Chapter - 14
More on classes

Friends

Nobody sees my private stuff except my friends.

In C++ a function that is the “friend” of a class can access that private data for that class:

- Friends must be named by the class
- Friends are not the same as member functions

Example

```
    private:

    public:
        // ...
};

// .....

{

    if (s1.count == s2.count)// Friends can access private members

    // Rest of function
```

const member functions

The *const* suffix is used to identify which member functions can be called in a constant instance of a variable.

```
class int_set {
    private:
        // ... whatever
    public:

        int_set(const int_set &old_set); // Copy constructor

        void clear(int value); // Clear an element

};

//...........
int_set a_set;
```

const members

Declaration:

```
class data_list {
    public:
        // Number of items in the list
        const int data_size;

        // ... rest of the class
};
```

Initialization:

```
        data_list(void) : data_size(1024) {
        };
```

Defining a conventional *const* member.`

It's not easy.

You can define it outside the class (the old way):

```
const int foo_size = 100;// Number of data items in the list

class foo {
```

or use the "enum" trick

```
class foo {
    public:
        enum {foo_size = 100};// Number of items in the list
```

static member variables

Static member variables:

- Are shared by all instances of the class. (No matter how many instances (class variables) exists, there is only static member variable allocated)
- Can be accessed conventionally or as *class::var.*

Example

```
// Old way
int stack_count = 0; // Number of stacks currently in use
class stack {
    private:
```

Is the same as:

```
class stack {
    public:
        static int stack_count; // Number of stacks
currently in use
```

Access:

```
stack a_stack;
std::cout << a_stack.stack_count;
std::cout << stack::stack_count;
```

static member functions

Static member functions:

- Can only access static member variables
- Exists one per class, not one per instance of a class
- Can be called conventionally, *var.funct()*, or on using the convention: *class::funct()*

The meanings of *static*

Usage	Meaning
Variable outside the body of any function	The scope of the variable is limited to the file in which it is declared.
Variable declaration inside a function.	The variable is permanent. It is initialized once and only one copy is created even if the function is called recursively
Function declaration	The scope of the function is limited to the file in which it is declared.
Member variable	One copy of the variable is created per class. (Not one per variable.)
Member function	Function can only access static members of the class.