C#技术及应用

韦鹏程　朱盈贤　石　熙　著

科学出版社

韦鹏程　朱盈贤　石　熙　著

科学出版社
北京

内 容 简 介

本书主要以案例触发知识点的方式，结合 C#实际开发中需要掌握的要点，对 C#语言及其应用进行比较深入、全面的剖析。

本书共 9 章，第 1～5 章介绍 C#语言和 WinForm 应用程序开发的基础理论，包括 C#语言基础，数组、自定义类型与泛型，面向对象程序基础，继承与多态性，委托与事件等；第 6～9 章介绍开发 WPF 应用程序的技术，包括初识 WPF 应用程序、WPF 控件、资源与样式控制、使用 ADO.NET 进行数据库编程等。

本书既可作为高等院校本科计算机专业的教学用书，也可供具有编程经验的相关从业人员，以及想要学习 C#并注重实践能力的各界人士参考使用。

图书在版编目(CIP)数据

C#技术及应用/韦鹏程，朱盈贤，石熙著. —北京：科学出版社，2017
ISBN 978-7-03-053286-2

Ⅰ. ①C… Ⅱ. ①韦…②朱…③石… Ⅲ. ①C 语言-程序设计
Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2017）第 128642 号

责任编辑：吕燕新 李 海 陈将浪 / 责任校对：王万红
责任印制：吕春珉 / 封面设计：东方人华平面设计部

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

北京京华虎彩印刷有限公司 印刷
科学出版社发行 各地新华书店经销

*

2017 年 8 月第 一 版 开本：B5（720×1000）
2017 年 8 月第一次印刷 印张：20
字数：401 000

定价：82.00 元

（如有印装质量问题，我社负责调换〈京华虎彩〉）
销售部电话 010-62136230 编辑部电话 010-62135927-2014

前　言

C#语言是一种完全面向对象的基于.NET 的编程语言，随着.NET 的普及，C#语言已经成为开发基于.NET 的企业级应用程序的重要语言。

本书以 Visual Studio 2012 为开发工具，每章均以案例入手，让读者突破传统的思维方式，深入浅出地探究 C#程序设计的基础知识和编程方法。每章通过案例展开对相关技术的探讨，特别注重实践与理论的结合。

本书共分 9 章，各章的主要内容如下。

第 1 章主要介绍 C#语言基础，包括 C#控制台应用程序、C#窗体应用程序、C#的基本数据类型、运算符与表达式、流程控制语句等，是理解和学习后续章节的基础。

第 2 章主要介绍数组、自定义类型与泛型，包括对编程中常用的数组类型、数组列表、控件数组等的探讨，以及对结构、枚举类型的介绍和对泛型的探讨。

第 3～5 章主要探讨 C#面向对象编程的理论和方法。其中，第 3 章着重讨论类的定义、类的成员等面向对象编程的基础理论，第 4 章就面向对象编程中的继承和多态性进行深入讨论，第 5 章主要探讨委托、事件等高级面向对象编程的方法。

第 6 章和第 7 章探讨 WPF 应用程序基本编程的方式及常用 WPF 控件的使用方法。

第 8 章介绍 WPF 中资源与样式控制的方法。

第 9 章主要探讨数据库及 ADO.NET 的相关理论，以及 LINQ 语法、数据绑定等。

本书由重庆第二师范学院韦鹏程教授、朱盈贤讲师和石熙博士撰写，并且得到了重庆市交互式教育电子工程研究中心和重庆第二师范学院交互式儿童电子产品协同创新中心的支持，在此表示感谢。

由于作者水平有限，加之时间仓促，书中不足之处在所难免，恳请广大读者批评指正。

目　录

第 1 章　C#语言基础

.NET Framework 是微软公司提出的一种新的软件开发平台，它简化了 Web 服务应用程序的开发。微软公司希望.NET 实现“多语言，单平台”，即让使用不同编程语言的人都可以对其进行访问，因此.NET 可以支持 Visual Basic、C#、JScript、J#等 20 余种语言。

C#是创建.NET 程序的语言之一，从 C 语言与 C++语言演化而来，它吸收了其他编程语言的优点并克服了它们的不足。利用 C#可以创建多种类型的应用程序，如 Windows 窗体应用程序、Web 应用程序、Web 服务等。

.NET Framework 有三个主要的组件：公共语言运行库（Common Language Runtime，CLR）、.NET Framework 类库（.NET Framework Class Libraries）和 ASP.NET。CLR 是.NET Framework 的基础，它运行代码并且在程序开发过程中提供一系列的服务，使开发过程更加轻松。.NET Framework 类库为程序开发人员提供了一个统一的、面向对象的、层次化的、可扩展的类库，包括类、接口和值类型。ASP.NET 提供 Web 应用程序模型。

1.1　初　识　C#

1.1.1　C#控制台应用程序

下面用一个控制台输出程序来介绍 C#控制台应用程序的编写流程和相关技术。该程序运行时在命令窗口中输出一行文字“Hello C#”，如图 1.1 所示。

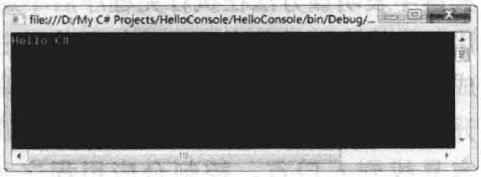

图 1.1　控制台应用程序

若要实现该程序，首先新建一个名为“HelloConsole”的“控制台应用程序”

项目，在“Program.cs”窗口中，修改代码如下：

```
class HelloConsole
{
    static void Main(string[] args)
    {
        System.Console.WriteLine("Hello C#");   //输出语句
        System.Console.ReadKey();  //等待接收键盘输入，窗口停驻显示
    }
}
```

按【Ctrl+F5】组合键运行该应用程序，验证该程序的显示内容，按任意键结束程序的运行。该程序涉及 C#中命名空间的概念、Main 方法、控制台应用程序的输入与输出，以及在控制台中格式化输出数据等相关内容。

1. 命名空间

Visual Studio 开发环境为程序开发人员提供了非常多的类，利用这些类可以快速完成各种各样复杂的功能。命名空间既是 Visual Studio 提供的系统资源的分层组织形式，也是分层组织程序的方式，其相当于在程序文件中建立了一个文件夹，如果几个程序属于同一个命名空间，则这些程序存储在这个文件夹中。命名空间有两种：一种是系统命名空间，另一种是用户自定义命名空间。系统命名空间用 using 关键字导入，用户自定义命名空间使用 namespace 关键字声明。

在 C#语言中，using 关键字的用途如下。

1）作为引用指令，用于为命名空间导入其他命名空间中定义的类型。

2）作为别名指令，用于简化命名空间的表达形式。

3）作为语句，用于定义一个范围。

2. Main 方法

Main 方法是 C#程序的入口点，其在类定义的内部声明，且只能声明为 public static int 或 public static void。其中，static 关键字是必需的，表明是静态方法；void 关键字表明该方法在执行完程序后不返回任何参数；int 类型的返回值用于表示应用程序终止时的状态码，其作用是退出应用程序时返回程序运行的状态（0 表示成功返回，非零值一般表示某个错误编号，错误编号所代表的含义也可以由程序开发人员自己规定）。Main 方法可以放在任何一个类中，但为了让程序开发人员容易找到入口点，控制台应用程序和 Windows 窗体应用程序默认将其放在 Program.cs 文件的 Program 类中。

3. 控制台应用程序的输入与输出

控制台是一个操作系统级别的命令行窗口，控制台应用程序实际上应该在命令行窗口下执行，但是在 Visual Studio 开发环境下，为了避免频繁地在开发环境和命令行窗口之间切换，也可以直接按【F5】键编译、调试并运行控制台应用程序。但是在运行程序时，程序开发人员会发现程序的执行结果会瞬间消失，无法看清楚具体执行的结果。为了在调试环境下直接观察输出的结果，一般在 Main 方法结束前加上“Console.ReadKey();”语句，其含义是读取键盘输入的任意一个字符，按【Space】键、【Enter】键或其他任何一个字符键，即可返回开发环境。

（1）控制台应用程序的输入

系统命名空间下的 Console 类提供了 ReadLine 方法，该方法可以从标准输入流依次读取从键盘输入的字符，并将其立即回显在控制台窗口，而且在用户按【Enter】键之前会一直等待输入。除了 ReadLine 方法之外，还可以使用 ReadKey 方法读取用户按下的某个字符键或功能键，并将键值回显在控制台窗口。

（2）控制台应用程序的输出

默认情况下，系统命名空间下的 Console 类提供 Write 方法和 WriteLine 方法，自动将各种类型的数据转换为字符串写入标准输出流，即输出到控制台窗口。Write 方法与 WriteLine 方法的区别是后者在输出数据后，自动输出一个回车换行符。

4. 在控制台中格式化输出数据

在控制台应用程序的 Console.Write 方法和 Console.WriteLine 方法的参数中，可以直接定义数据转换为字符串后的输出格式。其常用形式为

```
Console.WriteLine("格式化表示",参数序列);
```

或

```
Console.Write("格式化表示",参数序列);
```

（1）格式化表示的一般形式

使用格式化表示时，用“{”和“}”将格式与其他输出字符区分开。其一般形式为

```
{N [,M][: 格式码]}
```

其中，中括号中的内容为可选项。各字符的含义如下。

N：指定参数序列中的输出序号。

M：指定参数输出的最小长度。如果参数的长度小于 M，则用空格填充；如果参数的长度不小于 M，则按实际长度输出；如果 M 为负，则左对齐；如果 M

为正，则右对齐；如果未指定 M，则默认为零。

格式码：可选的代码字符串。

常用格式码及其用法示例如表 1.1 所示。

表 1.1　常用格式码及其用法示例

格式码	含义	示例	输出结果
C	将数字按照金额形式输出	Console.WriteLine("{0:C}",10);	¥10.00
		Console.WriteLine("{0:C}",10.5);	¥10.50
D 或 d	输出十进制整数。D 后的数字表示输出位数，不足指定的位数时，左边补 0	Console.WriteLine("{0:D}",10);	10
		Console.WriteLine("{0:D5}",10);	00010
F 或 f	输出小数点后固定位数（四舍五入），F 后面不指定位数时，默认为两位	Console.WriteLine("{0:F}",10);	10.00
		Console.WriteLine("{0:F4}",10.56736);	10.5674
		Console.WriteLine("{0:F2}",12345.6789);	12345.68
		Console.WriteLine("{0:F3}",123.45);	123.450
N 或 n	整数部分每三位用逗号分隔；输出小数点后固定位数（四舍五入），N 后面不指定位数时，默认为两位	Console.WriteLine("{0:n4}",12345.6789);	12,345.6789
P 或 p	以百分比形式输出，整数部分每三位用逗号分隔；小数点后输出固定位数（四舍五入），P 后面不指定位数时，默认为两位	Console.WriteLine("{0:p}",0.126);	12.60%
X 或 x	按十六进制格式输出。X 后的数字表示输出位数，不足指定的位数时，前面补 0	Console.WriteLine("{0:X}",10);	A
		Console.WriteLine("{0:X4}",10);	000A
0	0 占位符。如果数字位数不足指定的占位符位数，则左边补 0；如果数字位数超过指定的占位符位数，则按照实际位数原样输出。如果小数部分的位数超出指定的占位符位数，则多余的部分四舍五入	Console.WriteLine("{0:00000}",123);	00123
		Console.WriteLine("{0:000}",12345);	12345
		Console.WriteLine("{0:0000}",123.64);	0124
		Console.WriteLine("{0:00.00}",123.6484);	123.65
#	#占位符。对于整数部分，去掉数字左边的无效 0；对于小数部分，按照四舍五入原则处理后，再去掉右边的无效 0。如果这个数就是 0，而又不想让它显示的时候，可使用#占位符	Console.WriteLine("{0:####}",123);	123
		Console.WriteLine("{0:####}",123.64);	124
		Console.WriteLine("{0:####.###}",123.64);	123.64
		Console.WriteLine("{0:####.##}",0);	
		Console.WriteLine("{0:####.##}",123.648);	123.65

在格式化表示形式中，有两个特殊的用法：

1）如果要在格式中使用大括号，可以用连续的两个大括号表示一个大括号，如“{{、}}”。

2）如果既希望格式中的字符或字符串包含与格式符相同的字符，又希望让其原样显示，则可以用单引号将其引起来。

（2）利用 string.Format 方法格式化字符串

用户可以利用 string.Format 方法将某种类型的数据按照希望的格式转换为对应的字符串，该方法既可以在控制台应用程序中使用，也可以在其他应用程序中使用。

（3）利用 ToString 方法格式化字符串

如果只有一个变量，则使用 ToString 方法格式化字符串更简单。无论是控制台应用程序，还是 WinForm 应用程序、WPF 应用程序，或其他类型的应用程序，都可以利用 string.Format 方法或 ToString 方法定义数据的格式。

1.1.2　C#窗体应用程序

下面用一个 WinForm 输出程序来介绍 C#窗体应用程序的编写流程和相关技术。程序的运行结果如图 1.2 所示，程序完成的功能：单击“显示”按钮，窗体文本框中显示“Hello C#，这是第一个 Winform 程序。”；单击“清除”按钮，清除窗体文本框中的内容。

图 1.2　文本框中输出一行文字

为了实现该程序，首先新建一个名为“Hello Winform”的“Windows 窗体应用程序”项目，从“工具箱”窗格向窗体 Form1 中拖放一个文本框（TextBox）控件和两个命令按钮（Button）控件，调整控件的布局；然后按【F4】键打开“属性”窗格，设置 Button1 控件的 Name 属性为“btnShow”，Text 属性为“显示”。同理，设置 Button2 控件的 Name 属性为“btnClear”，Text 属性为“清除”；设置 TextBox 控件的 Name 属性为“txtBx”，Text 属性为空，Mutiline 属性为“True”，即允许多行显示，TextAlign 属性为“Center”，即居中对齐。

双击“显示”按钮，打开代码编辑器窗口，为“显示”按钮添加单击（Click）事件，代码如下：

```
private void btnShow_Click(object sender, EventArgs e)
{
    txtBx.Text = "Hello C#，这是第一个 Winform 程序。";//要插入的代码行
}
```

关闭代码编辑器窗口，或单击窗体设计器（Form1.cs[设计]*）的标题，切换到窗体设计器窗口，双击“清除”按钮，打开代码编辑器窗口为“清除”按钮添

加单击事件，代码如下。

```
private void btnClear_Click(object sender, EventArgs e)
{
    txtBx.Text = "";                    //要插入的代码
}
```

按【F5】键运行该应用程序，单击“显示”按钮，显示“Hello C#，这是第一个 Winform 程序。”；单击“清除”按钮，清除显示内容。关闭 Windows 窗体返回 Visual Studio 主界面。该程序涉及 Windows 窗体应用程序的相关操作方式。

1. Windows 窗体应用程序的启动和退出

WinForm 的程序入口也是 Main 方法，但由于这种应用程序基于消息循环和线程来管理应用程序，因此其用法与控制台应用程序不同。在 WinForm 编程模型的 Main 方法中，使用 Application 类提供的静态方法来启动、退出应用程序。Application 类提供的常用方法如下。

1）Run 方法：用于在当前线程上启动应用程序消息循环，并显示窗体。

2）Exit 方法：用于停止应用程序消息循环，即退出应用程序。

2. 窗体的相关操作

Windows 窗体应用程序提供以下窗体的操作方式。

（1）显示窗体

若要显示一个窗体，则必须先创建该窗体类的实例，再调用该实例提供的方法将其显示出来。

Show 方法将窗体显示出来后立即返回，此后程序会执行 Show 方法后面的语句，而不会等待该窗口关闭，这种类型的窗体称为无模式窗体。

如果使用 ShowDialog 方法来显示窗体，则该方法将窗体显示出来以后，在该窗体关闭之前，应用程序中的所有其他窗体都会被禁用，这种类型的窗体称为模式窗体。

（2）隐藏显示的窗体

对于无模式窗体，调用 Hide 方法即可将其隐藏起来。窗体隐藏后，其实例仍然存在，因此，可以重新调用 Show 方法再次将其显示出来。

（3）关闭窗体

在代码中直接调用 Close 方法即可关闭窗体。另外，无论程序打开了多少个窗体，也无论当前窗体是哪个，只要调用 Application 类提供的 Exit 方法，就会退出应用程序，其语法为

```
Application.Exit();
```

（4）注册事件

事件是响应用户操作的一种技术。在 Windows 窗体应用程序中，有三种注册事件的方法。

1）在窗体的设计界面中双击控件，此时会默认自动注册最常用的事件，如按钮的最常用事件是单击事件，所以双击按钮注册的是单击事件。

2）选择某个控件，单击“属性”窗格中的“事件”按钮，即可看到该控件对应的各种事件，双击指定事件，即可注册对应的事件。

3）在代码中通过“+=”注册指定的事件，通过“-=”注销指定的事件。当熟悉代码后，使用这种方法注册或注销事件是最灵活、方便的。

3. WPF 应用程序

虽然 WinForm 应用程序实现简单、开发方便，但是由于这种应用程序开发模型最初是为了开发在 Windows XP 操作系统上运行的程序而设计的，当需要高效率运行动画、三维图形和音频、视频等多媒体功能时，WinForm 应用程序只能依靠其他软件来实现，无法直接利用图形处理器（Graphic Processing Unit，GPU）的硬件加速功能，因此微软公司在 WinForm 的基础上推出了 WPF 技术。

4. Silverlight 应用程序

Windows 窗体应用程序及 WPF 应用程序主要用于开发基于客户机/服务器（Client/Server，C/S）的桌面客户端应用程序，而 Silverlight 应用程序是一种插件式的富客户端浏览器应用程序，其功能类似于 Flash。

Silverlight 应用程序主要靠客户端浏览器来承载运行，也可以独立运行。对于桌面计算机来说，Silverlight 应用程序不需要在客户端安装.NET Framework，因为它是依靠安装在客户端的 Silverlight 插件来运行的。

对于程序开发人员来说，从应用程序接口（Application Programming Interface，API）层面上看，Silverlight 应用程序和 WPF 应用程序非常相似，都是用可扩展应用程序标记语言（eXtensible Application Markup Language，XAML）来编写页面的，后台代码用 C#语言来编写；但是从内部实现来看，由于 Silverlight 应用程序需要解决跨浏览器、安装 Windows Phone 操作系统的手机及 Xbox 360 游戏机三种平台的问题，因此它实际上是与 WPF 技术不同的解决方案和实现技术。

1.2　变量、常量与表达式

变量是指在程序的运行过程中可以发生变化的量，是数据的临时存放场所。

C#中的变量必须先声明后使用。对于局部变量，C#中可以用 var 来声明其类型，编译时由系统自动判定其数据类型。常量是在程序运行过程中值不会发生变化的量。常量有直接常量和符号常量两种。直接常量，即数据值本身。C#中用 const 关键字声明符号常量。

表达式由运算符和操作数构成。C#中常用运算符及其说明如表 1.2 所示。

表 1.2 C#中常用运算符及其说明

运算符类型	说明
点运算符	指定类型或命名空间的成员
圆括号运算符()	指定表达式运算顺序；用于显式转换；用于方法或委托，即将参数放在括号内
方括号运算符[]	用于数组和索引，用于特性（Attribute），用于指针（非托管模式）
new 运算符	用于创建对象和调用函数，如 Class1 obj=new Class1()；用于创建匿名类型的实例
递增、递减运算符	++运算符将变量的值加 1，--运算符将变量的值减 1
赋值运算符	=、+=、-=、*=、/=、%=、<<=、>>=、&=、^=、|=
算术运算符	加（+）、减（-）、乘（*）、除（/）、求余数（%）
关系运算符	大于（>）、小于（<）、等于（==）、不等于（!=）、小于等于（<=）、大于等于（>=）
条件运算符	&& 条件“与”，|| 条件“或”
??运算符	如果类不为空值，则返回它自身；如果类为空值，则返回之后的操作
?:运算符	三元运算符
逻辑运算符（按位操作运算符）	逻辑与（&）、逻辑或（|）、逻辑非（!）、逻辑异或（^）、按位求反（~）、逻辑左移（<<）、逻辑右移（>>）
typeof 运算符	获取类型的 System.Type 对象
is 运算符	检查对象是否与给定类型兼容，x is T 含义为如果 x 为 T 类型，则返回 true，否则返回 false
as 运算符	x as T 含义为返回 T 类型的 x，如果 x 不是 T 类型，则返回 null

表达式是可以计算且结果为单个值、对象、方法或命名空间的代码片段。当一个表达式包含多个操作符时，表达式的值就由各操作符的优先级决定。

优先级是指当一个表达式中出现不同的运算符时，先进行何种运算。运算符的优先级如表 1.3 所示。结合性是指当一个表达式中出现两个以上同级运算符时运算符的先后规则。在多个同级运算符中，赋值运算符与条件运算符是自右向左结合的，除了赋值运算符以外的二元运算符都是自左向右结合的。

表 1.3 运算符的优先级

优先级	类别	运算符
1	一元运算符	+（取正）、-（取负）、!（非）、++x（前增量）、--x（前减量）
2	乘、除、求余运算符	*、/、%
3	加、减运算符	+、-
4	关系运算符	>、<、<=、>=
5	关系运算符	==、!=

续表

优先级	类别	运算符
6	条件“与”运算符	&&
7	条件“或”运算符	||
8	?:运算符	?:
9	赋值运算符	=、*=、/=、%=、+=、-=、<<=、>>=、&=、^=、|=

为了使表达式按正确的顺序进行运算，避免实际运算顺序不符合设计要求，也为了提高表达式的可读性，可以使用圆括号明确运算顺序。当使用圆括号时，括号内的数据先运算，括号外的数据后运算，不受优先级和结合性的影响。

1.3　C#的数据类型

C#的数据类型分为两种，一种是值类型，另一种是引用类型，如表 1.4 所示。

表 1.4　C#的数据类型

类别		说明
值类型	简单类型	有符号整型：sbyte、short、int、long 无符号整型：byte、ushort、uint、ulong Unicode 字符：char IEEE 浮点型：float、double 高精度小数型：decimal 布尔型：bool
	枚举	enum E{…}形式的用户自定义类型
	自定义结构类型	struct S{…}形式的用户自定义类型
	可空类型	具有 null 值的值类型的扩展
引用类型	类	所有其他类型的最终基类：Object Unicode 字符串：String class C{…}形式的用户自定义类型
	接口	interface I{…}形式的用户自定义类型
	数组	一维数组和多维数组，如 int[]、int[,]
	委托	delegate int D(…)形式的用户自定义类型

值类型与引用类型的区别在于，值类型变量直接在堆栈中保存变量的值，引用类型的变量在堆栈中保存的是对象的引用地址。值类型与引用类型的区别如表 1.5 所示。在 C#中，无论是值类型还是引用类型，其最终基类都是 Object 类。

表 1.5　值类型与引用类型的区别

特性	值类型	引用类型
变量中保存的内容	实际数据	指向实际数据的引用指针

续表

特性	值类型	引用类型
内存空间配置	堆栈（Stack）	受管制的堆（Managed Heap）
内存需求	较少	较多
执行效率	较快	较慢
内存释放时间点	超过变量的作用域时	由垃圾回收机制负责回收

1.3.1 基本数据类型

C#中的基本数据类型是系统预定义的数据类型，也称为内置数据类型。根据数据的性质，内置数据类型可以分为数值类型、字符类型、布尔（逻辑）类型和对象类型四类。

1. 数值类型

C#中的数值类型有整数类型与实数（浮点数）类型两种，前者不带小数，后者带小数，它们都属于内置数值类型。整数类型如表 1.6 所示。

表 1.6 整数类型

数据类型	数据范围	有无符号	占用空间/字节	.NET 等价类	类型指定符
byte	0～255 间的整数	无	1	System.Byte	
sbyte	-128～+127 间的整数	有	1	System.SByte	
short	-32768～+32767 间的整数	有	2	System.Int16	
ushort	0～65535 间的整数	无	2	System.UInt16	
int	-2147483648～+2147483647 间的整数	有	4	System.Int32	如果是十六进制数需要加 0x 前缀
uint	0～4294967295 间的整数	无	4	System.UInt32	扩展名：U 或 u
long	-9223372036854775808～+9223372036854775807 间的整数	有	8	System.Int64	扩展名：L 或 l
ulong	0～18446744073709551615 间的整数	无	8	System.UInt64	扩展名：UL 或 ul

实数（浮点数）类型包括 float（单精度浮点型）、double（双精度浮点型）、decimal（十进制型）等，如表 1.7 所示。

表 1.7 实数类型

数据类型	数据范围	有无符号	占用空间/字节	.NET 等价类	类型指定符
float	$\pm 1.5\times10^{-45}$～$\pm 3.4\times10^{38}$	有	4	System.Single	F 或 f