Object Oriented Programming using C++

unit - 4th

Method/functions - A repeated group of instructions in a program can be organized as a function.

Every function has the following elements associated with it -

- → Function declaration or prototype
- → Function Parameters (formal Parameters)
- → Combination of function declaration and its definition
- → Function Body
- → Return statement
- → function Call

example - Program for maximum of two integer Numbers.

```
# include <iostream.h>
int max (int x, int y);                    // Prototype
void main ()                               // function caller
{
    int a, b, c;
    cout << " Enter two integer a, b:";
    c= max(a, b);                          // function call
    cout << max(a,b) : " << c<< endl;
}
int max (int x, int y)                     // function call
{ if (x>y)                                 // function Return
    return x;
  else
    return y;                              // function Return
}
```

output - Enter two integer a, b : 20 40
max (a, b) : 20

Inline functions:- C++ provides an alternate way to normal function calls in the form of inline functions. inline functions are those whose function body is inserted in place of the function call statement during the compilation process.

→ The concept of inline function is similar to macro functions of C.

→ An inline function defination is similar to an ordinary function except that the keyword inline precedes the function defination.

Example- WAP in C++ for square of a number using inline functions

```
#include <iostream.h>
inline int sqr( int num)
  { return num*num ;
  }
void main ( )
  { float n;
    cout << " Enter a number :" ;
    cin >> n ;
    cout << " its square = " << sqr(n) <<endl;
  }
```

output:- Enter a Number:5
its square = 25

Constant Member functions – The constant member functions which are declared as constant in the program. The Object called by these functions can not be modified.

→ A constant member function can be called by any type of object.

→ Non- constant functions can be called by non-constant object only.

A constant (const) member function can be declared by using const keyword. It is used when we want a function that should not be used to change the value of the data members.

example– WAP. to demonstrate const member function in C++.

```
#include <iostream>
using namespace std;
class Demo {
    int val;
public:
    Demo(int x = 0)
    { val = x;
    }
    int getvalue() const {
        return val;
    }
};
int main() {
    const Demo d(20);
    Demo d1(8);
    cout << "The value of d:"
         << d.getvalue();
    cout << "The value of d1"
         << d1.getvalue();
    return 0;
}
```

Friend Functions - A friend function access the private & Protected data variables of another class.

To declare a function as a friend of a class, precede the function prototype in the class definition with keyword friend as follows -

```
class Box {
        double width;
        public:
        double length;
        friend void printWidth ( Box box);
        void setWidth (double wid);
                                    };
```

Friend Class :- A friend class is a class that can access the private and protected members of a class in which it is declared as a friend.

To declare all member function of class as friends a class must be declared as friend.

ex - friend class Box

ex -

```
# include <iostream>
using namespace std;
class Temperature
{ int celsius;
  public: Temperature()
          { celcius=0;
          }
  friend int temp(Temperature);
};
int temp (Temperature t) {
  t. celsius = 40;
  return t.celsius; }
int main () {
    Temperature tm;
Cout << "Temperature in celsius"
  << temp (tm) << endl;
  return 0;
}
```

Static Member functions –

A static member function is a special member function, which is used to access only static data members.

→ Any other normal data members can not be accessed through static member functions.

We can access a static member function with class name by using the following syntax –

```
class name :: function name (parameter);
```

To create a static member function we need to use the static keyword while declaring the function.

→ A static ~~data~~ function can only access other static variables or functions in the same class.

Ex

```
#include <iostream>
using namespace std;
class Demo {
    private:
        static int x;                    // static data member
        static int y;
    public:                              // static member functi
        static void Print()
        {
            cout << "value of x:" << x << endl;
            cout << value of Y:" << y << end l;
        }
};
```

```
// static data member initializations
int Demo :: x = 10;
int Demo :: y = 20;

int main()
{
  Demo OB;
  //accessing class name with object name
  cout << " Printing through object name : " << endl;
  OB.Print();
  // accessing class name with class name
  cout << " Printing through class name : " << endl;
  Demo :: Print();
  return 0;
}
```