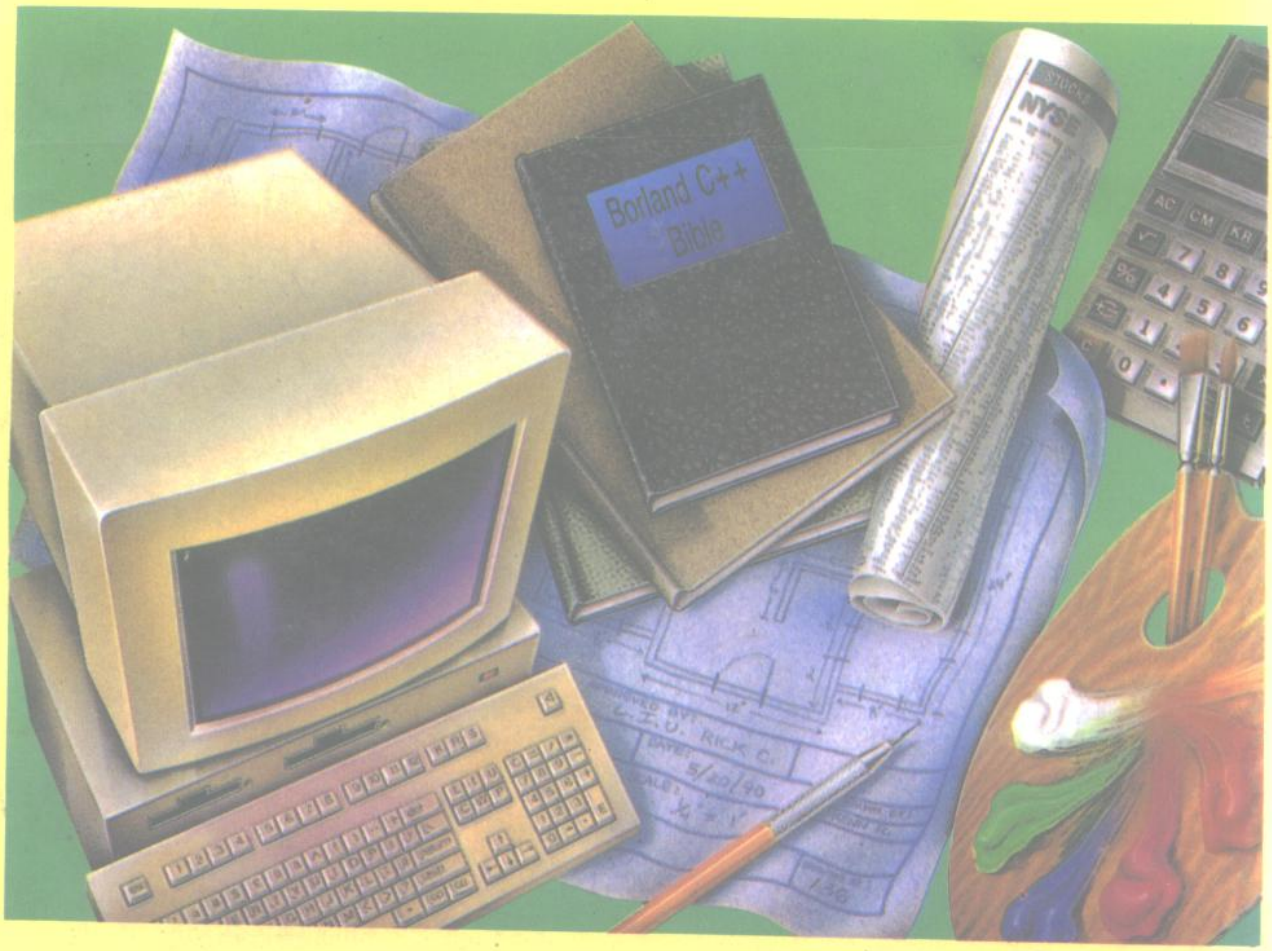
清华松岗系列丛书

Borland C++ 实用绘图设计

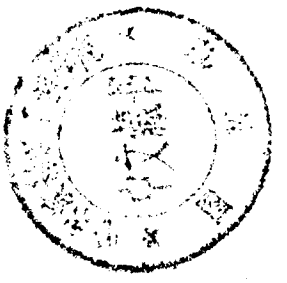
蔡明志 编著



清华大学出版社

Borland C++实用绘图设计

蔡明志 编著



清华大学出版社

(京)新登字 158 号

Borland C++绘图设计(繁体字版)
Borland C++实用绘图设计(简体字版)

蔡明志 编著

图书在版编目(CIP)数据

Borland C++实用绘图设计/蔡明志编著,—北京:清华大学出版社,1994.6
ISBN 7-302-01528-7

Ⅰ.B… Ⅱ.蔡… Ⅲ.①C 语言-计算机制图-程序系统②C 语言-计算机图形学-程序系统 Ⅳ.①TH126②TP391.4

中国版本图书馆 CIP 数据核字(94)第 06036 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
责任编辑:焦金生
印刷者:北京大学印刷厂
发行者:新华书店总店北京科技发行所
开本:787×1092 1/16 印张:23 字数:545 千字
版次:1994 年 7 月第 1 版 1994 年 7 月第 1 次印刷
本社分类号:TP·621
印数:0001—6000
定价:29.50 元

序　言

本书旨在利用 Borland C++(或 Turbo 系统的产品)来开发一绘图系统,在这里我们几乎皆使用 Borland C++所提供的绘图函数,当然在一些特殊的地方可能还利用了一些技巧。

本书先简介 C++的基本概念及绘图函数,进而剖析本绘图系统的各项子功能。对每项子功能笔者皆以一章的篇幅来解释,使读者完全理解每个子功能的意义及作法。

鼠标的使用也是本书的重点,当然有鼠标的辅助,才能算是具有一套优良的绘图系统。此处虽然没有将全部的鼠标功能加以介绍,但所介绍的足够应用了。

最后将每个子功能组合为一,在此您可以看到此完整的绘图系统,读者可以按其需求加以修改。

蔡明志

1993.9.27

目　录

第一章　C++的基本概念 …… 1
1-1　前言 …… 1
1-2　C++新增的命令 …… 1
1-3　类、对象与数据封装 …… 5
1-4　继承(Inheritance) …… 10
1-5　虚拟函数与多态性 …… 15

第二章　绘图函数 …… 22
2-1　前言 …… 22
2-2　介绍函数 …… 22

第三章　鼠标的功能 …… 37
3-1　前言 …… 37
3-2　鼠标功能的介绍 …… 37
3-3　定义鼠标的功能 …… 38
3-4　范例程序 …… 52

第四章　绘图系统封面 …… 73
4-1　前言 …… 73
4-2　计算各区域坐标 …… 73
4-3　范例程序 …… 75

第五章　消息框及立体字型的制作 …… 86
5-1　前言 …… 86
5-2　空白消息框的制作 …… 86
5-3　立体字型的制作 …… 92
5-4　各类消息框 …… 95
5-5　函数的分析 …… 100

第六章　铅笔功能的制作 …… 104
6-1　前言 …… 104

6-2 范例程序 …… 104
6-3 程序分析 …… 107

第七章 线条功能的制作 …… 111
7-1 前言 …… 111
7-2 范例程序 …… 111
7-3 程序分析 …… 115

第八章 连续线条功能的制作 …… 119
8-1 前言 …… 119
8-2 范例程序 …… 119
8-3 程序分析 …… 123

第九章 矩形功能的制作 …… 128
9-1 前言 …… 128
9-2 范例程序 …… 129
9-3 程序分析 …… 133

第十章 实心矩形功能的制作 …… 138
10-1 前言 …… 138
10-2 范例程序 …… 138
10-3 程序分析 …… 143

第十一章 橡皮擦功能的制作 …… 147
11-1 前言 …… 147
11-2 范例程序 …… 147
11-3 程序分析 …… 151

第十二章 填图功能的制作 …… 155
12-1 前言 …… 155
12-2 范例程序 …… 155
12-3 程序分析 …… 159

第十三章 喷枪功能的制作 …… 165
13-1 前言 …… 165
13-2 范例程序 …… 165
13-3 程序分析 …… 168

第十四章 画圆功能的制作 …… 172
14-1 前言 …… 172
14-2 范例程序 …… 173
14-3 程序分析 …… 178

第十五章 写字功能的制作 …… 185
15-1 前言 …… 185
15-2 范例程序 …… 186
15-3 程序分析 …… 192

第十六章 区块文件功能的制作 …… 199
16-1 前言 …… 199
16-2 范例程序 …… 200
16-3 程序分析 …… 217

第十七章 打印功能的制作 …… 232
17-1 前言 …… 232
17-2 范例程序 …… 232
17-3 程序分析 …… 237

第十八章 文件存取功能的制作 …… 243
18-1 前言 …… 243
18-2 范例程序 …… 244
18-3 程序分析 …… 251

第十九章 图标【ICON】的制作 …… 259
19-1 前言 …… 259
19-2 范例程序 …… 260
19-3 程序分析 …… 267

第二十章 组合绘图系统功能 …… 274
20-1 前言 …… 274
20-2 简介函数及分析新函数 …… 274
20-3 完整程序 …… 287

第一章　C++的基本概念

1-1　前　　言

我们写这本书是以专题研究的方式进行，一步一步的来制作一个绘图系统。若读者已学会C++程序语言则可以很快地浏览第1章和第2章。大部分的人都是先学C语言后才学C++语言，因为C++语言是C语言的扩充，基本语句【例如for循环、while循环、if语句】的用法都是一样。但是C++增加了新的命令，写程序的风格也有变，C++是以面向对象程序设计的观念来设计程序的。

这本书对没有学过C++语言的读者也适用。因为我们所介绍的C++的基本概念及特点足够用来写我们所要写的绘图系统。在此并没有深入探讨C++的所有特点，若想要进一步了解C++则可参考《Borland C++ 3.1——软件的IC不是梦》一书。以下将逐步介绍C++的特点。

1-2　C++新增的命令

一、注解

C++的注解是以 // 符号开始止于该行尾端。例如下面所示：

```
//从这里起是语句的注解
```

在C语言里所使用的注解是以 /* 符号为开头，以 */ 为注解结束的符号，两符号之间的语句便是注解。在C++的COMPILER中这两个注解可以混合使用。// 符号适用于只有一行的注解，/* …… */ 符号适用于有两行以上的注解。在这里举一个范例程序让读者了解在什么情况下应选用那一个注解。

```
/* File exam1.cpp
 * This is an example of how to use comment,
 * it will send a message to screen.
 */
#include <stdio.h> //include standard I/O library
//here is a main function
int main ()
{
 printf(" Send a good message to screen \n");
 /* we declare main function is an int function
  * so we need to return a value.
```

```
 */
return 0; // return a value
}
```

二、输入输出

C++提供了在文字方式下的新输入输出命令。它们分别是 cout 和 cin。两者都定义在 iostream.h 标题文件。调用格式如下所示：

```
char Name[30];
cout << "Input your name please !";
cin  >> Name;
cout << "Your name is " << Name;
```

cout 的发音为 see out
cin 的发音为 see in

以上的格式看起来很简洁，但在应用上比 printf()命令好用。printf()命令中若要作输出时必须事先说明所要输出的参数类型是什么类型，在 cout 命令中并不需要说明其类型，cout 自己会判断所要输出的是什么类型。同样的 cin 命令也不需要给定所要输入的类型是什么类型，只要指定变量就可以了，cin 自己会判断输入的类型是什么类型。我们来看看一个范例程序 exam2.cpp。

```
// File name = exam2.cpp
// It is a program to test cin and cout
#include <iostream.h>

void main()
{
    char Name[30];
    unsiged age;
    cout << " Input your name please ! \n";
    cin  >> Name;
    cout << "Input your age please ! \n";
    cin  >> age;
    cout << " Your name is " << Name <<"\n"
         << " Your height is "<< 1.95
         << "meter";
}
```

输出的结果：

```
Input your name please !
John
Input your age please !
20
Your name is Joun
You are 20 years old.
your height is 1.95 meter.
```

从范例程序中可以看出来，输出输入并不需要给定参数的类型。在这里 cout 和 cin 的输入输出符号【>>和<<】和右移位，左移位的符号是一样的，为什么结果不一样呢？因为在 C++中运算符是可以重复定义的，所以只要重新定义运算符，运算符就代表新的意义。

当我们输出浮点数时，若浮点数的小数点是 0，则只会输出整数部分，并不会把小数点同时输出。例如下面所示：

```
cout << 10.00;
cout << 10.01;
```

输出结果：

```
10
10.01
```

此外，C++还提供了许多新的输出操纵函数(Manipulators)，如下表所示。

操纵函数	输出/输入	含　　义
endl	O	输出新的一行，等于"\n"
ends	O	输出结尾值，等于"\0"
flush	O	强迫立刻执行输出的动作
dec	I/O	十进位输出输入
hex	I/O	十六进位的输出输入
oct	I/O	八进位的输出输入
setbase(int i)	O	以 i 为进位输出【i=0,8,10,16。0 代表内定值】
setfill(char c)	I/O	以 c 填满所设宽的空位
setw(int w)	I/O	设定数据流列宽为 w
setprecion(int pr)	I/O	设定小数点精确度为 pr

应用范例

范例 1

```
#include <iostream.h>
void main()
  {
   cout <<"dec format :" << dec  << 10  << endl;
   cout <<"hex format :" << dex  << 10  << endl;
   cout <<"oct format :" << oct  << 10  << endl;
  }
```

输出结果

```
dec format: 10
hex format: a
oct format: 12
```

范例 2

```
#include <iostream.h>
```

```
void main()
  {
   cout <<setfill('$')<<setw(10) << 10 << endl;
   cout <<setfill('#')<<setw(10) << 10;
  }
```

输出结果

```
$$$$$$$$10
##########10
```

三、内存的分配

在讨论分配内存前先思考一下为什么要分配内存？这是刚学习程序语言时很想知道的一件事。事实上分配内存是为了使程序执行更快，因为我们把变量存放在内存后若要用到它时只需在内存上找就可以找到。在内存上找比在磁盘上找快很多。在处理图像文件或图形文件时分配内存是不可缺的工作。若图形存放在内存的话，要显示时只需从内存拷贝一份到屏幕上即可。若图形文件大于所能使用的内存时必须分两三次来读取文件，因此内存的大小也会影响显示图形文件的速度。

在 C 语言中常用 malloc()、alloc()、farmalloc()、faralloc()函数分配内存，每次使用所分配的内存后必须释放供以后需要时使用。释放内存的函数是 free()和 farfree()函数。

C++提供了新的分配和释放内存函数。新函数用起来比较方便，它们分别是 new 和 delete 命令。

new： 配置内存，若配置内存成功的话返回指向该内存的指针，若失败则返回空指针。

delete： 释放 new 函数所配置的内存。

使用范例

```
int *Num;
Num = new int;
```

以上的语句是分配两个 BYTE 给 Num。new 命令后面的类型是我们所要分配内存的类型，其大小便是分配到的内存的大小。若要分配更多的内存时在类型的后面必须给定所要分配内存的大小，大小的值放在[ ]内。请看下一个范例：

若我们想分配十个 int 类型时可以用以下的方式来分配。

```
Num = new int[10];
```

以上的语句共配置了 20 个 BYTE。

分配内存的同时可以给定初值，其方法如下：

```
Num = new int(10);
```

此时 Num = 10;

注意以下的方法是错的，

```
Num = new int[10](10); //错误的示范,数组不能给定初值。
```

若要分配字符则必须说明如下所示：

```
char *Str;    Str = new char;
```

若要分配一百个字符则，

```
Str = new char [100];
```

分配内存的同时可以给定初值，其方法如下：

```
Str = new char ('I');
```

此时 Str = 'I';

注意以下的方法是错的，

```
Str = new char[100]('I'); //错误的示范,数组不能给定初值。
```

以上示范了很多种内存分配的方式，不管以何种方式分配内存，释放内存的时候都是一样的。请看范例：

分配内存时

```
Num = new int;
```

释放内存时

```
delete Num;
```

分配内存时

```
Num = new int[10];
```

释放内存时

```
delete Num;
```

分配内存时

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

释放内存时

```
delete Num;
```

分配内存时

```
Str = new char[100];
```

释放内存时

```
delete Str;
```

请读者把这些弄清楚，因为我们所要设计的绘图系统皆会使用 new 和 delete 命令。

1-3 类、对象与数据封装

本节要介绍的是面向对象程序设计的精髓之一，类(class)、对象(object)与数据封装。C++提供很多类，对象及数据封装的特性。我们在这里只作介绍而不深入讨论类和对象的特性，但足够让您用来设计绘图系统程序。

一、类(class)

类就是使用者所定义的数据类型。例如 int,char,long 都是内定数据类型,它们用来区分不同的数据类型。现在我们来看一个例子:

假设我们想定义一个新的数据类型 STUDENT,其功用是记录个人数据,可以记录姓名及生日。其定义方法如下所示:

```
Class STUDENT {
  private:
        char Name[80]     //名字
        unsigned int dd;  //日期
        unsigned int mm;  //月份
        unsigned int yy;  //年次
  public:
        void InputData();
        void ShowData();
        };
```

一个类的实例

在后面会做进一步的解释其内容,代表意义及用途。现在先来看看对象的定义。

二、对象(object)

凡经由类定义的变量都称为对象。类和对象是用在一起的,对象的定义方式和定义 int,char 类型变量是一样的。

若我们要定义一个整数类型时可以定义如下:

```
int Number;
```

同样道理定义一个对象时可以定义如下:

```
STUDENT John;
STUDENT 是类,John 对象
```

三、数据封装(Data Encapsulation)

数据封装是指类中可以用的成员,将其存取和使用方式分类便称为数据封装。所以想要使用类中的变量或函数时必须透过规定的管道来使用。这样一来类中的数据不容易因疏忽被更改。这有什么好处呢?若程序不大的话不容易看出来其优点,假设程序范围很大,那所用到的变量也会增加。无论在何处都可以使用所说明的变量函数的话可能在不知不觉的状况下会把某个变量的值更改过,因而产生使用者不想要的值。这样的错误不容易找出来,所以类提供了此最佳的防备方法,请善加利用之。

类的成员

类中的数据成员皆称为该类的成员。成员可分成两类:

(1) 数据成员(data members):例如前面例子中的 Name,dd,mm,yy。

(2) 函数成员(member functions):例如前面例子中的 InputData(),ShowData()。

数据成员在 private 区,是私有的变量,使用这些变量需通过函数成员来存取。

S.dd =14;
S.mm=05;
S.yy =94;
} 不能直接由对象 S 直接存取

函数成员是在 public 区,表示可以由对象来自由存取,我们以一个例子来说明函数成员和数据成员之间的关系。

```
//
//     File name = exam3.cpp
//
#include <conio.h>
#include <stdio.h>
#include <iostream.h>
  class STUDENT {
    private:
            char Name[80];
            unsigned int dd;
            unsigned int mm;
            unsigned int yy;
    public:
           void InputData();
           void ShowData();
           };←————————说明类结束前必须加分号
void STUDENT::InputData()
    {
    cout << "Input put your name..";
    gets(Name);
    cout << "Your birthday (e.g 12 5 63) ";
    cin  >> dd>>mm>>yy;
    }
void STUDENT::ShowData()
    {
    cout << "Your name is "<<Name<<endl;
    cout << "Your birthday is "<<dd<<'-'<<mm<<'-'<<yy;
    }
void main ()
    {
    STUDENT S;
    clrscr();
```

```
S. InputData();
S. ShowData();
getch();
}
```

例题 exam3. cpp 中可以看出来函数成员可以直接使用成员数据。在这里我们要求输入名字时是用 gets()函数,也可以用 cin 命令要求输入名字,为什么我们不用呢? 先来分别两者之间的不同才会知道为什么不用 cin 命令。gets()函数读取字串到Enter键为止,其中包括空白键在内。

例如:Michael Thorne ↵

读取的字串为 Micheal Thorne

cin 命令读取字串到Enter键,若其中包括空白键在内则到空白键而已。

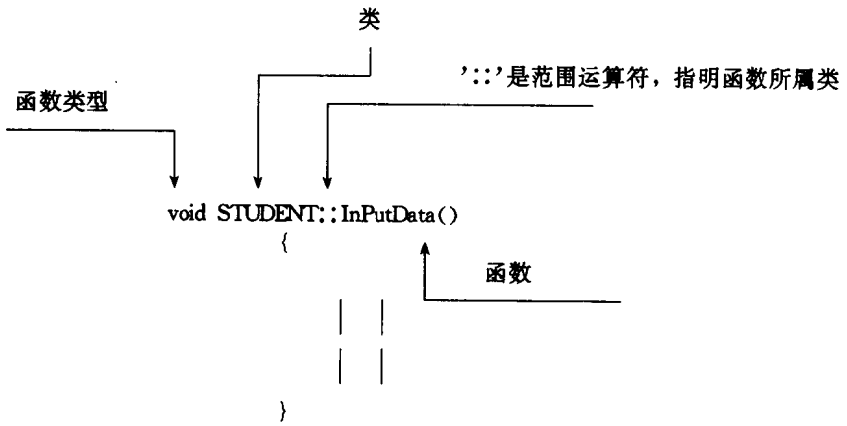
例如:Michael Thorne ↵

读取的字串为 Micheal

到目前为止我们已知道两种使用数据成员和函数成员的方式。

函数成员的定义

函数成员的定义方式如下:



函数也可以在类内定义,请看其定义方法:

```
//
// File name = exam4. cpp
//
#include <conio. h>
#include <stdio. h>
#include <iostream. h>
  class STUDENT {
    private:
          char Name[80];
          unsigned int dd;
          unsigned int mm;
          unsigned int yy;
```

```
    public:
        void InputData()
       {
        cout << "Input put your name..";
        gets (Name);
        cout << "Your birthday (e.g 12 5 63) ";
        cin  >> dd>>mm>>yy;
       }
        void ShowData()
       {
        cout << "Your name is "<<Name<<endl;
        cout << "Your birthday is "<<dd<<'-'<<mm<<'-'<<yy;
        }
       }
void main()
    {
    STUDENT S;
    clrscr();
    S. InputData();
    S. ShowData();
    getch();
    }
```

类中的静态变量

如果说明静态变量则必须在使用前给定初值，给定初值方式如下：

```
class COUNT {
  private:
    int x,y;
    static int count;
  public:
    void GetData();
   };

int COUNT:: count= 0;
```

类中的指针函数定义方式

```
class COUNT {
  private:
    int x,y;
    static int count;
  public:
    void *GetData();
   };
                '*'必须在双冒号'::'的前面
    ————
    ↓
void *COUNT::GetData()
```

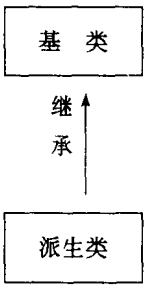
```
{
    | |
    | |
}
```

1-4 继承(Inheritance)

前面我们曾提到C++语言中的特性类和对象。这两个特性是面向对象程序设计的基石。现在我们要介绍的是C++语言的另外一个特性:继承。

什么是继承?继承的含义就是新定义一个类,新类可以继承使用旧类中的所有数据成员和函数成员。这样一来已经定义过的函数不需要再定义,只要继承就可以使用。

第一次定义的类我们称为基类,经由继承关系定义的类称之为派生类。



现在我们来看一个定义继承类的范例。

```
//
// File name = exam5.cpp
//
#include <conio.h>
#include <stdio.h>
#include <iostream.h>

      class MONEY {
    Private:
      unsigned int money;
    Public:
      void Input();
      void Show();
  };
void MONEY:: Input()
      {
      cout <<"Minput your money : ";
      cin >> money;
      }

    void MONEY:: Show()
      {
      cout << "Your money $"<<money;
      }
```