目 录

第一章 入 门

1.1 安装 MS C/C++ 7.0

在安装 Microsoft C/C++ 7.0 时，最好能有以下设备，这样在执行时才能发挥较高的效率：

- IBM 或与其 100%兼容的个人电脑。
- DOS 3.3 或速度更高的 CPU。
- 至少尚有 8 MB 空间的硬盘。如果要安装全套的 MS C/C++ 7.0，则必须至少有 27 MB 的硬盘空间。
- Microsoft Windows 3.x 或装有 DPMI 的服务器。

你可以选择通过 MS Windows 3.x(当前为 3.0 或 3.1)来安装 MS C/C++ 7.0，或通过 DOS 环境来安装。

1. 由 MS Windows 3.x 来安装

我们以 MS Windows 3.x 为例，示范如何安装 MS C/C++ 7.0。由于 MS Windows 3.x 本身即为 DPMI 的服务器，因此安装是比较容易的。

利用 Windows 3.1 来安装 MS C/C++ 7.0，可以有以下两种启动方法：

(1)进入 MS Windows 3.x 所在的硬盘子目录，然后在 DOS 的提示符号(Prompt)下键入：

C:\WIN31\WIN A:\SETUP

然后按下 Enter 键。假设由 A 驱动器来安装 MS C/C++ 7.0 至硬盘上，且 MS Windows 3.x 已经安装在硬盘 C 的 WIN31 子目录下。

(2)直接进入 A 驱动器中。在 DOS 提示符号下键入：

A:\>SETUP

然后按下 Enter 键。

如果利用第二种方法，MS C/C++ 7.0 的安装程序会自动找出硬盘中是否已经存在 Windows 3.1(或 3.0)。如果有，便先进入 Windows 3.1 再开始安装 MS C/C++ 7.0。

假设你已经安装了 MS Windows 3.1，并利用第一种方式启动安装程序，则安装程序会先进入 Windows 3.1，不久之后你将会看到一个画面。

输入目录后按 Continue 钮继续，则出现另一个画面。在此画面中，可以选择三种安装的方式：

1)Default installation

最简单省时的安装方式，用户只要根据提示，将 MS C/C++ 7.0 磁盘放到安装驱动器(如 A 驱动器)中即可。

2)Custom installation

此种安装过程中，用户必须逐一指定 MS C/C++ 7.0 所提供的各种功能是否要安装至硬盘中，如果你不能确保硬盘空间足以容纳全部的 MS C/C++ 7.0 软件，则最好选择此功能。对于初学者而言，选择 Defalut Installation 会比较适合。

3)Build Additional Libraries

此选择项用来加入新的函数库到事先已经安装完成的 MS C/C++ 7.0 中，也就是当 MS C/C++ 7.0 已经安装在硬盘时，选择此功能才有意义。

假设你选择了第二种安装方式，Custom Installation，则将出现另一个画面。在此画面中，必须设置是否要安装该画面左方所列举出的项，而右方的按钮则可以设置更详细的安装情形，例如，如果你设置安装 Run－Time Libraries，则再按下 Libraries…按钮可产生. EXE 或 . DLL的 Windows 执行文件，以及设置哪种数学运算模式。

你可以由画面下方的 Space Required 项右方所显示的数字知道你所选择的选择项需要多少硬盘空间，这个数字不可以超过 Space Available(表示你尚有多少硬盘空间)项右方所显示的数字。

一切设置完成后，按下 Continue 钮，会出现一个信息窗口，说明安装程序将开始拷贝磁盘中的文件至硬盘中，如果你有任何觉得不妥之处，请利用 Back 按钮回到先前的步骤重新设置。如果你认为没有任何问题，请按下 Continue 钮，安装程序便根据你所设置的情况开始拷贝 MS C/C++ 7.0 磁盘(共 11 片)中的文件至硬盘中。

在安装的过程中，你必须根据安装程序所要求的磁盘号码逐一插入安装驱动器中(如 A 驱动器)，安装程序会在漫长的拷贝过程中安排你浏览 README. TXT 文件，在这个文件中，你将可以了解到很多必要的信息。

拷贝过程完成后，安装程序会询问你是否要将 MS C/C++ 7.0 的图标(Icons)加入到程序管理员(Program Manager)中，以及是否要让安装程序更改系统设置文件(AUTOEXEC . BAT以及 CONFIG. SYS 两个文件)，我们建议你让安装程序为你更改系统设置文件。

最后在安装 MS C/C++ 7.0 结束前，安装程序会出现一个窗口，告诉你在安装完成后，可以试着去完成那些工作。

2. 由 DOS 环境中来安装 MS C/C++ 7.0

要通过 DOS 环境来安装，如果已经安装 Windows 3. x 在硬盘中，则请在 A 驱动器中键入 CSETUP，如：

A:\>CSETUP

然后按下 Enter 键来启动 MS C/C++ 7.0 的安装程序。

如果你未曾安装 Windows 3. x，则只要键入 SETUP 即可，如：

A:\>SETUP

再按下 Enter 键来启动安装程序。

安装过程不再赘述。

由于 Windows 3. x 本身为 DPMI 的服务器，因此在安装过程不会有任何打开 DPMI 的信息。但在 DOS 下安装时，安装程序可能会警告你找不到 DPMI 服务器，所以将来你使用 MS C/C++ 7.0 的 PWB 或命令列来编译程序时，必须先启动 DPMI 服务器，如 386MAX 等，这是非常重要的。但如果你将 MS C/C++ 7.0 安装在 Windows 3. x 中不需要做这项工作。

如果你在 Windows 3.x 中执行 MS C/C++ 7.0 的 PWB，发生了类似死机或其他无法执行的问题时，则请依下列步骤进行补救：

1. 执行 Main 程序群中的 PIF Editor。

2. 进入 PIF Editor 后，利用 File/Open 打开在目录 D:\C700\BIN 下的 PWB. PIF 文件。

3. 打开后，注意 Excution 项的设置情况，如果是设置在 Background 上，则请关闭此设置，然后改为 Exclusive。

4. 然后再利用 File/Save 将更改后的设置存储即可。

1.2 启动 MS C/C++ 7.0

安装好 MS C/C++ 7.0 后，即可进入 MS C/C++ 7.0 的程序设计集成开发环境，也就是所谓的 PWB(Programmer's WorkBench，程序员平台)。

安装在 Windows 下的用户，需要在先进入 Windows 环境后，再选择 Microsoft C/C++ 7.0 程序组中的 Programmer's WorkBench 图标，然后用鼠标双击此图标便可以进入 MS C/C++ 7.0 的 PWB 环境。

而安装在 DOS 环境下的用户，则必须先启动 DPMI 服务器，如 386MAX 后，再进入 PWB. EXE 所在的子目录，如 D:\C700\BIN，然后在 DOS 的提示符号下键入：

D:\C700\BIN>PWB

按下 Enter 后便可以进入 MS C/C++ 7.0 的 PWB 环境了。

当你第一次进入 MS C/C++ 7.0 时，会出现 MS C/C++ 7.0 的 PWB 集成环境画面。其中最上方为主菜单，中央部分为程序编辑工作区，最底一行为信息区。

如果你对于菜单中的选择项的用法不太明了，则可以将光条移动至该选择项，然后仔细阅读信息区中的文字。若还想进一步的了解该选择项的意义或用途，则可以按下 F1 键，立刻会出现有关该功能选择项的辅助信息。

如果用户不了解如何打开一个新的程序文件，则可以将光条移至 File|New 的位置，然后查看屏幕下的信息区。或者再按下 F1 键，以便得到更详尽的辅助信息。

要关闭辅助信息窗口，只要按下 Esc 键，或将鼠标光标(如果你有驱动鼠标)移到窗口右下角的〈Cancel〉钮上，按下鼠标左键即可。

1.3 使用 PWB

1. 打开文件

要设计程序，首先必须利用 File|New 或 File|Open 菜单来打开文件，假使我们选用 File/New 菜单，就可以发现工作区中多出了一个窗口，该窗口便是代表着我们新打开的程序文件，假设文件名为 Untitled. 001，每个程序编辑窗口的左上角都会有一个关闭钮，用来关闭窗口并使其由屏幕上消失，而程序编辑窗口的右上角则有一个指向上的箭头，代表着极大钮，

可以将编辑窗口的编辑区放至最大，而指向下的箭头即代表着极小钮，会将编辑窗口的编辑区缩至最小，只剩一个图标来代表。在关闭钮的右方还有一个数字，当前显示为 1，每一个打开在工作区中的编辑窗口皆含有一个数字，表示窗口的编号。利用 Alt＋窗口编号，可以直接将工作编辑窗口移到指定的编辑窗口上。

所谓工作编辑窗口，是指可以输入文字、编写程序以及编译程序的窗口。工作区中可以同时打开多个程序文件，每一个文件具有一个独立的编辑窗口，但一次只能有一个编辑窗口可以被用来编写或编译程序。

现在我们要开始将程序编译成在 DOS 环境下的可执行文件，在进行编译前，必须先完成几项工作。首先选择 Options 菜单下的 Options|Project Templates|Set project Template... 选项，接着必须设置编译器应将所编写的程序以哪一种语言来编译，请在 Runtime Support 栏中选择 C++ 一项（将光条移动到 C++ 项上即可）。接着按下 Tab 键切换选择到 Project Templates with Runtime Support for ：C++ 中。在此选择栏内，必须设置程序在编译并连接后，要产生那一种环境下的可执行文件。请选择 DOS EXE 一项。

2. 存储程序

现在，请先将当前编辑窗口中的程序存储起来。MS C/C++ 7.0 的 PWB 中提供了三个存储文件的方法。

(1)File|Save

存储当前作用窗口中的程序，其文件名便是工作窗口头上所显示的文件名。

(2)File|Save as

存储当前工作窗口中的程序，如果要改换文件名，将会出现一个要求你输入新文件名的窗口。当输入新文件名并按下 Enter 键后，PWB 便会产生这个新的文件，并将此新文件换为当前工作编辑窗口中所编辑的文件。

(3)File|Save all

存储工作区中所有曾经被更改的文件。

现在利用 File|Save As... 选择项将程序存储成 hello. cpp。此时应可以见到编辑窗口的标头也随之改变成所输入的文件名。特别需要注意的是，程序文件的附加文件名必须是. CPP 才能以 C++ 的语法来进行编译。

3. 产生可执行文件

程序必须先经过编译、然后连接才能产生可执行的应用程序，利用 PWB 中的 Project|Build 命令可以一次完成这两个步骤，而产生可执行文件。如果你只想进行编译方式，则可以选择 Project|Compile File 命令。

Project|Build 命令选择项的右方同时会显示所产生的可执行文件的文件名，在本例中便为 hello. exe。

每次编译或连接后在所显示的窗口中，它告诉程序是否曾在编译或连接的方式中发生错误或警告的情况。如果没有任何情况发生，就表示成功地产生一个可执行文件。此时你可以选择该窗口中的〈Run Program〉选择项来执行所产生的应用程序，或选择〈View Result〉来

查看整个编译或产生可执行文件的过程中发生了那些情况。如果选择了〈View Result〉菜单，则 PWB 会立刻产生一个名为 Build Results 的窗口，在此窗口中记录着所有编译或产生可执行文件的过程中所发生的所有情形，如错误信息与警告信息等。

4. 使用项目文件

以 Project|Build 方式来编译当前工作窗口的程序文件并产生可执行的应用程序虽然很方便，但是唯一的缺点便是一次只能编译一个文件。对于一个大型应用程序的开发，相信没有任何程序设计者会将所有的程序码集中在同一个文件中，然后进行编译与连接。优秀的程序设计者应会尽量将应用程序的开发分成若干文件来编写，并分别编译，最后再一起连接产生可执行的文件。但是这便造成编译与连接的过程过于复杂。于是 MS C/C++ 7.0 的 PWB 提供了 Project 菜单，以解决这个问题，使得在 MS C/C++ 7.0 中设计应用程序更为得心应手。

假设有一个应用程序共分为七个程序文件来开发，则设计者便可以建立一个项目文件来编译并连接这七个程序文件，以产生最后的应用程序。

首先选择 Project 选单下的 New project... 选择项，准备打开一个项目文件，在此窗口内你必须输入一个欲打开的项目文件名，附加文件名预设为. MAK，请输入 new，然后选择 OK。

打开新的项目文件时，起初并无任何程序文件，现在设计者可以开始将应用程式所需要的程序文件加入到 NEW. MAK 中，选择 Project 下的〈Add/Delete〉选择项将所需要的程序文件由 File List 栏中逐一加入（先将 File List 栏中的光条移动到你所希望加入的文件上，再按下 Alt+A 即可）。全部加入后，选择〈Save List〉即可。

之后设计者只要执行 Project|Build 或 Project|Compile，PWB 便会依据当前打开的项目文件中有哪些程序文件，逐一进行编译或连接，最后产生一个与项目文件同名的可执行程序。

必须特别注意，由于 PWB 中一次只能打开并使用一个项目文件，因此当你希望编译其他程序文件时，必须先关闭当前使用中的项目文件，因为当 PWB 中某个项目文件被打开时，则编译器（Compiler）永远不会去理会不含在此项目文件内的程序文件。

第二章　C++特性

在进行面向对象程序设计以前，必须先熟悉在C++中提供了哪些不同于一般传统C语言的功能。

2.1　注　解

C语言的注解像读者所熟知的，由以下的格式所构成：

```
/*.....*/
```

如以下的注解都是正确的：

```
/* 单行注解 */
/* 多
行
注
解*/
```

而在C++中，除了以上的方法外，提供了一种更为方便的格式，以//开始，止于该行的行尾。如下所示：

```
// C++的注解格式，只适用于单行注解
```

下面是一个更完整的范例，读者可以充分了解在C++中的注解表示方法：

```
/* 本程序是一个适用于
* C++程序设计初学者的好例子
*/

# include <stdion.h>        // 包含标准的I/O库

// main() function
/* 由于 main 函数不返回任何值，
* 因此最好将它声明为一个 void 函数。
*/
void main()
{
    printf("Hello World !\n");
}
```

程序的输出结果如下：

Hello World!

由于 main 函数不返回任何值，所以必须声明为 void，在 C 中编译器视无声明的函数的返回值为整数(int)类型，但在 C++ 中每一函数都必须有返回值(无论是否有返回值，若无则为 void 类型)声明。

2.2 简单输出与输入

在 C++ 中，对于输出与输入所提供的函数库除了 C 语言中旧有的 stdio 外，还提供了一个相当方便的 I/O 函数库——iostream。

C++ 为面向对象特别提供了一套相当方便的 I/O，其定义可在 iostream.h 中找到。

首先为读者介绍两个标准的输出输入：

cout：标准输出

cin：标准输入

两个运算符——插入运算符：

<<：输出运算符

>>：输入运算符

有 C 语言经验的读者可能会觉得很奇怪，运算符<<及>>不是右移及左移运算符吗？怎么又是什么 I/O 用的插入运算符？理由是 C++ 语言允许运算符以及函数是可以重复定义的(overloading，重载)，意即相同的运算符可以有不同的功能，这也是 C++ 语言的一大特色，后面我们还会有详细的说明。

使用的方法很简单，将要输出或输入的数据通过输出运算符将它送到标准输出(cout)，或通过输入运算符将数据由标准输入(in)输入。

首先我们来看看输出。请读者看一下下面这个程序：

```
#include <iostream.h>                    // 首先包括 iostream.h 和 stdio.h
void main()
{
    cout << "Hello World ! \n";          // 打印一个字符串
    cout << 23;                          // 打印整数
    cout << "\n";                        // 换行
    cout << 25.64;                       // 打印实数
    cout << "\n";
                                         // 打印混合数据
    cout << "Hello! My name is Yi Ou ";
    cout << "I'm " << 23 << " years old.\n";
}
```

上面程序的输出结果如下：

```
Hello World!
23
25.64
Hello! My name is Yi Ou I'm 23 years old.
```

一般来说，用 cout 来做输出工作要比用 printf 方便多了。从程序中我们可以发现，cout 函数可以自动判别我们所要输出的数据为何种类型而自动调整输出的格式，不必像 printf 那样，每个都要由用户指定。这样不但方便，而且还可以减少错误的机会。连续的输出也可以像 printf 一样连接起来，例如：

```
cout <<"I'M"<<23<<"years old.\n";
```

同理，输入也是通过输入运算符将数据读入，参见如下程序：

```
#include <iostream.h>

void main()
{
    char name[20];
    int age;
    cout << "Please give your name (less than 20 characters):";
    cin >> name;
    cout << "Please tell me how old are you ?";
    cin >> age;
    cout << "Your name is " << name << ".\n";
    cout << "You are " << age << " years old.";
}
```

程序的输出结果：

```
Please give your name(less than 20 characters):Yi Ou
Please tell me how old are you ? 22
Your name is Yi Ou.
You are 22 years old.
```

其中加着重线的为用户的输入值。

2.3 C++的动态内存分配

在程序开发过程中，动态内存分配是常用到的一个技巧。在 C 中，我们最常用的便是 malloc 函数向操作系统申请一块适当大小的内存，等到使用结束，再用 free 函数将它释放回操作系统。但在 C++ 中提供了一组更为方便的运算符来做动态内存分配：

new：申请一块新的内存，并返回所分配空间的指针

delete：释放由 new 函数所分配的内存

程序例子如下：

```
void main()
{
    int *number;                    // 整型指针
    char *string;                   // 字符指针

    number = new int;               // 从内存为 number 分配两个字节
    delete number;                  // 释放两个字节
    number = new int(2);            // 分配和初始化，number 为 2
    delete number;                  // 释放两个字节
    string = new char[20];          // 从内存为 string 分配 20 个字节
    delete string;                  // 释放 20 个字节
}
```

new 运算符后接类型标识符则可分配一块内存，其大小便是该类型所占的内存大小。如上面程序中的

```
number=new int;
```

便是从内存中申请两字节大小的空间给整数指针变量 number。而类型标识符后若接有括弧()，则将为该变量设置初值。如上面程序中的

```
number=new int(2);
```

便是从内存中申请两字节大小的空间给整数指针变量 number，并指定变量 number 的值为 2。

若类型标识符后接中括弧[]，则可申请分配具有数个该类型大小的空间，如上面程序中的

```
string=new char[20];
```

便是从内存中申请 20 个字符(字节)大小的空间，并赋给字符指针变量 string。

释放空间时，便使用 delete 运算符，如上面程序中的

```
delete number;
delete string;
```

便是将原先由 new 运算符所申请并赋给指针变量 number 及 string 的空间释放回系统，以作其他用途。

2.4 定义与声明

1. 数据的定义与声明

让我们先来看看两个例子：

```
int val1;
extern int val2;
```

我们称第一行为定义，而第二行则为声明。

所谓定义，是指当我们包含一个变量名称时，也同时为该变量分配一块内存。如同第一行的

```
int val1;
```

此时我们便是告诉编译器要产生一个名为 val1 的整数变量，并且编译器会为它分配两个字节大小的空间，这便是数据定义。而就第二行

```
extern int val2;
```

而言，我们告诉编译器程序中将引用整数变量 val2，但该变量可能在某个程式中的某个地方已经定义过了，因此我们在其前方加入关键字 extern 告诉编译器此变量为外部变量，则此时编译器便不会为该变量分配任何内存，这便是数据声明。

此外，值得注意的一点是，在 C++ 中变量数据的声明与定义是可以位于程序中任何地方的，这一点与在 C 中大为不同，在 C 中所有的变量的声明与定义都必须位于程序或函数的最开始处。

请看以下程序：

```
#include (conio.h)
#include (iostream.h)

void main()
{
    cout (( "Now we are going to print all ASCII code : \n" ;
    cout (( "Press any key to continue...\n" (( flush;
     getch();
    // 不在函数 main()的头上定义变量 i
    for(int code=32; code(256; code++) {
        putch(code);
        if (code%70 -- 0)          // 每行 70 个字符
        cout (( "\n";              // 换行
```

```
    }
    cout << "\nThat's all ! \n" << flush;
     getch();

}
```

在上面程序中，读者可以发现整数变量 code 的定义不必一定要在程序的最开始处：

```
    ...
    // 整数变量 code 定义位于程序中
    for(int code=32;code<256;code++)
        putch(code);
        if (code%70==0)
        cout <<"\n";
    }
```

在上面程序中第一次使用了操纵函数(Manipulators)，flush。这是有关 C++ 输出方面的应用，利用 flush 操纵函数可以强迫应用程序将当前输出缓冲区(buffer)中的数据送到输出设备上，也就是屏幕。另外一个好用的操纵函数还有 endl，利用此操纵函数可以达到换行并回到第一行的作用，就好像是使用\n 的情形一样：

```
    printf("Hello ! Microsoft C/C++ 7.0 \n");
```

的作用与以下相同：

```
    cout <<"Hello ! Microsoft C/C++ 7.0"<<endl;
```

2. 函数的声明与定义

函数原型

就像不能使用一个未经编译器认可的变量一样，也不能使用一个编译器不知道的函数，因此函数的声明便是提供函数的原型来告诉编译器我们所要使用的函数的信息。函数的声明格式如下：

```
    类型标识符  函数名称(参数);
```

例如：

```
    void noreturn();
    int retinteger(int agru);
    float retfloat(int agru1,float agur2);
```

以上都是合法的函数声明，函数声明的主要用意是告诉编译器在程序中将调用这些已声明的函数。

在 C++ 中，所有的函数都必需先声明后才可使用，这便是在 C++ 中所特别要求的函数原

型(function prototypes)。

函数的定义

而函数定义便是将一个已声明的函数实例化,意即函数定义便是该函数的实际程序。

函数的定义必须包含两部分:声明符(declarator)以及函数本体(function body)

如:

```
int noreturn()          //声明符
{                       //函数体开始
    ...
}                       //函数体结束
```

声明符必须与当初所声明的函数完全一致。所谓的完全一致是指有相同的返回值类型、函数名称、参数类型、参数个数。

让我们举一个完整的例子来加强读者的印象,程序清单如下:

```
#include <graph.h>
#include <conio.h>
#include <iostream.h>
#include <stdlib.h>

enum Size { Small, Big };                   // 比 7 大或比 7 小
enum Result { Lost, Win };                  // 游戏结果

void main()
{
    // 定义所需的数据
    int money=1000;                         // 原有的钱数
    int bet=0;                              // 下注的数目
    int number;                             // 计算机所给的数字

    // 函数原型
    int getbet(int *mon);                   // 下注
    Size getplayer();                       // 取玩者的输入

    // 取计算机给出的数字
    int getcompernumber();

    // 确定谁赢
    Result compare(Size sob, int num);
```

```
    // 显示游戏结果
    void showresult(int mon,Result ret);

    // 游戏部分
    Size small_or_big;
    Result win_or_lost;
    //_clearscreen(_GWINDOW);
    int conti = 0;                          // 表示要玩多次
    while ( ! conti ) {                     // 缺省为只玩一次
     bet = getbet(&money);                  // 取下注数
     small_or_big = getplayer();            // 取大小比较结果
     number = getcompernumber();            // 取数字
     cout << "\nThe number is " << number << " ";
     win_or_lost = compare(small_or_big,number);
                              // 赢否 ?
     if ( win_or_lost == Win )              // 赢
         money += bet * 2;
         showresult(money,win_or_lost);     // 显示结果
         cout << "Do you want to play again ? (Y/N) ";
         char c;
         cin >> c;                          // 还玩吗 ?
         cout << "\n";
         conti = ( c == 'Y' || c == 'y' ) ? 0 : 1;
    }

}

int getbet(int *money)
{
    int bet;
    cout << "How much do you want to bet ? \n";
    cin >> bet;
    if ( bet > *money ) {
     cout << "You don't have so much money, maybe\
next time ! \n";
     exit(0);
    }
```

```
    *money -= bet;
    return bet;
}

Size getplayer()
{

    Size input;
    cout << "(B)igger or (S)maller than 7 ? ";
    char c;
    cin >> c;
    cout << "\n";
    input = ( c == 'B' || c == 'b' ) ? Big : Small;
    return input;
}

int getcompernumber()
{

    int number = rand();
    number = number%13;
    return number;
}

Result compare( Size sob, int number )
{
    Result result;
    Size smallorbig = ( number > 7 ) ? Big : Small;
    result = ( sob == smallorbig ) ? Win : Lost;
    if ( number == 7 )
        result = Lost;
    return result;

}

void showresult(int money, Result result)
{
    char *wol[] = {     "Lost !",
```