

Title: Program to implement concepts:

- a] Inline functions
- b] Friend functions
- c] static functions
- d] object as a function argument & returning object.

Aim: - To write program to implement

- a] Inline functions
- b] Friend functions
- c] static functions
- d] object as a function argument & returning object.

Theory:

A] Inline functions

C++ inline function is powerful concept that is commonly used with classes. If a function is inline, the computer places a copy of the code of that function at each point where the function is called at compile time.

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

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

A function definition in a class definitions is an inline function definition even without the use of a `inline` specifier.

2] Friend Functions

A friend function 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 definition, friends are not member functions.

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

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 b);  
        void setwidth(double wid);  
};
```

3] Static Functions

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

A static member function can only access static data member, other static member functions and any other functions from outside the class.

Static member functions have a class scope and they do not have access to the `this` pointer of the class. You should use a static member function to determine whether some objects of the class have been created or not.

d] object as an argument and returning object.

i] object as an argument.

object are passed to function as argument like variable of any primitive data type. There are 2 types:

a] Pass by value

b] Pass by reference

• Pass object by value

In this, a copy of object is sent to the function, hence any changes made to the object inside function do not affect object to call the function.

• Pass object by reference

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

ii] Returning object

A function can also return object either by value or by reference. When an object is returned by value from a function, a temporary object is created within the function, which holds the return value. This value is further

assigned to another object in the calling function.

```
class_name function_name(parameter) {  
    // body of function  
}
```

Conclusion :-

By doing this practical, I understood about inline, friend & static function and got cleared with concept of each function.

Program 8 A :

```
#include <iostream>           //header file
using namespace std;

inline int sum(int a, int b)  // inline function
{
    return a + b;
}
int main()                    // main function
{
    int a, b;                  // declaring variables

    cout << "Enter two numbers " << endl;
    cin >> a >> b;

    cout << "Sum of " << a << " and " << b
         << " is " << sum(a, b) << endl;    // calling inline function
    return 0;
}
```

Output 8 A :

Enter two numbers

45 46

Sum of 45 and 46 is 91

Program 8 B :

```
#include <iostream> //header file
using namespace std;

class rectangle
{
public:
    int lgnt, bdth;           // declaring variables

    rectangle() {}
    rectangle(int l, int 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 function
{
    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;
}

```

Output 8 B :

Perimeter of Rectangle is 100

Area of Rectangle is 600

Program 8 C :

```

#include <iostream> //header file
using namespace std;

class sqre
{
public:
    int len;          // declaring variable
    static int objCount; // declareing static variable

```

```

sqre() {}
sqre(int L)    //constructor
{
    cout << "Constructor Called " << endl;
    len = L;
    objCount ++;
}
void Area(void)
{
    cout << "AREA IS " << len * len;
}
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

    sqre A(10);
    sqre B(20);
    sqre C(30);

    cout << "Final count " << sqre::count() << endl; // calling satic function
    return 0;
}

```

Output 8 C :

Initial count 0

Constructor Called

Constructor Called

Constructor Called

Final count 3

Program 8 D :

```
#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) // member function
    {
        cout << "Perimeter of Rectangle is " << 2 * (lgnt + bdth) << endl;
    }
};

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(); // calling member function

    R1 = Area(R); // calling friend function
    cout << "Area of R Eactagle is " << R1.area << endl;
    return 0;
}
```

Output 8 D :

Perimeter of Rectangle is 100

Area of Reactagle is 600
