C++ Enumeration

Enum in C++ is a datatype that contains fixed set of constants.

It can be used for days of the week (S, M, T, W, T, F, S), directions (East, South, West, North) etc. C++ enum constants are static and final implicitly.

C++ Enums can be thought of as classes that have fixed set of constants.

- enum improve type safety.
- enum can be easily used in switch.
- enum can be traversed.
- enum can have fields, constructors, methods
- enum may implement many interfaces but cannot extend any class because it internally extends Enum class.

ex.

```
enum week { Mon, Tue, Wed, Thu, Fri, Sat, Sun }
int main() {
    week day;
    day = Fri;
    cout << day+1 << endl
}
```

Output

5.

C++ 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.

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

Declaration

```
class class_name {
    friend data_type function_name (arguments);
};
```

In the above declaration, friend function is preceded by keyword friend. The function can be defined anywhere in the program like a normal C++ function.

Characteristics of friend function -

(i) The function is not in the scope of the class to which it has been declared as a friend.

(ii) It cannot be called using the object as it is not in the scope of that class.

(iii) It can be invoked like a normal function without using the object.

(iv) It cannot access the member name directly and has to use an object name and dot operator with member name.

(v) It can be declared either in private or public part

example

```
class Box {
    private :
        int length;
    public :
        Box (): length(0){}
        friend int printLength (Box);   // friend function
};

int printLength (Box b) {
        b.length += 10;
        return b.length;
}
```

```
int main (){
    Box b;
    cout << printlength(b) << endl;
}
```

Output

10.

{ Thus, we accessed private member length using this function }

example

```
class B;                    // forward declaration
class A {
    int x;
    public:
        void setdata (int i) {
            x = i;
        }
        friend void mini(A,B);
};
class B {
    int y;
    public:
        void setdata (int i) {
            y = i
        }
        friend void mini(A,B);
};
```

```
void mini (A a, B b) {
    if ( a.x <= b.y ) {
        cout << a.x << endl;
    }
    else {
        cout << b.y << endl;
    }
}

Int main () {
    A a;
    B b;
    a.setdata (10);
    b.setdata (20);
    mini (a,b);
}
```

output

10.

In above example, mini function is friendly to two classes. Thus mini can access private members of both A and B.

C++ Friend Class

A friend class can access both private and protected members of the class in which it has been declared as friend.

ex.

```
class A {
    int x = 5;
    friend class B;        // friend class
};
class B {
    public:
        void display (A &a) {
            cout << " value of x:" << a.x;
        }
    };
int main () {
    A a;
    B b;
    b.display (a);
}
```

Output

value of x: 5

class B can access private and protected members of A.