Program 1:

```
#include <iostream.h>
using namespace std;
int main()
{
  int a, b, temp;
 cout<<" Enter a number: \n" <<endl;
 cin >> a;
 cout << " Enter another number: \n"<<endl;
  cin >> b;

 cout<<" Numbers before swapping : \n"<<endl;
 Cout<< " a = " << a << "b = " << b << endl;
  temp=a;
  a=b;
  b= temp;

 cout <<" Numbers after swapping : \n" <<endl;
 cout <<" a = "<< a << " b=" <<b<<endl;
  return 0;
}
```

(3) Explain function overloading with example to overload function area to find area of circle, triangle and rectangle.

> C++ allows one or more functions having same name but different arguments lists. The arguments may differ in the type or number or both.
However, the return types of overloaded methods can be the same or different is called function overloading.

```
#include <iostream.h>
#include <stdlib.h>
#define pi 3.14
Class Area
{
public:
  void area(int);
  void area(int, int);
  void area(float, int, int);
};

void Area :: area(int r)
{
cout<<"Area of circle:\n"<<pi*r*r;
}
void Area:: area(int a, int b)
{
cout<<"area of rectangle:\n"<<a*b;
}
void Area :: area(float t, int a, int b)
{
cout<<"Area of triangle:\n"<<t*a*b;
}
void main()
{
  int ch;
  int a, b, r;
  clrscr();
  Area obj;
  cout<<"\n1. Area of circle \n2. Area of rectangle \n
```

```
3. Area of triangle \n 4. Exit !";
cout << " Enter your choice :";
cin >> ch;
Switch (ch)
{
Case 1 : cout << " Enter radius of the circle : ";
cin >> r;
  Obj. Area (r);
  break;
Case 2 : cout << " Enter sides of rectangle :";
        cin >> a >> b;
        Obj. Area (a, b);
        break;
Case 3 : cout << " Enter sides of the triangle !";
        cin >> a >> b;
        Obj. area (0.5, a, b);
        break;
Case 4 : exit (0);
default: cout << " Invalid choice !!! \n";
   }
getch();
}
```

(3) Explain basic concepts of OOC.

> Abstraction : It means using simple things to represent complexity.

Abstraction means simple things like objects, classes and variables represent more complex underlying code

and data.

Encapsulation :- This is the practice of keeping fields within a class private, then providing access to them via public methods. It's a protective barrier that keeps the data and code safe within the class itself.

Inheritance :- this is a special figure of OOP in java. It lets programmers create new classes that share some of the attributes of existing classes.

Polymorphism :- This concept lets programmers use the same word to mean different things in different Context.

(5) Explain function prototyping with example and what is constructor ? list and explain different type of Constructors with example.

A prototype describes the function's interface to the compiler. It tells the compiler the return type of the function as well as the number, type, sequence of its formal arguments.

Syntax:- returntype function_name(argument list);

Ex:-
```
#include <iostream.h>
int add (int, int);

void main()
 {
  int x, y, z;
  Cout << "Enter a number: ";
    Cin >> x;
  Cout << "Enter another number!";
   cin >> y;
```

```
    z = add(x, y);
    cout << z << endl;
}
int add (int a, int b)
{
    return (a+b);
}
```

Constructor: It is a special member function whose task is to initialize the objects of its class.
Its name is the same as the class name.

Types of constructor:

Default constructor:-
A constructor that accepts no parameters is called the default constructor.

```
Ex:- ClockType yourClock;
```

Parameterized Constructor:-
The constructor that can take arguments is called as Parameterized Constructor.

```
Ex:- Class integer {
        int m, n;
    public:
        integer (int x, int y);
    };

integer:: integer (int x, int y)
{
    m = x; n = y;
}
```

Copy constructor :-

a constructor can accept a reference to its own class as a parameter which is called as copy constructor.

Ex:- Integer I2(I1);

This would define an object I2 at the same time initialize it to the value of I1.

(6) How do namespace help in preventing pollution of the global name space?

> name space enables c++ program to prevent pollution of global name space that lead to name clashes.

global namespace refer to the entire source code. It includes all the directly & indirectly included header file.

By default, name of each class is visible in the source code i.e in the global space. This can lead to problem.

Name space used to define a scope where identifiers like variable functions, class etc, are declared. The main purpose of using a namespace is to prevent ambiguity that may occur when 2 identifier have same name.

Syntax :-

```
namespace A1
{
 Class A
};
name space A2
{
 class A
};
```

Consider :

A1.h A2.h

```
 Class A
{
 class A
};
#include "A1.h"
#include "A2.h"
main()
{
A Aobj;
}
```

(7) What are friend functions? Explain in detail and what are static members of a class? Explain.
Write a C++ program to count the number of objects created.

A friend function is a non-member function that has special rights to access private data members of any object of the class of whom it is a friend.
A friend function is prototyped within the definitions of the class of which it is intended to be a friend.

Ex:
```
class A
{
  int x;
  public:
    friend void abc(A&);   //prototype of friend function
};

void abc(A& obj)           //definition of the friend function
{
  A obj.x++;               //accessing private members
}                          //of the object.

void main()
{
  A A1;
  abc(A1);
}
```

Static data members hold global data that is common to all objects of the class.

Ex:
```
class Account
{
  static float interest_rate;   //a static data member.
};
```

Program:- object created

```
#include <iostream.h>
#include <string.h>
using namespace std;
class Message{
        char str[30];
        static int count;
    public:
        Message()
          {
           count ++;
          }
void print message (void)
     {
     cout << str << endl;
     }
static int total objects (void)
     {
      return count;
     }
};
int message :: count = 0;
  int main()
  {
   M1. init Message ("Message one");
   M2. Init Message ("Message two");
   M3. Init Message ("Message three");
  M1. print Message();
   M2. print Message();
   M3. print Message();
cout<<"total objects Created :" << Message :: total object()<<endl;
 return 0; }
```

(8) Explain (i) Inline function (ii) Constant & member function (iii) Mutable data members.

> (i) Inline function() :-

When the program executes the function call instruction the CPU stores the memory address of the instruction following the function call, copies the argument of the function on the stack and finally transfers control to the specific function.

Syntax:- inline return-type function_name()
{
// function code
}

Ex:-
```
#include <iostream.h>
inline double cube(double x)
void main()
{
  double a, b;
  double c=13.0;
  a=cube(5.0)
  b=cube(4.5+7.5);
  cout<<a<<endl<<b<<endl;
  cout<<cube(c++)<<endl;
}
```

(ii) Constant member function() :-

The constant member functions are the functions which are declared as constant in the program. The object called by these functions can't be modified. A constant member function can be called by any type of object. A constant object can be created by prefixing the const

keyword to the object declaration. Any attempt to change the data members of const objects results in a compiler time error.

Eg:
```
#include <iostream.h>
    using namespace std;
    class Test
{   int value;
    Public:
     Test (int v = 0)
     { value = v; }
    int get value ( )
    const {return value;}
    };
int main( )
{
  Test t(20);
  Cout<< t.get value( );
   return 0;
}
```

(iii) Mutable data members:- Mutable data members are those members whose value can be changed in runtime even if the object is of constant type. The keyword mutable is mainly used to allow a particular data member of const object to be modified. when ~~the~~ we declare a function as const, this pointer passed to function becomes const. Adding mutable to a variable allows a const data pointer to change members.

```
#include <iostream.h>
using std::cout;
class Test
  {
   Public:
   int x;
    mutable int y;
    Test()
    {
      x=4; y=10;
    }
};
int main()
  {
  const Test t1;
    t1.y = 20;
  cout << t1.y;
    return 0;
  }
```