Supplement III.A: Preprocessor Directives

For Introduction to C++ Programming
By Y. Daniel Liang

1 Introduction

The C++ compiler command performs three tasks in sequence: *preprocessing*, *compiling*, and *linking*. The compiler first processes the directives. The directives start with the **#** sign. Preprocessor directives are not C++ statements, so they do not end with a semicolon (**;**).

This supplement summarizes the directives used in the text.

2 The #include Directive

The **#include** directive causes a copy of the specified file to be included in the place of the directive. Two forms of the #include directive are

```
#include <filename>
#include "filename"
```

The first line is for including standard C++ header files such as **<iostream>**, **<cmath>**, **<ctime>**, and **<iomanip>**. The second is for including user-defined files.

3 The #define Directive

The **#define** directive defines a symbolic constant. The syntax is

```
#define constantSymbol value
```

For example,

```
#define PI 3.14159
```

defines **PI** to represent the value **3.14159**. This directive tells the compiler to replace all subsequent occurrences of PI after the directive to be replaced by **3.14159**.

4 Conditional Directive

The conditional directive is like an **if** statement to tell the compiler whether to execute the code directive under a condition. The syntax is:

```
#ifndef identifier
```

```
        #define identifier
        Other code to be executed
        #endif
```

The conditional directive determines whether the **identifier** is defined. If not, the #define directive defines the **identifer** and other code before the #endif directive is executed. If the **identifier** is already defined, the code between the directive **#ifndef** and **#endif** is ignored.