* Generic Programming

The term ***generic*** refers to code that is **type independent**. C++ data representations come in many type, integers, numbers with fractional parts, characters, strings of characters, and user-defined compound structures of several types. If, for example, **you wanted to sort data of these various types, you would normally have to create a separate sorting function for each type**. Generic programming involves extending the language so that you can write a function for a generic (that is, an unspecified) type once and use it for a variety of actual types. **C++ templates provide a mechanism for doing that**.

* **Linking** combines your object code with object code for the functions you use and with some standard startup code to produce a runtime version of your program. The file containing this final product is called the executable code.
* C++11 support. For instance, g++, beginning with version 4.3, currently uses the std=c++0x flag when compiling a source code file:

**g++ -std=c++0x use\_auto.cpp**

* To access functions defined in the math library, you may have to add the **-lm** flag to the command line:

**g++ main.cpp –lm**

* **C++ is case sensitive;** that is, it discriminates between uppercase characters and lowercase characters. This means you must be careful to use the same case as in the examples. For example, this program uses cout, and if you substitute **Cout** or **COUT**, the compiler rejects your offering and accuses you of using unknown identifiers.
* C++ can, in fact, use printf(), scanf(), and all the other standard C input and output functions, **provided that you include the usual C *stdio.h* file.**
* If your program is to use the usual C++ input or output facilities, you provide these two lines:

***#include <iostream>***

***using namespace std;***

* The C ***math.h*** header file supports various C math functions. Initially, C++ did the same. For instance, the header file supporting input and output was named **iostream.h.** But C++ usage has changed. Now the .h extension is reserved for the old C header files (which C++ programs can still use), whereas **C++ header files have no extension**. **There are also C header files that have been converted to C++ header files**. These files have been **renamed by dropping the h extension** (making it a C++-style name) and **prefixing the filename with a c** (indicating that it comes from C). **For example, the C++ version of *math.h* is the *cmath* header file.**
* **Namespaces**

If you use ***iostream*** instead of ***iostream.h***, you should use the following namespacedirective to make the definitions in iostream available to your program:

***using namespace std;***

**If not, one potential problem is that you might use two prepackaged products that both have, say, a function called** **wanda()**. If you then use the wanda() function, the compiler won’t know which version you mean. The namespace facility lets a vendor package its wares in a unit called a namespace so that you can use the name of a namespace to indicate which vendor’s product you want. So Microflop Industries could place its definitions in a namespace called Microflop. Then ***Microflop::wanda()*** would become the full name for its wanda() function. Similarly, ***Piscine::wanda()*** could denote Piscine Corporation’s version of wanda().

**In this spirit, the classes, functions, and variables that are a standard component of C++ compilers are now placed in a namespace called std**. **This takes place in the h-free header files.**

***using namespace std;***

**This using directive makes all the names in the std namespace available**. Modern practice regards this as a bit lazy and potentially a problem in large projects. The preferred approaches are to use the std:: qualifier or to use something called a using declaration to make just particular names available:

***using std::cout;*** // make cout available

***using std::endl;*** // make endl available

***using std::cin;*** // make cin available

* **The Newline Character**

C++ has another, more ancient, way to indicate a new line in output, the C notation ***\n***:

***cout << "What’s next?\n";*** *// \n means start a new line*

The ***\n*** combination is considered to be a single character called the **newline character.**

* **using namespace std作用域**

#include <iostream>

**void simon(int);**

int main()

{

**using namespace std;** **// affects all function definitions in this main()**

simon();

……

}

void simon()

{

**using namespace std;** **// affects all function definitions in this simon()**

cout << "Simon says touch your toes " << endl;

}

#include <iostream>

**using namespace std;** **// affects all function definitions in this file**

void simon(int);

int main()

{

simon(3);

int count;

cin >> count;

simon(count);

cout << "Done!" << endl;

return 0;

}

void simon(int n)

{

cout << "Simon says touch your toes " << n << " times." << endl;

}

也可以直接用

**std::cout** << "I’m using cout and endl from the std namespace" << **std::endl;**