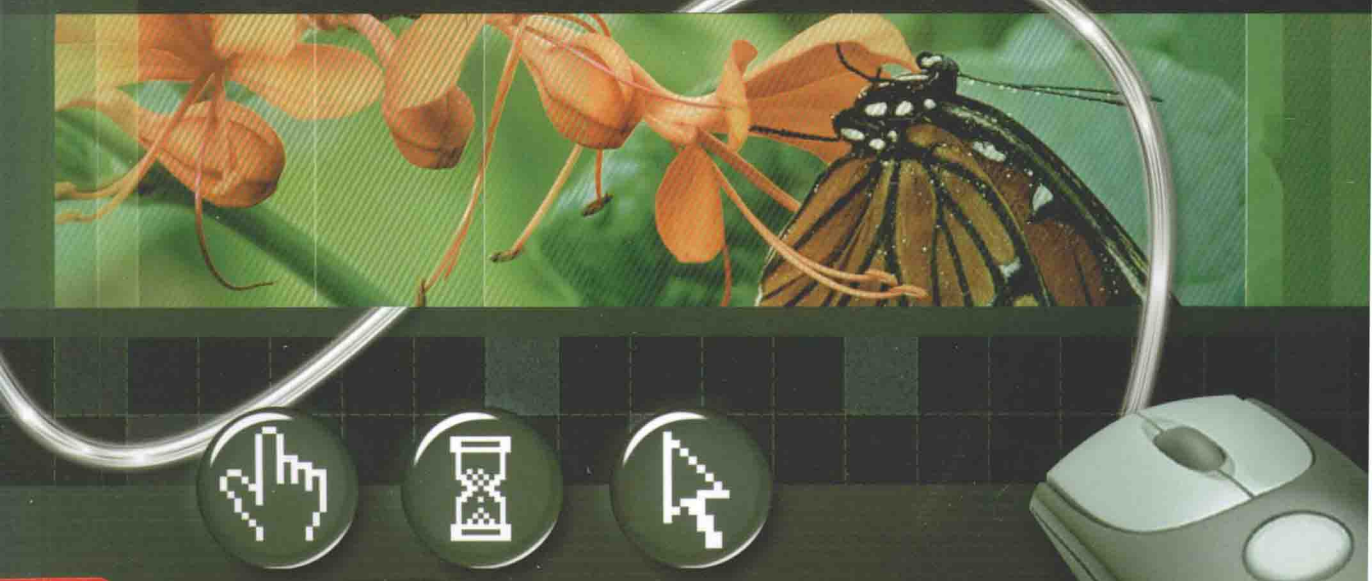
大学计算机规划教材

C++ 程序设计基础（第5版）（下）

——.NET 环境下的 Windows 应用程序设计

◆ 徐红云　周霭如　黄小兵　编著



中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

大学计算机规划教材

C++程序设计基础

（第5版）（下）

——.NET环境下的Windows应用程序设计

徐红云　周霭如　黄小兵　编著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本书的例程以 Microsoft Visual Studio 2010 为运行环境，全书分为上、下两册：上册《C++程序设计基础（第5版）（上）》（ISBN 978-7-121-28595-0）介绍 C++程序设计的基础知识，下册介绍 VC++在.NET 环境下的 Windows 应用程序设计。

在第 4 版的基础上，下册增加了一个贯穿各章的应用案例——简易教务管理系统的设计与实现。

下册共 7 章，主要内容包括：C++/CLI 托管编程、控制台程序设计、常用控件、复杂界面设计、图形图像应用、数据库应用、网络编程。应用案例的介绍分散在以上各章中，第 1 章介绍案例的总体设计，第 2 章介绍案例登录界面的设计与实现，第 3 章介绍学生信息管理部分模块的设计与实现（包括类的设计与实现），第 4 章利用复杂界面的相关知识对系统进行优化，第 5 章对系统中的统计数据进行可视化，第 6 章将系统的数据存储改用数据库实现，第 7 章介绍了反馈信息处理功能的设计与实现。

本书免费提供配套的电子课件和例程的源代码，请登录华信教育资源网（www.hxedu.com.cn）注册后下载。

本书可以作为高等学校计算机类、信息类、电类专业本科生高级语言程序设计课程教材，也可以作为教师、学生和 C++语言爱好者的参考书。

图书在版编目（CIP）数据

C++程序设计基础. 下，.NET 环境下的 Windows 应用程序设计 / 徐红云，周霭如，黄小兵编著. —5 版.
—北京：电子工业出版社，2016.8
大学计算机规划教材
ISBN 978-7-121-29600-0

Ⅰ. ①C… Ⅱ. ①徐… ②周… ③黄… Ⅲ. ①C 语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2016）第 179686 号

策划编辑：冉 哲
责任编辑：冉 哲
印 刷：三河市鑫金马印装有限公司
装 订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1 092 1/16 印张：21 字数：650 千字
版 次：2003 年 8 月第 1 版
2016 年 8 月第 5 版
印 次：2016 年 8 月第 1 次印刷
定 价：48.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：ran@phei.com.cn。

前　言

C++语言是优秀的计算机程序设计语言，它的功能相当强大。我们编写这本书的目的是，为没有任何程序设计基础的理工科大学生提供一本适用教材，使他们掌握从理论到实践都要求很高的C++语言。

一门课程的设置应该放在整个教学培养计划中统筹考虑。我们的教学目标不是马上培养一个会使用某种语言（例如C++语言）的程序员，而是强调对程序设计语言的理解和应用，“计算机语言”的角色是第一位的。所以，在教材编写和组织教学的过程中，我们力图通过对基本语法现象的剖析，由浅入深地让学生理解、掌握语言规则的原理，懂得用计算机指令的模式去分析和求解问题，并在机器上实现简单的程序。至于深入的算法及大程序的组织讨论，将由相关的后续课程（例如，数据结构、算法分析、计算方法、软件工程等）完成。因此，对高级程序设计语言规则的理解和应用是本教材编写的立足点。

我们根据多年从事计算机程序设计教学的经验，按照学生学习的认知规律，精心构造整本教材的体系和叙述方式，原则是：循序渐进、难点分散、通俗而不肤浅。本教材以语法范畴和程序组织为脉络，清晰渐进，从字、词、数据、表达式、语句，到函数、类，是语法范畴构成的基本脉络；在程序功能方面，则以组织数据和组织程序为另外一条基本脉络，并以渐进的、粒度扩大的方式逐步导入分析。

例如，数据的组织方式：基本数据类型—数组—结构—链表，体现如何利用基本数据类型根据需要组织数据；程序的组织方式：语句—函数—类，体现结构化思想和面向对象思想对程序不同的组织方式。

指针是C++语言的重要概念，是操作对象的有力工具。本书没有一般C语言、C++语言教材中专门的“指针”一章。我们从最简单的变量开始，建立对象的名和地址的概念，用对象的不同访问方式贯穿于各章节。从结构化程序设计到面向对象程序设计，采取了比较平滑的过渡。首先，在一开始介绍基本数据类型、程序流程控制、函数等结构化程序设计的基本知识时，就非正式地使用“对象”这个术语（从计算机程序的角度，任何占有存储空间的实体都是对象）；继而，掌握结构到类的演变，给出对象的准确定义；进一步，展开介绍面向对象程序的几个基本特性，即封装、继承、多态和类属在C++语言中的实现方法。同时，我们在本书的阐述中体现一个思想：没有一种对所有问题都是最好的程序设计方法，对特定问题，选择合适的解决方案是程序员必备的素质。

本书之所以取名为《C++程序设计基础》，原因有二：第一，它不是一本C++语言手册，不可能包罗所有语法规则和特定版本提供的各种功能；第二，它没有涉及复杂的算法和工程化的面向对象分析设计方法。这两个问题与教材的定位相关。对第一个原因，我们认为学生在掌握了程序设计的基本概念和基本方法之后，可以通过语言平台（例如Visual C++）或者其他资料学习，拓展对语言功能的了解。我们在有关章节中，也做了类似的引导，例如，STL标准类库的介绍，这些内容提供给教师选择或学生自学。至于第二个原因，那些是计算机专业后续课程的教学内容。本书介绍的程序设计方法和使用到的算法都立足于基本概念和方法，所以，例程通常是简单和小规模的。

本书分别在2003年、2006年、2009年和2012年出版了第1～4版，目前为第5版。本书分为上、下两册。上册《C++程序设计基础（第5版）（上）》（ISBN 978-7-121-28595-0）介绍程序

设计的基础知识，在第 5 章中增加了一节讨论结构参数；第 11 章增加了文件应用的例程；全书配有按章节的同步练习和综合练习，以方便学生自学。下册介绍 Visual C++在.NET 环境下的 Windows 应用程序设计，增加了一个贯穿下册各章的应用案例——简易教务管理系统的设计与实现。

下册共 7 章，主要内容包括：C++/CLI 托管编程、简单 Windows 应用程序、常用控件、复杂界面、图形图像、数据库应用、网络编程。应用案例的介绍分散在以上各章中，第 1 章介绍案例的总体设计，第 2 章介绍案例登录界面的设计与实现，第 3 章介绍学生信息管理部分模块的设计与实现（包括类的设计与实现），第 4 章利用复杂界面的相关知识对系统进行优化，第 5 章对系统中的统计数据进行可视化，第 6 章将系统的数据存储改用数据库实现，第 7 章介绍反馈信息处理功能的设计与实现。

本书免费提供电子课件及全部例程和案例的源代码，请登录华信教育资源网（www.hxedu.com.cn）注册后下载。

本书可以作为高等学校计算机类、信息类、电类专业本科生高级语言程序设计课程教材，也可以作为教师、学生和 C++语言爱好者的参考书。

本书的编写过程，是作者不断向学生学习，向同行学习，向 C++学习的过程。在此，对所有使用本书的教师、学生，以及热心向我们提出宝贵意见的读者致以诚挚的感谢！希望继续得到读者的支持和帮助。

作　者

目　录

第 1 章　C++/CLI 托管编程

C++/CLI 是为.NET 平台特别设计的，它提供了既优雅又强大的新语法支持。VS.NET 通过新的编译模式确保.NET 框架 CLI 的一致性和可验证性，提供了 C++本机环境和 C++/CLI 环境的无缝合并，把.NET 的全部强大功能带给 C++，同时也把 C++强大的功能带给.NET。程序员可以非常自然地在.NET 环境下开发 C++应用程序。用托管代码编写 C++程序，不需要重新学习新的编程语言，只需要了解一些简单的规则而已。

.NET 框架提供的公共语言运行库的功能很庞大。本章主要通过控制台程序介绍 C++/CLI 基本语法，为了与后续 Windows 窗体应用程序衔接，编写了几个简单窗体应用程序。

1.1　.NET 框架

.NET 是微软公司于 2000 年推出的面向互联网时代构筑的新一代平台，.NET 框架（.NET Framework）简化了在高度分布式环境中开发应用程序的工作。一整套基于.NET 框架设计的工具组件，被集成到 Visual Studio .NET（简称 VS.NET）开发环境中，包括 Visual Basic .NET、Visual C++ .NET、Visual C# .NET、Visual J# .NET、ASP.NET 等多种开发工具。Visual C++ .NET（简称 VC.NET）是在 C++基础上产生的，它的基本语法与 C++相同，并具有很多新的特性，增强了 C++的性能，使用起来也更加方便。

VS.NET 是一套完整的开发工具，各种语言都使用相同的集成开发环境（IDE），开发人员可以轻松地创建用混合语言设计的解决方案。

.NET 框架有两个主要的组件：公共语言运行库（CLR，Common Language Runtime）和.NET 框架类库（FCL，Framework Class Library）。公共语言运行库在执行时管理代码，提供核心服务（如内存管理、线程管理和远程处理），强制实施严格的类型安全以及可确保安全性和健壮性的其他形式的代码准确性。类库是综合性的、面向对象的、可重用类型的集合，可用来开发应用程序。

（1）公共语言运行库

CLR 是一个公共语言运行库。在.NET 中，编译生成不是最终的机器码，而是一种中间语言（MSIL，Microsoft Interlanguage）代码。运行时由 CLR 的实时编译器（JIT，Just In Time）“翻译”成特定的机器代码，然后执行。

CLR 支持多种语言，实现各种语言互操作。因此，CLR 提供了公共类型系统（CTS，Common Type System），对数据类型提供定义、管理和使用的严格说明；提供了代码必须遵守的公共语言规范（CLS，Common Language Standard），它包括许多应用程序所需要的基本语言功能。

通过 CLR 运行的代码称为托管（Managed）代码。在 CLR 控制之外的代码称为非托管（UnManaged）代码，通常称为本地代码。

通常所说的 C++，是 ISO/ANSI 标准的，为执行普通的应用程序——非托管的 C++而设计。而 C++/CLI 标准是专门为编写 CLR 托管程序而设计的，它是对 C++的一种扩展。CLI 的意思是通用语言结构（Common Language Infrastructure）。

所有 VB.NET 和 C#.NET 代码都是托管代码。但 VC.NET 提供了两种代码的互操作，允许程序员在项目中的不同程序模块之间，甚至同一个文件中，混合使用托管（C++/CLI）代码和本地（C++）代码，而不同语言编写的组件也可以进行交互。这体现了 VC.NET 功能的强大。

（2）.NET 框架类库

.NET 框架类库 FCL 是一个与公共语言运行库紧密集成、可重用的类的集合。该类库包括类、接口、结构和枚举，它们可以加速和优化开发过程并提供对系统功能的访问。.NET 框架类库使用 CTS 数据和 CLS 规范，是生成.NET 应用程序、组件、控件的基础。.NET 框架类库包含了将近 100 个命名空间。本书对涉及的命名空间会做简单说明。

1.2 简单控制台程序

在一个 VC.NET 应用程序中，既可以使用非托管代码，也可以使用托管代码。本节通过简单的控制台程序，初步认识托管代码和非托管代码的区别。

1.2.1 一个简单程序

【例 1-1】 在 VC.NET 的 IDE 中，建立一个 CLR 空项目，然后添加一个 C++文件（cpp），输入以下代码。

```
//托管代码使用的命名空间
using namespace System;
//非托管代码使用的头文件和命名空间
#include "iostream"
using namespace std;
//托管代码，计算正方形的周长和面积
void writeSquare()
{
    Double len;
    Console::Write("length of a side=");
    len=Double::Parse(Console::ReadLine());
    Console::WriteLine("girth of square={0:f}", 4*len);
    Console::WriteLine("area of square={0:f}", len*len);
}
//非托管代码，计算圆的周长和面积
void coutCircle()
{
    double r;
    cout<<"radius=";
    cin>>r;
    cout<<"girth of circle="<<2*3.14*r<<endl;
    cout<<"area of circle="<<3.14*r*r<<endl;
}
int main()
{
    writeSquare();
    coutCircle();
    return 0;
}
```

从上述程序中，很容易比较托管代码与非托管代码的区别。

（1）使用头文件和命名空间

托管代码中使用的各种数据类型，以及表示控制台应用程序的标准输入流、输出流和错误流的 Console 类都在 System 命名空间中声明，因此，需要语句

```
using namespace System;
```

而语句

```
#include "iostream"
using namespace std;
```

声明了非托管代码中输入、输出需要的头文件和命名空间。

如果在 IDE 中建立 CLR 控制台程序，则系统自动生成程序框架如下：

```
//CLR 控制台应用程序.cpp：主项目文件
#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{
    Console::WriteLine(L"Hello World");
    return 0;
}
```

程序员可以在这个框架上添加和修改代码。注意，这时的

```
#include "stdafx.h"
```

指令不能删除，并且要放在起始位置。

（2）数据类型说明

在函数 writeSquare 中，变量 len 被说明为 Double 类型。数据类型是 System 命名空间定义的结构。.NET 框架把 C++的类型关键字定义为对应结构的同义词。所以，在函数 coutCircle 中定义变量 r 类型为 double，它是 Double 的同义词。

（3）控制台数据输入

在非托管代码中，iostream 类预定义的标准输入、输出流对象 cin、cout 用于控制台程序的数据交互。而托管代码中，.NET 框架类库 System::Object::Console 类定义了控制台应用程序的标准输入流、输出流和错误流。

上述程序中，用 Console::ReadLine()方法获取输入的一行字符串，然后用 Double::Parse 方法把输入串转换为浮点数形式。Console 类的 Read 有以下几种形式。

- 从标准输入流读取下一个字符：

  int Read()

- 从标准输入流读取下一行字符：

  String^ ReadLine()

- 获取用户按下的字符或功能键，并显示在控制台窗口中。返回一个 ConsoleKeyInfo 对象，描述标准键常数和对应于按下的控制台键的 Unicode 字符：

  ConsoleKeyInfo ReadKey()

- 获取用户按下的字符或功能键，可以选择是否显示在控制台窗口中：

  ConsoleKeyInfo ReadKey(bool intercept)

（4）控制台数据输出

Console 类的 Write 方法和 WriteLine 方法用于输出数据，前者表示输出数据不换行，后者表示输出数据后插入行终止符。

输出方法有很多个重载版本，可以用于输出一个表达式项，也可以用于输出多个表达式项。若输出多个表达式项，则需要用格式字符串参数指定输出格式。语法形式如下：

Write(<字符串>, <输出对象列表>);

WriteLine(<字符串>, <输出对象列表>);

其中，<字符串>可以包含直接输出的字符串和用一对花括号“{ }”相括的格式串。例如，在以下代码中：

```
Console::WriteLine("girth of square={0:f}", 4*len);
```

字符串"girth of square={0:f}"中的"girth of square ="是直接输出的字符串，"{0:f}"称为格式串，第 1 个数字 0 表示该格式用于逗号之后的第一个输出对象，冒号之后的"f"表示用定点格式输出。如果不指示格式符，例如只用"{0}"，则系统自动选择最节省的输出格式。格式串之间用逗号分隔。<字符串>之后是<输出对象列表>，输出对象可以是表达式或具有输出值的对象名。

Console 类还有一些常用的方法，见表 1.1。这些方法的具体例程，请读者参阅 MSDN 资源。

表 1.1 Console 类其他常用方法

名　称	说　明
Beep	通过控制台扬声器播放提示音
Beep(Int32, Int32)	通过控制台扬声器播放具有指定频率和持续时间的提示音
Clear	清除控制台缓冲区和相应的控制台窗口的显示信息
ResetColor	将控制台的前景色和背景色设置为默认值
SetBufferSize	将屏幕缓冲区的高度和宽度设置为指定值
SetCursorPosition	设置光标位置
SetWindowPosition	设置控制台窗口相对于屏幕缓冲区的位置
SetWindowSize	将控制台窗口的高度和宽度设置为指定值

1.2.2 格式串

格式设置是指将类、结构或枚举值的实例转换为字符串的表示形式，或者把字符串形式的数据还原为原始数据类型。.NET 提供了丰富的格式设置支持，以满足开发人员的各种要求。

格式串用于设置各输出项的输出格式。各种数据类（例如，Int32 等）的 ToString 方法、Console 类和 StreamWriter 类的 Write 和 WriteLine 方法、String::Format 方法及 StringBuilder::AppendFormat 方法都支持格式串。

一个格式串可以包含多个用花括号相括的格式项。格式项的语法形式为：

{ <索引> [, <对齐>] [: <格式说明>] }

① <索引>

“索引”参数说明符是一个从 0 开始的数字，标识输出参数列表中对应项的序号。参数说明符为 0 表示输出列表的第一个对象，参数说明符为 1 表示列表的第二个对象，其余类推。

通过指定相同的“索引”参数，多个格式项可以引用对象列表中的同一个元素。例如：

```
Console::WriteLine("{0:X} {0:E} {0:N}", object);
```

分别将同一个 object 数值设置为十六进制数、科学记数法和数字格式输出。

每个格式项都可以引用列表中的任一对象。例如，如果有三个输出对象，则可以通过指定类似于"{1} {0} {2}"的复合格式字符串来设置第二个、第一个和第三个对象的格式。未被格式项引用的对象会被忽略。如果“索引”参数说明符的值超出对象列表范围的项，将导致运行时的异常。

② <对齐>

可选的“对齐”参数是一个带符号的整数，指示格式的字段宽度。

若“对齐”值小于设置了格式的字符串的长度，则“对齐”参数被忽略；

若“对齐”值为正数，则字段中设置格式的数据为右对齐；

若“对齐”值为负数，则字段中设置格式的数据为左对齐；

若需要填充，则使用空白。

③ <格式说明>

可选的“格式说明”参数是适合正在设置格式的对象类型的格式字符串。数值型的“格式说明”参数可以附带示数精度指示。表 1.2 中列出了三种标准格式说明符的形式。在后续的例程中，将会应用这些格式说明符输出各种类型的数据。实际上，.NET 还提供各种自定义格式和复合格式字符串，有需要的读者请查阅相关资源。

若不指定“格式说明”参数，则由系统根据对象类型选择输出格式。

若输出多个表达式的值，而不需要提示字符串，可以用这样的形式：

```
Console::WriteLine(" {0},{1},{2}", x, y, z);
```

此时，x、y、z 的值以逗号分隔输出，输出格式由 x、y、z 的类型决定。

如果 Write 方法和 WriteLine 方法只用于输出一个表达式的值，则可以省略格式串。例如：

```
Console::WriteLine(4*len);
```

表 1.2　标准格式说明符

格式说明符	说　明	示　例
标准数字格式字符串		
C 或 c	货币值	-123.456("C")→￥-123.46
D 或 d	十进制整型	1234("D")→1234 -1234("D6")→-001234
E 或 e	科学指数计数法	1052.0329112756("E")→1.052033E+003 1052.0329112756("e2")→-1.05e+003
F 或 f	定点	1234.567("F")→1234.57 -1234.56("F4")→-1234.5600
G 或 g	常规，用最紧凑方式表示数据	-123.456("G")→-123.456 123.4546("G4")→123.5 -1.234567890e-25("G6")→-1.23457E-25
N 或 n	带分隔符输出，默认精度	1234.567("N")→1,234.57
P 或 p	百分比，结果乘以 100 并显示百分比符号，默认精度	1("P")→100.00 % -0.39678("P1")→-39.7 %
R 或 r	数字与对应字符串转换	"123456789.12345678"("R")→123456789.12345678 -1234567890.12345678("R")→"-1234567890.1234567"
X 或 x	十六进制数	255("X")→FF -1("X4")→00FF
枚举格式字符串		
G 或 g F 或 f	如有可能，将枚举项显示为字符串值，否则显示当前实例的整数值	
D 或 d	以尽可能短的表示形式将枚举项显示为整数值	
X 或 x	将枚举项显示为十六进制值	
标准日期和时间格式字符串		
d	短日期模式	2016-7-3 16:01:41→2016-7-3
D	长日期模式	2016-7-3 16:01:41→2016 年 7 月 3 日
f	完整日期/时间模式（短时间）	2016-7-3 16:01:41→2016 年 7 月 3 日 16:01
F	完整日期/时间模式（长时间）	2016-7-3 16:01:41→2016 年 7 月 3 日 16:01:41
g	常规日期/时间模式（短时间）	2016-7-3 16:01:41→2016-7-3 16:01
标准日期和时间格式字符串		
G	常规日期/时间模式（长时间）	2016-7-3 16:01:41→2016-7-3 16:01:41
M 或 m	月/日模式	2016-7-3 16:01:41→7 月 3 日
O 或 o	日期/时间模式	2016-7-3 16:01:41→2016-7-3T16:01:41.2187500+08:00
R 或 r	RFC1123 规范模式	2016-7-3 16:01:41→Sun, 03 Jul 2016 16:01:41 GMT
S	可排序日期/时间模式	2016-7-3 16:01:41→2016-7-3T16:01:41

续表

格式说明符	说　明	示　例
标准日期和时间格式字符串		
t	短时间模式	2016-7-3 16:01:41→16:01
T	长时间模式	2016-7-3 16:01:41→16:01:41
u	通用的可排序日期/时间模式	2016-7-3 16:01:41→2016-7-3 16:01:41Z
U	通用完整日期/时间模式	2016-7-3 16:01:41→2016 年 7 月 3 日 8:01:41
Y 或 y	年月模式	2016-7-3 16:01:41→2016 年 7 月

注：① 表中圆括号内表示使用的输出格式说明符。

② 表中箭头“→”左边为要输出的数据，右边为按特定格式的输出结果。

1.3 C++/CLI 数据

在 C++的数据表示中，有基本数据类型、结构类型、指针类型等。这些数据类型在 C++/CLI 中都有对应的定义，并扩展了一系列的功能。

System 命名空间包含基本类和基类、模板类，这些类定义常用的值和引用数据类型、事件和事件处理程序、接口、属性和异常处理。这个命名空间包括应用程序使用的基础数据类型的类，例如：Object（继承层次结构的根）、Byte、Char、Array、Int32 和 String 等。

1.3.1 基本数据类型

.NET 框架内置了 C++基本数据类型的关键字，并把这些关键字预定义为 System 命名空间中托管类型的别名，当使用.NET Framework 类型编写代码时，可以应用 C++相应的关键字。

表 1.3 列出了 C++和 C++/CLI 类型标识符的对照，以及数据类型的取值范围。

表 1.3　C++与 C++/CLI 基本类型标识符的对照

C++类型	C++/CLI 类型	字　节	说　明
bool	Boolean	1	表示逻辑值 true 或 false
signed char	SByte	1	有符号整数，取值范围-128～+127
unsigned char	Byte	1	无符号整数，取值范围 0（Byte::MinValue）～255（Byte::MaxValue）
wchar_t	Char	2	16 位 Unicode 字符
Decimal	Decimal	16	128 位十进制数值
Short signed short	Int16	2	有符号整数，取值范围-32768～+32767
unsigned short	UInt16	2	无符号整数，取值范围 0～65535。不符合 CLS
int signed int long signed long	Int32	4	有符号整数，取值范围-2 147 483 648（Int32::MinValue）～2 147 483 647（Int32::MaxValue）
unsigned int unsigned long	UInt32	4	无符号整数，取值范围 0～4 294 967 295。不符合 CLS
_int64 signed _int64	Int64	8	有符号整数， 取值范围-9 223 372 036 854 775 808～+9 223 372 036 854 775 807
unsigned _int64	UInt64	8	无符号整数，取值范围 0～18 446 744 073 709 551 615。不符合 CLS

续表

C++类型	C++/CLI 类型	字　节	说　明
float	Single	4	单精度浮点数，取值范围-3.402823e38～+3.402823e38
double	Double	8	双精度浮点数，取值范围-1.79769313486232e308～+1.79769313486232e308
void	Void		为不返回值的方法（函数）指定返回值类型

C++/CLI 数据类型与 C++数据类型运算方式一致，并且重载了 Object 类的 Parse 方法，把数值数据转换成对应字符串。

Parse 方法将数字的字符串表示形式转换为等效指定类型的数值，语法形式如下：

<type> Parse(String^ s)

其中，<type>为 System 命名空间定义的数据类型。此方法可以用类或实例形式调用，例如：

```
Int16 t;
Double p1,p2;
t=Int16::Parse("618");
p1=Double::Parse("3.1415");
p2=p1.Parse("3.1415");
```

ToString 方法将实例的数值转换为字符串表示形式：

String^ ToString()

String^ ToString(String^ format)

其中，format 参数是指定格式的字符串。

图 1.1　数值与字符串的转换

【例 1-2】　数值与字符串的转换。程序运行结果如图 1.1 所示。

```
using namespace System;
int main()
{
    Int16 t;
    Double p1,p2;
    t=Int16::Parse("618");              //把字符串转换成整数
    p1=Double::Parse("3.1415");         //把字符串转换成浮点数
    p2=p1.Parse("3.1415");
    Console::WriteLine("t={0:d}\np1={1:f5}\np2={2:f5}",t,p1,p2);
    String ^s1=p1.ToString();           //把浮点数转换成对应字符串
    Console::WriteLine(s1);
    String ^s2=p2.ToString("e");        //按格式转换浮点数
    Console::WriteLine(s2);
    Console::Read();                    //停留在控制台显示，按任意键时结束程序
}
```

【例 1-3】　格式化输出负整数和浮点数。程序运行结果如图 1.2 所示。

图 1.2　格式化输出负整数和浮点数

```
using namespace System;
int main()
{
    Console::Clear();
    Console::WriteLine("格式化输出负整数和浮点数");
    Console::WriteLine(
        "(C) Currency: .....{0:C}\n" +
        "(D) Decimal: .....{0:D}\n" +
        "(E) Scientific: .....{1:E}\n" +
        "(F) Fixed point: .....{1:F}\n" +
        "(G) General: .....{0:G}\n" +
```