 **Friend function**

If a function is defined as a friend function in C++, then the protected and private data of a class can be accessed using the function.

By using the keyword friend compiler knows the given function is a friend function.

For accessing the data, the declaration of a friend function should be done inside the body of a class starting with the keyword friend.

**See the video on Friend function using following link:**

<http://www.infocobuild.com/education/audio-video-courses/computer-science/ProgrammingInCpp-IIT-Kharagpur/lecture-32.html>

**Declaration of friend function in C++**

class class\_name

{

   friend data\_type function\_name(argument/s);           // syntax of friend function.

};

In the above declaration, the friend function is preceded by the keyword friend. The function can be defined anywhere in the program like a normal C++ function. The function definition does not use either the keyword friend or scope resolution operator.

**Characteristics of a Friend function:**

* The function is not in the scope of the class to which it has been declared as a friend.
* It cannot be called using the object as it is not in the scope of that class.
* It can be invoked like a normal function without using the object.
* It cannot access the member names directly and has to use an object name and dot membership operator with the member name.
* It can be declared either in the private or the public part.

In this below example **temp** is a friend function of the class **Temperature**. So, it can access all the private and protected members of the class.

**Example 1:**

**Let us implement a programming Example to better understand the usage of Friend Function.**

**Output:**

**Explanation:**

Here we declared a function 'temp' as the friend function of the class 'Temperature'. In the friend function, we directly accessed the private member celsius of the class 'Temperature'.

When the first statement of the main function created an object 'tm' of the class 'Temperature' thus calling its constructor and assigning a value 0 to its data member celsius.

The second statement of the main function called the function 'temp' which assigned a value 40 to celsius.

We can also have the same friend function of two classes as shown in the next example.

**Example 2:**

**Output:**

**Explanation:**

In this example, we declared sum() as a friend function of the classes A and B. So, this function can now access the private and protected members of both these classes. The objects of both the classes are passed into as an argument to the function.

First, we created an object for both the classes, thus calling their respective constructors and initializing the values of their respective data member value to 5 and 3. The function 'sum' then returned the sum of the data members of the two classes by calling the data members of the two classes by their respective objects.

**Frequently Asked Questions about Flowchart**

Some of the most frequently asked questions are:

**Q1**: Guideline that should be followed while using friend function?

 Answer: When the application is needed to access a private member of another class, the only way is to utilize the friend functions.

The following are the guidelines to handle friend functions :

1. Declare the function with the keyword ‘friend’ that is followed by return type and followed by the function name.

2. Specify the class whose private members are to be accessed in the friend function within parenthesis of the friend function.

3. Write the code of the friend function in the class.

- The following code snippet illustrates the use of friend function :

friend void display(car); //Friend of the class 'car'

void display(car myCar)

{

  cout<<"\nThe color of my car is : "<<myCar.color;

  cout<<"\nThe speed of my car is : "<<myCar.speed;

}

**Q2**: What are the characteristics of friend functions?

Answer: A friend function is not in the scope of the class, in which it has been declared as friend.

 It cannot be called using the object of that class.

 It can be invoked like a normal function without any object.

 Unlike member functions, it cannot use the member names directly.

 It can be declared in public or private part without affecting its meaning.

**Q3**: Which of the following is correct about friend functions?

a) Friend functions use the dot operator to access members of a class using class objects

b) Friend functions can be private or public

c) Friend cannot access the members of the class directly

d) All of the mentioned

Ans: d

**Q4:** What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class B

{

              int b;

   public:

              B(){}

              B(int i){

                              b = i;

              }

              int show(){

                              return b;

              }

};

class C

{

              B b;

   public:

              C(int i){

                              b = B(i);

              }

              friend void show(){

                              C c(10);

                              cout<<"value of b is: "<<c.b.show()<<endl;

              }

};

int main(int argc, char const \*argv[])

{

              C c(1);

              c.show();

              return 0;

}

Answer: Error

Q 5: What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class B

{

              int b;

    public:

              B(){}

              B(int i){

                              b = i;

              }

              int show(){

                              return b;

              }

};

class C

{

              B b;

    public:

              C(int i){

                              b = B(i);

              }

              friend void show();

};

void show()

{

              C c(10);

              cout<<"value of b is: "<<c.b.show()<<endl;

}

int main(int argc, char const \*argv[])

{

              show();

              return 0;

}

Answer :