Appendix-F: *C++ Specific Keywords*

The following are the specific keywords in C++. Only the highlighted will be describe in detail.

asm	**friend**	**public**	**throw**
catch	**inline**	**reinterpret_cast**	**try**
class	**new**	**___rtti**	**typeid**
const_cast	**operator**	**static_cast**	**typesafe_downcast**
delete	**private**	**templete**	**virtual**
dynamic_cast	**protected**	**this**	**wchar_t**

asm

Syntax

asm <opcode> <operands> <; or newline>
_asm <opcode> <operands> <; or newline>
__asm <opcode> <operands> <; or newline>

Description

Use the asm, _asm, or _ _asm keyword to place assembly language statements in the middle of your C or C++ source code. Any C++ symbols are replaced by the appropriate assembly language equivalents.
You can group assembly language statements by beginning the block of statements with the asm keyword, then surrounding the statements with braces ({}). The initial brace must be on the same line as the asm keyword; placing it on the following line generates a syntax error.

class

Syntax

<classkey> <classname> [<:baselist>] { <member list> }

<classkey> is either a class, struct, or union.

<classname> can be any name unique within its scope.
<baselist> lists the base class(es) that this class derives from. <baselist> is optional
<member list> declares the class's data members and member functions.

Description

Use the class keyword to define a C++ class.
Within a class:

the data are called data members
the functions are called member functions

friend

Syntax

friend <identifier>;

Description

Use friend to declare a function or class with full access rights to the private and protected members of an outside class, without being a member of that class.
In all other respects, the friend is a normal function in terms of scope, declarations, and definitions.

private

Syntax

private: <declarations>

Description

A private member can be accessed only by member functions and friends of the class in which it is declared.
Class members are private by default.
You can override the default struct access with private or protected but you cannot override the default union access.
Friend declarations are not affected by these access specifiers.

protected

Syntax
protected: <declarations>

Description
A protected members can be accessed by member functions and friends of the class in which it was declared, and by classes derived from the declared class.
You can override the default struct access with private or protected but you cannot override the default union access.
Friend declarations are not affected by these access specifiers.

public

Syntax
public: <declarations>

Description
A public member can be accessed by any function.
Members of a struct or union are public by default.
You can override the default struct access with private or protected but you cannot override the default union access.
Friend declarations are not affected by these access specifiers.

templete

Syntax
template < template-argument-list > declaration

Description
Use templates, (also called generics or parameterized types) to construct a family of related functions or classes.

this

Syntax
```
class X {
        int a;
public:
        X (int b) {this -> a = b;}
```

Description
In nonstatic member functions, the keyword "this" is a pointer to the object for which the function is called. All calls to nonstatic member functions pass "this" as a hidden argument.

The keyword "this" is a local variable available in the body of any nonstatic member function. Use it implicitly within the function for member references. It does not need to be declared and it is rarely referred to explicitly in a function definition.
For example, in the call x.func(y), where y is a member of X, the keyword "this" is set to &x and y is set to this->y, which is equivalent to x.y.
Static member functions do not have a "this" pointer because they are called with no particular object in mind. Thus, a static member function cannot access nonstatic members without explicitly specifying an object with . or ->.

virtual

Syntax
virtual class-name
virtual function-name

Description
Use the virtual keyword to allow derived classes to provide different versions of a base class function.
Once you declare a function as virtual, you can redefine it in any derived class, even if the number and type of arguments are the same.
The redefined function overrided the base class function.