IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 9, September 2024

DOI: 10.17148/IJARCCE.2024.13918

friend Function in C++

Mrs. Suwarna Vijay Nimkarde[1], Mrs. Shobhana Avinash Gaikwad[2]

Lecturer, Computer Technology, BVIT, Navi Mumbai, India[1]

Lecturer, Computer Technology, BVIT, Navi Mumbai, India[2]

Abstract: Friend function is a normal C++ function that can access private members of the class to whom it is declared as friend. Private members of a class cannot be accessed from outside the class. However there could be a situation where more than one classes want to share a particular function.

Keywords: friend, private, protected, public.

I. INTRODUCTION

You do not access private or protected data member of any class, to access private and protected data member of any class you need a friend function. Private members of a class cannot be accessed from outside the class. However there could be a situation where more than one classes want to share a particular function. For example consider two classes 'manager' and 'scientist'.

We can use a function incom_tax ( ) that will operate on the objects of both these classes and calculate the total income tax. Here the function income_tax () need to access private members of both the classes. This can be done with the help of friend function. F

friend function is a normal C++ function that can access private members of the class to whom it is declared as friend. A function can be made a friend function using keyword friend. Any friend function is preceded with friend keyword. The declaration of friend function should be made inside the body of class (can be anywhere inside class either in private or public section) starting with keyword friend.

II. CHARACTERISTICS OF FRIEND FUNCTION

1) friend function must be declared with friend keyword.
2) It can be declared either in the public or private sections of a class without affection its meaning.
3) friend function must be declared in all the classes from which we need to access private or protected members.
4) friend function is declared inside the class .
5) friend function will be defined outside the class without specifying the class name.
6) friend function will be invoked like normal function, without any object.
7) The function definition does not use scope resolution operator.
8) Usually, it has the objects as arguments.
9) Unlike member function, it cannot access the data members directly and has to use an object name and dot (.) operator.

Syntax:

```
class class_name {
 ......
friend returntype function_name(arguments);
}
```

Example: 1

```
class sum
{
int a,b,c;
```

DOI: 10.17148/IJARCCE.2024.13918

```
    public:
    void get()
    {
    cout<<"Enter a and b:";
    cin>>a>>b;
    }
    friend void add(sum x);

};
    void add(sum x)
    {
    x.c=x.a+x.b;
    cout<<"sum="<<x.c;
    }
    void main()
    {
    clrscr();
    sum x;
    x.get();
    add(x);
    getch();
}
```

Example: 2

```
    class employee
     {
     private:
    friend void sal();
    };
    void sal()
    {
     int salary=4000;
     cout<<"Salary: "<<salary;
}
    void main()
    {
    employee e;
    sal();
    getch();
     }
```

In the above example the private members are accessible outside the class using friend function.

III. TWO CLASSES IS HAVING SAME FRIEND

A non-member function may have friendship with one or more classes. When a function has declared to have with more than one class, the friend classes should have forward declaration. It implies that it needs to access the private members of both classes.

Syntax:

```
    class second;
    class first
    {
    private:
    //data members;
    public:
    friend return-type fname(first,second);
};
```

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 9, September 2024
DOI: 10.17148/IJARCCE.2024.13918

```
    class second
    {
    private:
    //data members;
    public:
    friend return-type fname(first,second);
};
```

Example:

```
    class swap2;
    class swap1
    {
    int a;
    public:
    void get()
    {
    cout<<"Enter a:";
    cin>>a;
    }
    friend void swap(swap1,swap2);

    };
    class swap2
    {
    int b;
    public:
    void get()
    {
    cout<<"Enter b:";
    cin>>b;
    }
    friend void swap(swap1,swap2);

    };
    void swap(swap1 p,swap2 q)
    {
    int temp;
    temp=p.a;

    p.a=q.b;
    q.b=temp;

    cout<<"a="<<p.a<<endl;
    cout<<"b="<<q.b<<endl;
    }
    void main()
    {
    clrscr();
    swap1 p;
    swap2 q;
    p.get();
    q.get();
    swap(p,q);
    getch();
    }
```

DOI: 10.17148/IJARCCE.2024.13918

IV. CONCLUSION

By using friend functions, you can maintain encapsulation while providing necessary access to certain functions that require it. friend functions are often used for operator overloading or when two or more classes need to work closely together.

REFERENCES

[1]. D Ravichandran, Programming with C++, McGraw-Hill Education ISBN-10: 0070681899, ISBN- 13: 978-0070681897.

[2]. Stroustrup B., The C++ Programming Language, Pearson Education New Delhi ISBN-10: 0275967301, ISBN-13: 978-0275967307.

[3]. Robert Lafore, Object Oriented Programming in C++, Pearson Education India ISBN-10: 8131722821, ISBN- 13: 978-8131722824.