

Title: Program to implement concepts: a] Inline tunctions b] Friend tunctions c] static tunctions all object as a function argument & returning object.

Aim: To write program to implement a) Inline tunctions

b] Friend tunctions

of statio functions

d) object as a function argument t returning object.

## Theory:

Al Inline tunctions

ett inline tunction is prowerful concept that is commanly used with classes. It a tunction is Inline, the computer places a copy of the code of that tunction at each point where the tunction is called at compile time

Any change to an inline function could require all clients of the function to bu recompiled because compliler would need to replace all the code of that tunction once again otherwise it will continue with old tunctionality.



To incline a tunction, place the keyword inline betore the tunction name and define the tunction betore any calls are made to the tunction. The compiler can ignore the inline qualifier in case defined tunction is more than a time.

A tunction defination in a class

definitions is an inline function defination

even without the use of a inline

specific.

### 3] Friend Functions

A friend tunction of a class is defined outside that class' scope but it has the right to access all private and protected members of the class Even though the prototypes for friend functions appear in the class defination, triends are not member functions.

A friend can be a function tunction template or member function, or a class or class template in which case the entire class and all of its member are friends.

of a class. Precede the function prototype in the class defination with keyword mend as tollows -



# class Box ?

double Width;

Public:

double length;

hiend void printwidth (BOX b); void setwidth (double wid);

## 3] Stutic Functions

By declaring a tunchion member as static, you make it independent of any particular object of the class. A static member tunchion can be called even it no objects of the class exist and the static tunchions are accessed using only the class name and the scope resolution operator ::

only access static data member, other static member tunctions and any other functions from outside the class.

class scope and they do not have access
to the this pointer of the class you
should use a static member trunction
to determine whether some objects of
the class have been created or not.

a) object as an argument and returning

object as an argument.

Object are passed to tunction as argument like variable of any primitive data tupe. There are 2 tupes:

al pass by value b] pass by returned

lass object by value

In this, a copy of object is sent to

the tunction, hence any changes made
to the object inside function do not affect
object to call the function.

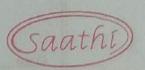
Pass object by returned

In this address of object is passed
to the function, hence any changes
made to object inside tunction is retuch
to actual object.

ii] Returning object

A tunction can also return object either by value or by retemned. When an object is returned by value from a tunction, a temporary object is created whitin the tunction, which holds the return value. This value is turtly

Date \_\_\_ /\_\_\_ /\_\_



assigned to another object in the calling function.

class name tunction name (parameter) {

// body of tunction

(onclusion :-

By doing this practical. I understood about inline, triend & statio tunction and got cleared with concept of each tunction.

#### Program 8 A:

#### Output 8 A:

Enter two numbers

45 46

Sum of 45 and 46 is 91

------

#### Program 8 B:

```
lgnt = l;
bdth = b;
}
friend void Area(rectangle); // declaration of friend function
void perimeter(void) // member function
{
    cout << "Perimeter of Rectangle is " << 2 * (lgnt + bdth) << endl;
}
};

void Area(rectangle R) // defining friend funtion
{
    cout << "Area of Rectangle is " << R.lgnt * R.bdth << endl;
}

int main() // main function
{
    rectangle R(20, 30); // declare object
    R.perimeter(); // calling member function
    Area(R); // calling friend function
    return 0;
}</pre>
```

#### Output 8 B:

Perimeter of Rectangle is 100

Area of Rectangle is 600

\_\_\_\_\_

-----

#### Program 8 C:

```
sqre() {}
    sqre(int l) //constructor
        cout << "Constructor Called " << endl;</pre>
        len = l;
        objCount ++;
    void Area(void)
        cout << "AREA IS " << len * len;</pre>
    static int count(void) // static function
        return objCount;
};
int sqre :: objCount = 0; //Initializing static variable
int main() // main function
    cout << "Initial count " << sqre::count() << endl; // calling satic function</pre>
    sqre A(10);
    sqre B(20);
    sqre C(30);
    cout << "Final count " << sqre::count() << endl; // calling satic function</pre>
    return 0;
```

#### Output 8 C:

Initial count 0

Constructor Called

**Constructor Called** 

Constructor Called

Final count 3

\_\_\_\_\_\_

-----

```
#include <iostream> //header file
using namespace std;
class rectangle
public:
    int lgnt, bdth, area; // declaring variables
    rectangle() {}
    rectangle(int l, int b)
        lgnt = l;
        bdth = b;
    friend rectangle Area(rectangle); // declaration of friend function
    void perimeter(void)
        cout << "Perimeter of Rectangle is " << 2 * (lgnt + bdth) << endl;</pre>
};
rectangle Area(rectangle R) // object as function argument and returning object
    R.area = R.lgnt * R.bdth;
    return R;
int main() // main function
    rectangle R(20, 30), R1; // declare object
    R.perimeter();
    R1 = Area(R); // calling friend function
    cout << "Area of REactagle is " << R1.area << endl;</pre>
    return 0;
```

#### Output 8 D:

Perimeter of Rectangle is 100

Area of Reactagle is 600