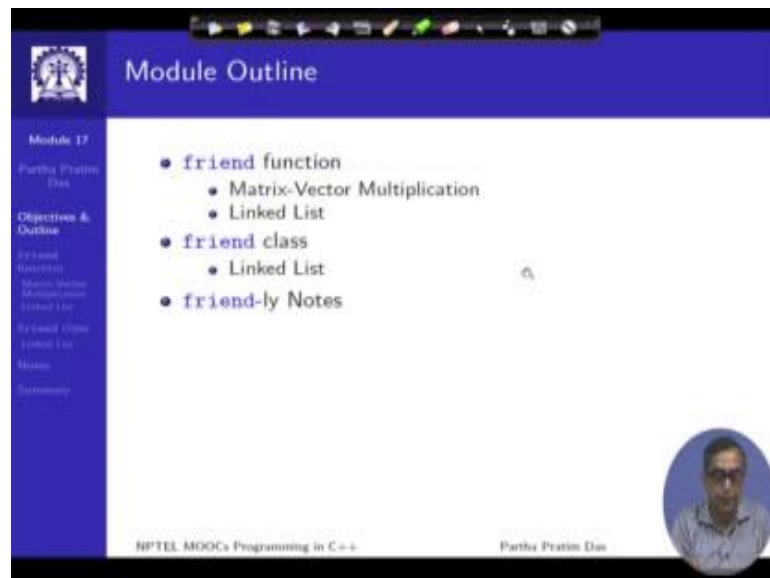


Programming in C++
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 32
friend Function and friend Class

Welcome to module 17 of Programming in C++. In this module, we will talk about friend Function and friend Class trying to understand what do they mean and why are they useful in C++ design process.

(Refer Slide Time: 00:38)



The screenshot shows a presentation slide titled "Module Outline" for "Module 17". The slide content is as follows:

- friend function
 - Matrix-Vector Multiplication
 - Linked List
- friend class
 - Linked List
- friend-ly Notes

At the bottom of the slide, there is a small circular video inset of Prof. Partha Pratim Das. The footer of the slide reads "NPTEL MOOCs Programming in C++" and "Partha Pratim Das".

These will be the module outline we will take examples of Matrix Vector Multiplication and Linked List, and finally will end with some notes. As you know as usual the outline will be available on the left of your slide

(Refer Slide Time: 00:57)

Program 17.01: friend function - Basic Notion

Ordinary function	friend function
<pre>#include<iostream> using namespace std; class MyClass { int data; public: MyClass(int i) : data(i) {} void display(const MyClass& a); }; void display(const MyClass& a) { cout << "data = " << a.data; // Error 1 } int main() { MyClass obj(10); display(obj); return 0; }</pre>	<pre>#include<iostream> using namespace std; class MyClass { int data; public: MyClass(int i) : data(i) {} friend void display(const MyClass& a); }; void display(const MyClass& a) { cout << "data = " << a.data; // Okay } int main() { MyClass obj(10); display(obj); return 0; }</pre>
<ul style="list-style-type: none">• display() is a non-member function• Error 1: 'MyClass::data: ' cannot access private member declared in class 'MyClass'	<ul style="list-style-type: none">• display() is a non-member function friend to class MyClass• Able to access data, even though in class MyClass• Output: data = 10

NPTEL MOOCs Programming in C++ Partha Pratim Das

So, let us first introduce the basic notion of friend function. On the left you have the view of ordinary function. So the situation is like this, that I have a class myclass which has some private data, it has a constructor, and it has; I am sorry if just ignore this line. I have this function written outside of this class which is trying to do something with the class. What is it doing? It takes an object of the class by reference parameter call by reference, and then it tries to print the data element component of that objective.

Now, what do we know, this is private therefore, if I have a function outside I do not have the rights to access this object. So, just consider this is not there. This will give rise to an error and you will get an error of this kind cannot access the private member declared in myclass, because the reason you know very well by now is this is a global function and this is a private data so you cannot access this data directly.

Now, let us look into the right hand side. In the right hand side, we have exactly the same code except that we have introduced this in the class, much in the way we define the member function but with the difference that this is this inclusion is prefixed by a keyword friend. Now this does not make display a member function of the class, display is not a member function of the class, it continues to be a global function. A special type of a global function which is friend of myclass, but with this included when I say that

this global function display which takes an object a of myclass by reference and returns void, when I say that that is a friend of myclass then this does not remain an error this error disappears, this is now allowed.

So the concept is like this as if you know in our home also if somebody would step in we would normally allow them only to the drawing room, which is kind of the public place and we will not allow them to the inner rooms which is our private space. But if I have a friend stepping in then I will probably not keep him waiting in the drawing room rather I will take him to my inner rooms my bed room or kitchen and so on. So it is like a similar concept. So here, he has saying that this function the myclass is saying that the display function which is outside of this class which is not a member function is still a friend and therefore, the private data of this class will be exposed to this member. That is a basic idea of a friend function and therefore it will generate like this will give a compilation error but this will be a right, output will be produced.

(Refer Slide Time: 04:59)

friend function

- A **friend** function of a class
 - has access to the private and protected members of the class (breaks the encapsulation)
 - must have its prototype included within the scope of the class prefixed with the keyword **friend**
 - does not have its name qualified with the class scope
 - is not called with an invoking object of the class
 - can be declared **friend** in more than one classes
- A **friend** function can be a
 - global function
 - a member function of a class
 - a function template

NPTEL MOOCs Programming in C++ Partha Pratim Das 5

With this let us see a formal definition of what a friend function is, “A friend function of a class has access to the private and protected members of the class.” That is it can break the encapsulation which have still not discussed what are protected members but, hold that for a while when we discussed that this will become clear but they are pretty much

like that private members only. So it must have a prototype included in the scope of the class and must be prefixed with friend, we have just in the example of the display function as to how do you write the signature of that function within the class with a friend in the front to say that this particular function is a friend of this class. Now this function does not belong to the class is not a member function therefore its name is not qualified by the class name as we do for normal non friend member functions.

It is not called with invoking an object because it is not a part of the class. A particular member function or a particular function can be friend of more than one class. Now, if you look at what all can be friend function; any global function could be a friend of a class, any member function of a different class could also be a friend of a class, or a function template could be a friend function. Now, function template again you do not know, but when you come to that you will understand how that works.

(Refer Slide Time: 06:37)

```

#include <iostream>
using namespace std;

class Matrix; // Forward declaration
class Vector { int n; int a[];
public:
    Vector(int n) : n(n) {
        // Arbitrary initialization
        for (int i = 0; i < n; ++i)
            a[i] = i + 1;
    }
    void Clear() { // Set a zero vector
        for (int i = 0; i < n; ++i)
            a[i] = 0;
    }
    void Show() { //Show the vector
        for (int i = 0; i < n; ++i)
            cout << a[i] << " ";
        cout << endl << endl;
    }
    friend Vector Prod(Matrix *pM,
                       Vector *pV);
};

class Matrix { int n, m; int a[][];
public:
    Matrix(int n, int m) : n(n), m(m) {
        // Arbitrary initialization
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < m; ++j)
                a[i][j] = i + j;
    }
    void Show() { //Show the matrix
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < m; ++j)
                cout << a[i][j] << " ";
            cout << endl;
        }
    }
    friend Vector Prod(Matrix *pM,
                       Vector *pV);
};

Vector Prod(Matrix *pM, Vector *pV) {
    Vector v(pM->n); v.Clear();
    for (int i = 0; i < pM->n; ++i)
        for (int j = 0; j < pM->m; ++j)
            v.a[i] += pM->a[i][j] * pV->a[j];
    return v;
}
    
```

* Vector Prod(Matrix*, Vector*) is a global function
 * Vector Prod(Matrix*, Vector*) is friend of class Vector as well as class Matrix
 * This function accesses the private data members of both these classes

NPTEL MOOCs Programming in C++ Partha Pratim Das

With that, now let me introduce a more concrete problem. Here the problem is I have two classes, I have a vector class which keeps a linear vector of a size n, and have a matrix class which keeps a square matrix, not necessarily square matrix it is a keeps a matrix of m by n dimension. So, this is constructor for the vector just to keep things simple so that I do not have to enter the values of the vector here I have set as if the constructor itself

set certain values. Similarly, for the constructor of matrix as if when he construct the values are automatically initialized. I mean what values exist in that matrix or vector is not for our interest. What is of our interest is? Now given a vector object and a matrix object I want to define a function which will be able to compute the product of them. So I want a function like this.

A function prod which takes a matrix object here, I have taken it by a pointer takes a vector object multiplies them, of course following the rules of matrix vector multiplication and gives me the result which will of course in this case be a vector. What you see here is, this particular function prod and this is implementation I am not going through the implementation you can check it at a later point of time this is a typical matrix vector multiplication cout. What I am interested to highlight is, a fact that this function here I have written it as a global function. This function needs to access the internal of the vector it will need to certainly access this array and the dimension. Similarly, this will also need to access the matrix and its dimensions to be able to actually compute their product.

Now, prod is not a member of either of these classes. In fact, making it a member does not solve the problem, because if I make prod a member of vector then it can access these easily, but it still cannot access them. If I make prod a member of matrix then I can access these but I cannot access this. So here I am providing a solution that I will make it a global function which is outside of each of these classes but, what I do I make prod a friend function in both these classes. If I have made a prod a friend of class vector then it necessarily means that prod will be able to access these private data members because it is a friend of a vector.

Similarly, I need to access the private members of matrix class, so I make prod a friend in the matrix class as well. Once I have done that then this code will compile because whenever I am making references to the private data members of vector or private data members of matrix both are accessible because prod is a friend of both these classes and this code will compile and if I write a main application with that.

(Refer Slide Time: 10:17)

The slide displays the following C++ code and output:

```
int main() {
    Matrix M(1, 3);
    Vector V(3);

    Vector PV = Prod(M, V);

    M.Show();
    V.Show();
    PV.Show();

    return 0;
}
```

Output:

```
0 1 2
1 2 3
1 2 3
8 14
```

Notes:

- Vector Prod(Matrix*, Vector*) is a global function
- Vector Prod(Matrix*, Vector*) is friend of class Vector as well as class Matrix
- This function accesses the private data members of both these classes

NPTEL MOOCs Programming in C++ Partha Pratim Das

Where have I have constructed some matrix of dimension two three vector of dimension three as I am already mentioned that their values are getting filled in by default and this is what the matrix is filled up with, the so function shows that this is a vector is filled up with and then if I do the multiplication PV is the multiplication of the matrix and the vector and PV will turn out to be this output.

Here I could write this function, I would not to have been able to write this function unless I had the friend function feature. Because I can make prod a member of vector or I can make it a member of matrix but I certainly cannot make it a member of both, but to be able to compute the multiplication I need to access the members of both these classes. So, whenever we come across such situation where for completing functionality I need to access the internals or the private data members and methods of two or more independent classes I need the friend function feature to come in place.

(Refer Slide Time: 11:30)

The slide displays the following C++ code for a linked list:

```
#include <iostream>
using namespace std;

class Node; // Forward declaration

class List {
public:
    List(Node *h = 0);
    head();
    tail();
    void display();
    void append(Node *p);
};

class Node {
private:
    int info; // Data of the node
    Node *next; // Ptr to next node
public:
    Node(int i): info(i), next(0) {}
    friend void List::display();
    friend void List::append(Node *);
};

void List::display() {
    Node *ptr = head;
    while (ptr) {
        cout << ptr->info << " ";
        ptr = ptr->next;
    }
}

void List::append(Node *p) {
    if (!head) head = tail = p;
    else {
        tail->next = p;
        tail = tail->next;
    }
}

int main() {
    List l; // Init null list
    Node n1(1), n2(2), n3(3); // Few nodes
    l.append(n1); // Add nodes to list
    l.append(n2);
    l.append(n3);
    l.display(); // Show list
    return 0;
}
```

The diagram illustrates a linked list with three nodes. The first node contains the value 2, the second contains 3, and the third is null. The 'head' pointer points to the first node, and the 'tail' pointer points to the last node. Each node has a 'next' pointer that points to the subsequent node in the sequence.

NPTEL MOOCs Programming in C++ Partha Pratim Das

This is just another example here; the example is I am trying to build a list. So how do I build a list? I have a node class. A node class defines basic node information which has a data part and a link part to link to the next node in the list. And I have a list class, which actually holds this whole list so it is like this. The list class has as you can see two pointers, the header pointer and the tail pointer so in general it will look something like this, I will have a list like this, say this is 2 this is 3 and let say the list has ended here. So, my head will point to the beginning of the list and tail will point to the end of the list. So I want to actually make such a structure and then I would like to use the list as I want.

Suppose, if I want to implement these kind of functionality in the list that is display, that is go over the list and output all the elements that you have or you append to the list. If I want to write say list colon colon display, this is a member function of list. So if I want to implement this then I need the internal information of the node, because unless I can access the info of the node I cannot print that value. Unless I can access the next field of the node I cannot go to the next node on the list so I need to access them.

But display basically is a member function of the list class, so it does not allow me to access the private members of the node class and that is where I make use of the friend. What I do? In the node class I write the signature of the display function. Note the

display is a member function so the name of the display function is actually list colon colon node, so I am writing here as I am sorry, the name of the member function is display and the full name is list colon colon display. So, I am writing the whole signature here and I prefix with the keyword friend.

Both the display and append which are member functions of list are made into friend of the node class, so that means that when I try to implement display or I try to implement append they will be able to access the private data members of the node class and I will be able to get a very nice list implementation in place, and if there is an application simply which creates a null list creates some node objects and then it appends the node object one after the other and it will finally print the list. If you do it you will get a printing of 1 2 3 because we have created 3 nodes with value 1 two and 3, that is not very important.

What I wanted to show that using the friend function you can actually very nicely link up to such classes which are somewhat related but, completely independent one which gives the primitive node that you want to keep the value in and the other that actually manages the list structure, manages the up and delete, display kind of functionality in them. And we are cutting across their encapsulation boundary we are creating a whole function in the encapsulation of node by making some of the member functions of the list class as friend of the node class.

Compared to the earlier example of matrix vector multiplication which showed how a global function can be made a friend of two classes, here I am necessarily shown that a member function, actually two member functions of one class has been made friend of another class. The matrix vector multiplication can also be solved in this tilde that I could have made prod a member function of vector and made that vector colon colon prod a friend of matrix or vice versa that is make prod a member function of matrix and then make matrix colon colon prod a friend of vector.

This is typically how the friend function feature can be used for better advantage.

(Refer Slide Time: 16:18)

The slide is titled "friend class" and is part of "Module 17" on "Partha Pratim Das". It contains the following bullet points:

- A friend class of a class
 - has access to the private and protected members of the class (breaks the encapsulation)
 - does not have its name qualified with the class scope (not a nested class)
 - can be declared friend in more than one classes
- A friend class can be a
 - class
 - class template

A diagram shows two classes, C1 and C2, represented as boxes. C1 is on the left and C2 is on the right. A red arrow points from C2 to C1, indicating that C2 is a friend of C1. The names C1 and C2 are written above their respective boxes.

At the bottom of the slide, it says "NPTEL MOOCs Programming in C++" and "Partha Pratim Das". There is also a small circular inset photo of the presenter in the bottom right corner.

There is certain bit of extension in this, that extension is driven from the fact that if you just think about the list and node example, then we needed both the member functions of list append as well as display to be friend of node. Now, if the list classes are delete function it will also need to be friend of node and so on. There is a shortcut that you can do and that is in terms of a friend class. So if I have two classes then we can make one class as a friend of the other by simply including that class name and prefixing with friend. If a class is a friend of the other then it has access to the private and protected members of the class. For a friend function that particular function has access, for a friend class all methods of that class have access to the private data members and member functions of the friend class irrespective of which member function I am talking of.

Naturally, since this is a class as a whole this class mean is not qualified by the name of the class to which it is the friend, because it is not a nested class it is just that is two independent classes, I have one class here, I have one class here, I have c 1 here, I have c 2 here. I am just saying that c 2 is a friend of c 1 therefore, their name will remain to be their isolated name and certainly a class can be declared as friend in more than one other classes. What can be a friend class of course, a class can be a friend class and a class template can be a friend class. Class template we have not done yet when we complete

talk about template this point will become clear as to how class templates also become friend classes.

(Refer Slide Time: 18:23)

```
#include <iostream>
using namespace std;

class Node; // Forward declaration

class List {
    Node *head; // Head of the list
    Node *tail; // Tail of the list
public:
    List(Node *h = 0):
        head(h),
        tail(h) {}
    void display();
    void append(Node *p);
};

class Node {
    int info; // Data of the node
    Node *next; // Ptr to next node
public:
    Node(int i): info(i), next(0) {}
    //Friend void List::display();
    //Friend void List::append(Node *);
    friend class List;
};

void List::display() {
    Node *ptr = head;
    while (ptr) {
        cout << ptr->info << " ";
        ptr = ptr->next;
    }
}

void List::append(Node *p) {
    if (!head) head = tail = p;
    else {
        tail->next = p;
        tail = tail->next;
    }
}

int main() {
    List l; // Init null list
    Node n1(1), n2(2), n3(3); // Few nodes
    l.append(n1); // Add nodes to list
    l.append(n2);
    l.append(n3);

    l.display(); // Show
    return 0;
}
```

• List class is now a friend of Node class. Hence it has full visibility into the internals of Node class.
• When multiple member functions need to be friends, it is better to use friend class

Now, let us rework the link list example by using the friend class concept. So what I have done in the same earlier example I have just commented out the two lines which we are making the member function display and the member function append of the list class as friends of the node class, instead I have made the whole list class as a friend. This is the way to write it the class and the class name is the way you refer and then you put a friend keyword in front to say that this node class considers that the whole of the list class is its friend. It not only display an (Refer Time: 19:14) wherever whatever other members like delete, like search, whatever member that we might add to the list class all of them will be accepted as a friend of the node class, rest of the implementation is same. Rest of the implementation and this application is no different from earlier.

The only difference is, in these three lines where we instead of using friend function we are using a friend class. So, particularly if two classes have too many member functions to be made friend of then we should consider whether instead of individually making member functions as friends, whether you should actually make the whole classes friend

and that will often make your life easier because you do not have to specifically keep on listing all the different member functions as friend.

(Refer Slide Time: 20:11)

The slide content is as follows:

- **friend-ship** is neither commutative nor transitive
 - A is a friend of B does not imply that B is a friend of A
 - A is a friend of B and B is a friend of C does not imply that A is a friend of C
- **Visibility and Encapsulation**
 - **public**: a declaration that is accessible to all
 - **protected**: a declaration that is accessible only to the class itself and its subclasses
 - **private**: a declaration that is accessible only to the class itself
 - **friend**: a declaration that is accessible only to **friend**'s of a class. **friend**'s tend to break data hiding and should be used judiciously.

Like:

 - A function needs to access the internals of two (or more) independent classes (Matrix-Vector Multiplication)
 - A class is built on top of another (List-Node Access)
 - Certain situations of operator overloading (like stream operators)

NPTEL MOOCs Programming in C++ Partha Pratim Das

So, now you have seen how this friend stuff works. Now let me end with a couple of friendly notes. First point to be noted, is friendship that is when we say one class is a friend of another. This friendship is kind of a binary relationship between two classes and this binary relationship is neither commutative nor transitive. It is not commutative which means that if A is a friend of B then that does not mean that B is a friend of A. If I have to make B a friend of A then within the class scope of A I will have to put friend class B this statement, but by the mere fact that A is a friend of B does not imply that B is a friend of A, so it is not a commutative relation. Similarly, if I have A as a friend of B, B is a friend of C from within these classes that does not imply that A is a friend of C, so the transitivity does not work. Friendship is just binary, it just works between two classes and no further inferencing is possible.

Now having said this, please note that this friend feature of friend function and friend class actually changes the encapsulation and the visibility structure of the language. So far we have had three kinds of visibility of which two we have already discussed public and private, we will soon discuss protected visibility which applies to inheritance. But

friend is a fourth kind of visibility that exist in C++ where you can specifically one class can make some other classes, some other member function a friend and thereby giving it a visibility which is restricted, but is kind of puncturing the whole of encapsulation here. So, it should be what you make friend should be done very judiciously because if you arbitrarily make other classes and other functions, global functions or member functions as friends then all your advantages of encapsulating data and using them accessing them through a proper choice of member functions will get lost.

So, herein we have tried to put a few situations which are common where the friend using friend really helps. First is the matrix vector kind of situation where you have two independent classes, but you have a functionality where the data members, private members of both these classes participate as in case of multiplying a matrix with a vector. Or you have a situation where one class is being built on top of another class as a design component like in a list, list comprise of different node class objects so the list functionality certainly gets for easier to implement if you use node, if you allow node to announce list as a friend of it so that it can look through the whole of the internal of the list. This built up is another situation.

The third which we have not yet actually discussed but I would just put a pointer that, when we tried to overload operators their certain operators which are very difficult to overload with proper syntax and semantics if we did not have friend function kind of functionality available in C++. Example we will show in terms of output input streaming operators when you overload them for specific user defined classes, but in general you should be very cautious, restrictive and conservative in terms of when you use friend function and friend class and you should really make sure that you have one of these different situations and may be some of the very related situations to this really occurring, but otherwise if you just use the friend feature as a function or as a friend of class just to shortcut the design then you are actually breaking down the encapsulation which will go against the basic object oriented framework that we are building up so carefully through the definition of access specifiers and the creation of the objects and so on.

So, friend is a powerful feature and like any powerful feature, like any powerful weapon it must be used with a lot of care and in discretion.

(Refer Slide Time: 25:09)

Module Summary

- Introduced the notion of **friend** function
- Introduced the notion of **friend** class
- Studied the use of **friend** function and **friend** class with examples
- **friend** introduces visibility hole by breaking encapsulation – should be used with care

NPTEL MOOCs Programming in C++ Partha Pratim Das

To summarize in this module we have introduced the notion of friend function and the notion of friend class and we have studied the friend function and friend class with examples of matrix multiplication, and risk manipulation, and we have noted specifically that friend is a different kind of visibility and somewhat dangerous, somewhat risky to use because it can arbitrarily break the encapsulation and therefore the use of friend should be done with very judiciously choice of design with proper design justification of why this punctures is required.

As you go ahead and start doing lot of designs and implementation we will find that kind of almost always that you need to use a friend it will be one of the three situations that I have discussed here, and if you find that you are requiring a friend function or a friend class to be used in a situation which is not like the three that we have discussed then you should be very cautious and careful and really convince yourself that this is a situation which needs the friend to be used.