你问我答系列丛书

Visual C++

应用技巧与常见问题

冉光志　陈旭春
练　锴　董宇涵　等编著

机械工业出版社
CHINA MACHINE PRESS

你问我答
系列丛书

Visual C++应用技巧与常见问题你问我答

冉光志　陈旭春
练　锴　董宇涵
等编著

机械工业出版社

本书深入详尽地阐述了利用 VC 在 Windows 平台下进行编程的常见问题及高级技巧。全书共分四部分。第一部分对 C++语言进行了回顾，同时对 C++语言应用中的各种问题及技巧进行了总结，能使读者对 C++的掌握上一个台阶。第二部分总结了 Visual C++开发工具及一些辅助工具的使用技巧。第三部分通过示例展示了如何在 Windows 9x 和 Windows NT/2000 下进行系统编程。第四部分涉及 MFC 编程的各个方面，包括 MFC 内部机制剖析，高级界面设计，多线程编程，图形图像编程以及网络编程等。全书内容翔实，分析深入，实例丰富，具有很好的参考价值和指导性。适合各类编程人员及计算机爱好者阅读。

图书在版编目（CIP）数据

Visual C++应用技巧与常见问题你问我答/冉光志等编著. －北京：机械工业出版社，2003.4

（你问我答系列丛书）

ISBN 7-111-11815-4

I. V… II. 冉… III. C 语言—程序设计—问答 IV. TP312-44

中国版本图书馆 CIP 数据核字（2003）第 016915 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策　　划：胡毓坚

责任编辑：陈振虹

责任印制：路　琳

北京蓝海印刷有限公司印刷·新华书店北京发行所发行

2003 年 4 月第 1 版·第 1 次印刷

787mm×1092mm 1/16·22.75 印张·562 千字

0 001－5000 册

定价：34.00 元

凡购本图书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话（010）68993821、88379646

封面无防伪标均为盗版

出 版 说 明

随着知识经济的发展、科技的不断进步，计算机已经应用到我们日常生活的各个领域，人们已经清醒地认识到掌握计算机对未来个人发展的重要性。

作为现代人，不懂计算机，就是现代生活的“文盲”，这种说法毫不夸张。社会存在，竞争就存在。在优胜劣汰的激烈竞争中，掌握了计算机这门工具，就等于在生存竞争中增加了一块重要的砝码。

许多读者在学习各种计算机知识的过程中经常会遇到各种问题。基于此，我们编写这套《你问我答系列丛书》，旨在帮助读者及时、迅速地解决所遇问题，并从中得到一些解决问题的技巧。

本套书分为基础类、图形图像类和编程类。基础类适用于初学者，包括操作系统、办公软件、五笔字型输入法、网络知识、注册表、电脑故障等。基础类基本涵盖了初学者在计算机学习和使用中所涉及的各方面知识，读者可根据我们对问题的分类有针对性地阅读；图形图像类和编程类适用于有一定基础知识的读者，包括 Photoshop、3DSMAX、Flash、AutoCAD、Visual Basic、Visual C++、Visual FoxPro、Delphi、PowerBuilder 等，在书中不介绍软件的使用，而是介绍一些在软件使用过程或作品制作过程中遇到的问题和应用技巧，是各种实践经验的荟萃。此丛书问题前带有★★★的是典型的问题或常用技巧，这类问题是学习相关内容时经常遇到的问题，是必须解决的；问题前带有★★☆的是比较典型的问题；问题前带有★☆☆的则是可以作一般性了解的问题或小技巧。

本套书内容均以问答形式讲述，读者可以有针对性地去查找，不必为了某一问题翻阅全书，只要了解一些基本的知识或操作方法，即可现用现查，从容解惑，由此也将为读者赢得时间。

总结众多经验，解决身边问题，愿我们的这套丛书能在计算机学习和使用中助您一臂之力。

机械工业出版社

前　言

Visual C++（以下简称 VC）工具由于其强大的功能和灵活的开发方式而在实际开发中得到了广泛应用，但是要深入掌握 VC 开发工具却不是一件容易的事，必须经过不断的实践，掌握常见的开发技巧。

本书介绍 VC 实际应用中的常见问题和开发技巧，主要面向有一定基础的中高级读者。读者应当了解 C/C++语言，并且会使用 VC 工具。书中所介绍的开发工具版本主要是 VC 6.0。虽然微软已经推出了 Visual C++.NET，不过后者只不过是 VC 6.0 的升级版本，同时加入了一些新功能，二者在很多方面都是相同的，因此针对 VC 6.0 的开发技巧同样适用于 Visual C++.NET。

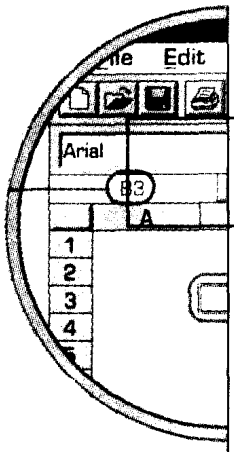
全书共分为四个部分。第一部分即第 1 章，总结了 C++语言编程的常见问题和技巧。C++语言本身就博大精深，容易学却难用好，因此书中不可能进行全面的总结，只是列出了一些典型的问题。第二部分即第 2 章，总结了 VC 工具的高级使用技巧。工欲善其事，必先利其器。要熟练地使用 VC 进行开发，必须先掌握好开发的工具。第三部分为第 3 章至第 5 章，内容集中在 Win32 API 编程方面。书中并没有详细介绍 API 函数，而是介绍这些函数的应用技巧。第 3 章和第 4 章主要介绍了 Windows 9x 和 Windows NT/2000 平台下的系统编程，某些内容是与特定系统有关的；第 5 章则介绍了如何用纯粹的 Win32 API 构造应用程序。第四部分包括第 6 章至第 12 章，主要内容是 MFC 编程。第 6 章回顾了 MFC 类库和 MFC 关键机制；第 7 章重点探讨文档/视图结构；第 8 章介绍菜单和工具栏的应用技巧；第 9 章为对话框与通用控件的使用；第 10 章为 MFC 多线程编程；第 11 章为 MFC 图形图像编程，介绍 MFC GDI 编程和 OpenGL 编程技巧；第 12 章为 MFC 网络编程，包括 WinSock、WinInet 和 RAS 编程等内容。全书讲解清晰，实例丰富。

为了使读者阅读，我们把问题作了重要程度的分类，凡是问题前带有★★★的表明是最重要的或最典型的；带有★★☆的则是比较重要的；而带有★☆☆号的则是可以一般了解的。读者可以根据需要来选择阅读。

本书由冉光志主持编写，参加编写工作的还有陈旭春、董宇涵、练锴、杜吉祥、张小毅、张发强、张浩、帅荣、马晓洪、姜岩松、刘武、王芳、唐明、王凯、王丽、李克兢、马本生、常妍、何冰冰等。由于作者知识和水平有限，错误和疏漏在所难免，恳请读者批评指正。

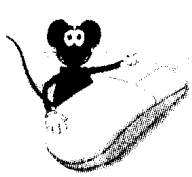
编　者

此为试读，需要完整PDF请访问：www.ertongbook.com



目 录

CONTENTS



此为试读，需要完整PDF请访问：www.ertongbook.com

第1章 C++语言基础

1.1 奇妙的 const

如何用 const 修饰变量

用 const 修饰变量将限定变量为只读，该变量值不允许被改变。下面是 const 用于定义数组大小的一个简单例子：

```
const int ArraySize = 10;
int array[ArraySize];
```

const 变量必须初始化，这个一次性的初始化是设置其数值的惟一机会。

```
const int i;    //错误，没有初始化
```

const 也可以用于修饰指针。考虑下面几种情况：

```
char chA = 'A';
const char cchB = 'B';
char *const ptr_1 = &chA;          // 常指针
const char *ptr_2 = &cchB;         // 指向常量的指针
char const *ptr_3 = &cchB;         // 指向常量的指针
const char *const ptr_4 = &cchB;   // 指向常量的常指针
```

当 const 与*并存时，正确区分的原则是：如果 const 位于*前，则指针本身是不能被修改的（常指针），至于指针所指向的变量，则可能允许被修改；如果 const 位于*后，则指针所指向的变量是不能被修改的，即该指针是指向常量的指针，至于指针本身，则可能允许修改。而 const 与类型标识的位置关系则无关紧要，所以 ptr_2 和 ptr_3 是一样的。依据这一原则，很容易判断出下面的操作是否合法。

```
ptr_1 = &cchB;        //错误
*ptr_1 = cchB;        //正确

ptr_2 = &chA;         //正确
*ptr_3 = chA;         //错误

ptr_4 = &chA;         //错误
*ptr_4 = chA;         //正确
```

const 也可以用于修饰引用，但不能用引用改变所指 const 变量的值。

const 定义常量对象的引用，可以用常量初始化，也可以用变量初始化。如果用变量初始化 const 型的引用，则引用成为该变量的只读别名。也可以用常量表达式初始化 const 的引用，但不可以用常量表达式初始化变量的引用。

```
const int i=1;
int j=2;
const int& m=i;
const int& n=j;

int& k=i + j;              // 错误
const int& p=i + j;        //正确
```

定义本身为常量的引用变量是无意义的(虽然合法)，因为所有引用变量自动成为常量(因为一旦引用初始化，就不能再引用其他变量)：

```
int n;
int& const k=n;   //Legal but meaningless
```

如何用 const 修饰函数

const 可以修饰函数的参数和返回值。

1. const 参数

如果函数是值传递的，const 将保证参数在函数中不能被修改。如：

```
void f (const int i) {
      i++;  // 错误   }
```

如果函数是地址传递的（传递的是指针或引用），const 将保证该地址内容不被修改。如：

```
Void f(const int *i) {
      *i=3;  //正确   }
```

2. const 返回值

让函数返回 const 可以在不降低效率的情况下减少用户出错的概率。const 返回值有以下几种情形：

```
const var_type operator*(const var_type& a, const var_type& b );
const class_type a;
const int * f ( ){
      static int i;
      return &i;
}
```

让函数返回一个常量值：可避免用户修改函数返回值。可防止以下错误行为的发生：（a*b）=c。

让函数返回一个 const 对象：返回的 const 对象的值将不能作为左值使用，不能被赋值，也不能被修改。

让函数返回一个指向 const 的指针：则该地址的内容不会被修改。

还需说明的是，在 C++中，若返回常量类型，则只有使用常量类型的变量才能接受此类函数的返回值。

怎样在类中使用 const

类对象定义中加入 const 表示不能改变该类中的任何数据成员的值。

1. 类的 const 成员变量的初始化

类的 const 成员变量必须初始化，这与类的其他类型成员变量不同。

与类的一般成员变量初始化方法相同：不允许在类定义时初始化，而是使用 C++提供的成员初始化器表来初始化一个或几个数据成员。成员初始化器表放在构造函数定义中参数表括号的紧后面，包括冒号和一个或几个用逗号隔开的成员初始化器。成员初始化器包括数据成员名和括号内的初始化值(构造函数中的成员初始化器表部分只在构造函数的定义时加入，而不在构造函数说明时加入)。例如，下面类中的构造函数包含一个成员初始化器表，每个数据成员有一个成员初始化器。

```
class MyClass
{
    private:
        const int Cint1=5;     //错误，不能初始化
        const int Cint1;       //正确//
        const int& Cint2;
        ...
    public:
        MyClass(int n): Cint1(5),Cint2(n)
        {
            printf("Cint2=%d\n", Cint2)
        }
        ...
};
int main(int argc, char* argv[ ])
{
    class MyClass *myclass1=new MyClass(8);
    return 0;
}
```

程序运行结果：Cint2=8。

2. 调用 const 对象的成员函数

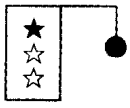
由于通常成员函数可以改变一个或几个数据成员值，编译器不允许调用 const 对象的成员函数。

为了调用 const 对象的成员函数，必须在函数定义中包括关键字 const。const 关键字放

在形参表的括号之后，此种形式的函数定义只能用于类的成员函数，它表示该函数不能改变任何数据成员，否则编译器在编译函数码时会发生错误。

```
class my_class2
{
    ...
    public:
    void f(int *m, ,int * n)const
    {
        *m=i;
        *n=j;
    }
    ...
};
```

最好把所有不改变数据成员的成员函数都声明为 const 成员函数，使类的用户可以随意调用 const 对象的函数。



const 与#define 有什么区别

C++中引入了 const 关键字，使用 const 可以替代 C 语言中用于定义常数的宏#define。

1．类型检查

预处理器指令#define 建立宏时做的是值代替（即文本代替），没有类型检查的工具。而 const 提供了类型检查，可以避免预处理器做值代替时出现的一些问题。

2．内存分配

#define 不占用内存单元。C++编译器通常并不给 const 常量分配存储空间，而是把 const 变量的定义保存在符号表里。在 VC 中，const 变量与一般变量相同，都分配内存空间。

此外使用 const 可以减少不必要的内存分配，见下面的例子：

程序清单 1-1 const 变量与宏在内存分配上的不同

```
#include "stdio.h"
#define str1 "This is the first string!\n"
void main(void)
{
    const char str2[ ]="This is the second string!";
    printf(str1);    //为 str1 第一次分配内存
    printf(str2);    //为 str2 一次分配了内存，以后不再分配
    printf(str1);    //为 str1 第二次分配内存
```


此为试读，需要完整PDF请访问：www.ertongbook.com