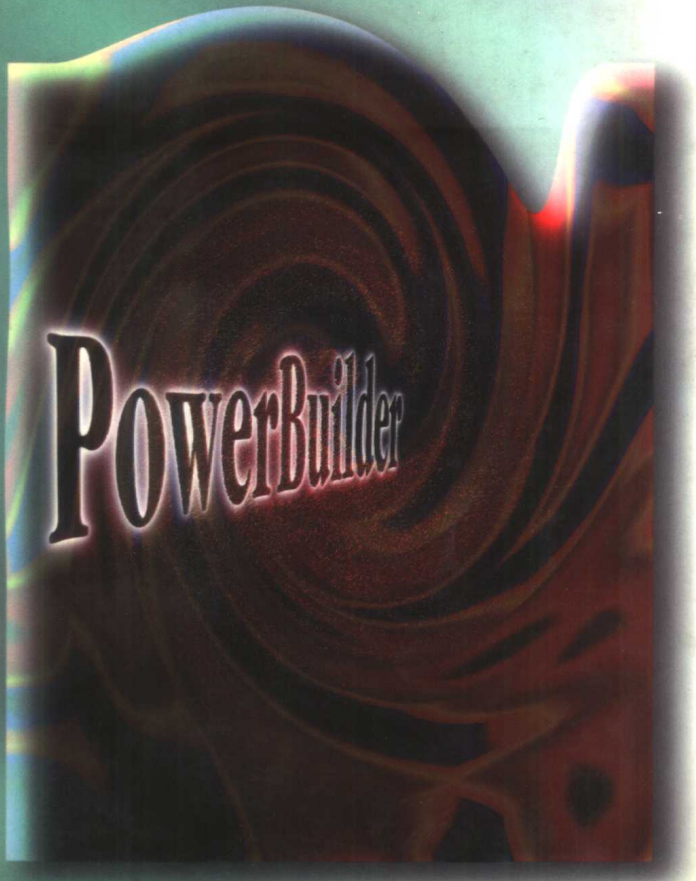
计算机语言函数应用丛书

PowerBuilder

计算机语言函数应用

袁晓君　李　明　李团结　编著



科学出版社

计算机语言函数应用丛书

PowerBuilder

计算机语言函数应用

袁晓君　李　明　李团结　编著

科学出版社

2000

内 容 简 介

本书详细介绍了 PowerBuilder 6.0 版本中所涉及的所有函数的内容，包括系统函数、对象函数、邮件函数、OLE 函数、分布式应用函数、支持动态 SQL 函数、Trace 函数、数据管道函数、Internet/Intranet 服务函数、数据窗口中使用的统计函数等。为读者更好地掌握该语言函数，本书还提供了大量范例。

本书面向已对 PowerBuilder 初步了解和使用的用户。

图书在版编目（CIP）数据

PowerBuilder 计算机语言函数应用/袁晓君等编．－北京：科学出版社，2000

ISBN 7-03-008143-9

Ⅰ．计… Ⅱ．袁… Ⅲ．数据库管理系统-软件工具，PowerBuilder-库函数 Ⅳ．TP311.56

中国版本图书馆 CIP 数据核字（1999）第 72833 号

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

北京双青印刷厂 印刷

新华书店北京发行所发行 各地新华书店经售

*

2000 年 1 月第 一 版 开本：787×1092 1/16

2000 年 1 月第一次印刷 印张：32 3/4

印数：1—5 100 字数：763 000

定价：45.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

前言

PowerBuilder 6.0 由 PowerSoft 公司最新推出，在 PowerBuilder 5.0 基础上增加和增强了多种特性，特别增强了 Internet 的应用。PowerSoft 公司将过去单独出售的所有 Internet 开发工具箱集成到 PowerBuilder 6.0 的开发环境中，并且装有 Java 代理生成器的 Java 客户机可以直接访问 PowerBuilder 6.0 对象和应用逻辑。

PowerBuilder 之所以能够成为当今最具有代表性的数据库前端开发工具之一，主要有以下几个因素：

(1)面向对象的应用程序开发方法。

(2)基于 Client/Server 的体系结构。

PowerBuilder 为诸如 Oracle，Informix，Sybase 等数据库提供了公用的前端开发工具，既可以用于各类数据库管理系统，又可以给已经存在的数据库资源创建新的窗口界面。

(3)可视化的开发环境。

PwerBuilder 的使用者无需掌握 C 或 Windows 编程，就可以利用图形化的交互操作完成应用系统的设计和实现。

本书详细介绍了 PowerBuider 涉及的各类函数，全书分成十一章描述：

第一章　PowerBuilder 中的系统函数；第二章　PowerBuilder 中的对象函数；第三章　PowerBuilder 中的邮件函数；第四章　PowerBuilder 中的 OLE 函数；第五章　PowerBuilder 中的分布式应用函数；第六章　PowerBuilder 中的支持动态 SQL 的函数；第七章　PowerBuilder 中的 Trace 函数；第八章　PowerBuilder 中的数据管道函数；第九章　PowerBuilder 中的 Internet/Intranet 服务函数；第十章　PowerBuilder 中的数据窗口中使用的统计函数；第十一章　介绍了 PowerBuilder 中的其他函数。

目 录

1 系统函数

系统函数是 PowerScript 的通用函数，也是 PowerBuilder 提供的内部函数，不属于任何对象，在应用程序的任何地方均可直接使用。系统函数包括以下几类，他们是：文件操作函数、数组函数、Blob(大二进制对象)函数、日期、时间函数、DDE 客户函数、DDE 服务端函数、数据类型检查与转换函数、国际化函数、库管理函数、数值计算函数、打印函数、注册(Registry)函数、定时函数、窗口操作函数、字符串操作函数、系统与环境函数及其他函数。

下面将详细介绍这些函数。

1.1 文件操作函数

文件操作函数完成对文件的处理功能(如，文件的输入、修改、输出、删除等)。

首先打开文件，然后才能访问文件中的数据，访问完成后需要关闭文件。

打开文件的方式有两种：行模式(Line Mode)和流模式(Stream Mode)。

具体来讲，在行模式下读取数据时，一次读一行，直到读取到换行(LF)、回车(CR)符或文件结束符时停止，写文件时，一次写一行，在每行的末尾自动添加回车换行符。

在流模式下，每次从文件读写 32K 字节的数据，文件中的换行符、回车符和其他字符读写相同。当文件小于 32K 或剩余数据不到 32K 时，按字节依次读取数据，直到读到文件结束符(EOF)时停止；写数据时，写数据原样到文件中，最后加上 EOF 字符。

操作函数列表如下：

函 数	功 能
FileClose( )	将前面用 FileOpen( )函数打开的文件关闭。
FileDelete( )	按给定参数删除给定文件。
FileExists( )	按给定参数检查指定的文件是否存在。
FileLength( )	获取文件的长度(以字节为单位)。
FileOpen( )	按参数给出的读写方式打开文件。
FileRead( )	读指定文件
FileSeek( )	移动文件指针到指定位置。
FileWrite( )	写数据到指定文件。
GetFileOpenName( )	打开文件对话框，由用户选择想要打开的文件。
GetFileSaveName( )	通过保存文件对话框选择文件名

FileClose( )

语法:FileCLose(fileno)

功能:将前面用 FileOpen( )函数打开的文件关闭。

参数:fileno　整型变量,将要关闭文件的文件句柄。

该句柄可以用 FileOpen( )函数打开。

用法:该函数关闭由 fileno 参数给定的文件。

在关闭该文件的同时释放它占有的系统空间。

注意:如果在使用完文件后立即关闭文件,可以确保顺利写数据到文件,而且内存的利用率被显著提高。

返回值:整型变量。

函数执行成功返回 1,否则返回 -1。

如果 fileno 参数的值为 NULL,那么该函数返回 NULL。

参照:FileLength( ),FileOpen( ),FileRead( ),FileWrite( )

范例:下例打开文件,同时读出数据,之后再关闭文件。

```
String s
integer a
a = FileOpen("c: \ windows \ staff.dat",LineMode!)
FileRead(a,s)
FileClose(a)
```

FileDelete( )

语法:FileDelete(filename)

功能:按给定参数删除给定文件。

参数:filename　string 类型。

指出要删除文件的文件名,可包含路径。

注意:文件名一定是 DOS 文件名。

如果在 Windows95 下的文件名和 DOS 下的文件名不同,就不能保证删除成功。

用法:该函数从磁盘上删除参数指定的文件。

如果指定的文件不在当前应用程序库的搜索路径上,则在该参数中必须给定文件的驱动器号、文件名、扩展名和文件所在的目录。

注意:如果其他应用程序正使用该文件,则该函数返回 FALSE。

返回值:boolearn。

函数成功执行时返回 True,否则返回 False。

如果 filename 参数的值为 NULL,则该函数返回 NULL。

参照:FileExists( )

范例:下例删除用户在打开的文件窗口选择的文件。

```
integer a, value
string d, n
```

```
value = GetFileOpenName("选择文件," & d, n, "DOC", &
    "Doc Files (*.DOC), *.DOC")
IF value = 1
    THEN a = MessageBox("删除", & "删除文件?", Question!, OKCancel!)
IF a = 1
    THEN FileDelete(d)
```

FileExists()

语法:FileExists(filename)

功能:按给定参数检查指定的文件是否存在。

参数:filename　string 类型。

给定要检查文件的名字,可包含路径。

注意:文件名一定是 DOS 文件名。

如果 Windows95 下的文件名和 DOS 下的文件名不同,就不能保证正确查找。

用法:同"功能"所述。

如果文件不在当前应用程序库的搜索路径上,在参数中一定要指出文件的驱动器号、文件名、扩展名和文件所在的目录。

为了确保文件删除或打开顺利执行,一般在调用 FileDelete()或 FileOpen()前使用该函数。

此外,如果指定的文件被其他应用程序锁定,则该函数返回 False。

返回值:boolean 类型。

如果指定文件存在返回 True,否则返回 False。

如果 filename 参数的值为 NULL,则该函数返回 NULL。

参照:FlieDelete(),FileOpen()

范例:下例判断用户在 Save File window 选择的文件是否存在,如果存在,询问用户是否覆盖它。

```
string l_dname, ls_name
integer a
boolean exist
GetFileSaveName("选择文件," l_dname, & ls_name, "pbl", & "Doc Files (*.DOC),
    *.DOC")exist = FileExists(l_dname)
IF lb_exist THEN a = MessageBox("保存", & "覆盖" + l_dname, & Question!,
    YesNo!)
```

FileLength()

语法:FileLength(filename)

功能:获取文件的长度(以字节为单位)。

参数:filename　string 类型。

给出将得到其长度的文件和文件名,可包含路径。

用法:该函数获得参数给定的文件的长度。

如果给定的文件不在当前应用程序库的搜索路径上,则在参数中一定要指出文件的驱动器号、文件名、扩展名和文件所在的目录。

该函数在文件打开前后均可使用。

注意:如果在文件共享的环境中,设置安全限制对给定文件,则为了避免文件共享冲突,该函数的使用必须是:文件打开前或文件关闭后。

返回值:long 类型。函数成功执行则返回指定文件的长度(以字节为单位)。如果指定的文件不存在,函数返回 -1。如果 filename 参数的值为 NULL,那么 FileLength( )函数返回 NULL。

参照:FileClose( ),FlieOpen( ),FileRead( ),FileWrite( )

范例:下例返回文件当前目录下 staff.dat 的长度。

```
FileLength("staff.dat ")
```

下例确定 Windows 目录下 file.txt 的长度,同时打开该文件。

```
long Length
integer a
Length = FileLength("C: \ Windows \ file.txt ")
a = FileOpen("C: \ Windows \ file.txt ", & StreamMode!, Read!, LockReadWrite!)
```

FileOpen( )

语法:FileOpen(filename{,filemode},fileaccess{,filelock},writemode,{creator, filetype{{{{}}}})

功能:按参数给出的读写方式打开文件。

参数:creator 文件创建者的名称。

仅用于 macintosh 机,使用四个字符的字符串指定文件创建者。

注意:必须是四个字符,否则不能正常运行。

区分大小写。同样字母但大小写不同为不同的创建者。

如果想要打开的文件不存在,而该函数建立新文件,creator 为文件的创建者(如果没有给出 creator 参数,则设置新文件的创建者为 text);否则忽略该参数。

该参数必须与 filetype 参数同时设定。

fileaccess 枚举类型。

(可选项)文件访问方式。取值为:

Read! (缺省值)只读方式,打开的文件只能进行读操作;

Write! 只写方式,打开的文件只能进行写操作

filemode 枚举类型。

(可选项)文件打开方式。取值为:

LineMode! (缺省值)行模式;

StreamMode! 流模式;

filename String 类型。

将打开文件的名称,可包含路径。

filelock 枚举类型。

(可选项)文件加锁方式。取值为:

LockReadWrite! (缺省值)锁读写方式。仅打开该文件的用户可以访问该文件,其他用户对该文件的访问都被拒绝;

LockRead! 锁读方式。仅打开该文件的用户可以写该文件,但其他任何用户都可以读该文件;

LockWrite! 锁写方式。仅打开该文件的用户可以读该文件,但其他任何用户都可以读该文件;

Shared! 共享方式。所有用户可以读写该文件。

filetype 文件类型。

四个字符的字符串,表示标识文件的类型。

注意:必须是四个字符,否则不能正常运行。

区分大小写。同样字母但大小写不同为不同的创建者。

仅用于 macintosh 机。

该参数必须与 creator 参数同时设定。

如果想要打开的文件不存在,而该函数建立新文件,filetype 为文件的类型(如果没有给出 filetype 参数,则设置新文件的创建者为 TEXT);否则忽略该参数。

writemode 枚举类型。(可选项)文件写入方式。取值为:

Append! (缺省值)将数据添加到原文件尾部;

Replace! 覆盖原有数据。

用法:当文件以行模式打开时,每执行一次 fileread()函数读取一行数据;每执行一次 filewrite()函数,该函数自动在写出的字符串末尾增加一个回车(cr)换行(lf)符(如果数据没有这么多,那么 fileread()函数就读取所有余下的数据;执行一次 filewrite()函数时,最多可写入 32,765 个字节的数据,并且不添加回车换行字符。当文件以写方式使用 fileopen()函数时,如果指定的文件不存在,那么 fileopen()函数创建文件。

返回值:整型变量。

函数成功执行则返回打开文件的句柄,随后的文件操作函数利用该句柄完成对文件的操作。否则函数返回 -1。

如果任一参数的值为 NULL,那么该函数返回 NULL。

参照:FileClose(),FileLength(),FileRead(),FileWrite()

范例:例 1. 使用缺省参数打开文件 staff.dat 用于读。

缺省设置为 LineMode!, Read! 和 LockReadWrite!。FileRead 将一行一行读数据,直到关闭不会有其他用户可以访问该文件。

```
integer a
a = FileOpen("staff.dat")
```

例 2. 在流模式(StreamMode!),以只写模式(Write!)打开 department 目录下的 staff.dat 文件。现有数据被覆盖 (Replace!)。没有其他用户可以写入该文件(LockWrite!)。

```
integer a
a = FileOpen("C:\department\staff.dat", &
StreamMode!, Write!, LockWrite!, Replace!)
```

FileRead( )

语法:FileRead(fileno,variable)

功能:读指定文件。

参数:fileno　整型变量。

给定文件句柄。

variable　string 或 blob 类型的变量。

保存读取的数据。

用法:当指定文件以行模式(line mode)打开时,fileread( )函数一次读取一行数据,并把它保存到参数 variable 中,然后跳过行结束符(回车换行符,操作系统不同,使用的字符也不同),把文件指针移动到下一行的起始位置。

当文件以流模式(stream mode)打开时,fileread( )函数或一直读取到文件结尾,或读取 32,765 字节的数据,决定于两者哪个数据长度更短些。

返回值:整型变量。

函数成功执行则返回读取的字符数或字节数;如果在读取任何字符前读到了文件结束符(eof),则 fileread( )函数返回 -100;当指定文件以行模式打开时,如果在读取任何字符之前遇到了回车(cr)或换行(lf)字符,则 fileread( )函数返回 0。如果发生其他错误,fileread( )函数返回 -1。如果任何参数的值为 NULL,那么 fileread( )函数返回 NULL。

参照:FileClose( ),FileLength( ),FileOpen( ),FileSeek( ),FileWrite( )

范例:下例读取文件 data.txt。

```
integer a
string s
long ll _ FLength
ll _ FLength = FileLength("C: \ rc \ data.txt ")
a = FileOpen("C: \ rc \ data.txt ", & StreamMode!)
IF ll _ FLength < 32767 THEN
FileRead(a, s)
END IF
```

FileSeek( )

语法:FileSeek(fileno,offset,origin)

功能:移动文件指针到指定位置。

参数:fileno　整型变量。

文件句柄(由 FileOpen( )函数得到)

origin　枚举型。

指定开始移动文件指针的位置。取值为:

FromBeginning!　(缺省值)从文件头移动指针。

FromCurrent!　从当前位置移动文件指针。

FromEnd! 从文件尾移动文件指针。

offset Long 型。

相对于 origin 参数的新位置偏移量(字节数)。

该值可以取正值(代表文件指针向文件尾方向移动),或取负值(代表文件指针向文件头方向移动),或取 0(如果 origin 是 FromEnd!,则移动文件指针到文件尾,同时函数返回文件长度)。

用法:该函数将文件指针从给定的位置移动到指定数目的字节,紧接着将移动后文件指针从距离文件头的字节数返回。

该函数不会对下面的异常操作提出警告:

origin 与 offset 之和是负值时,该函数返回 -1。

如果文件句柄对应的文件不存在,该函数返回 offset 与 18 的和。

origin 与 offset 之和超过文件长度时不出现出错提示,而是返回两者之和。

返回值:long。

函数成功执行则返回指针移动的指针位置。

如果任何参数的值为 NULL,则该函数返回 NULL。

参照:FileRead(),FileWrite()

范例:例 1. 将文件指针设置到离文件尾 14 字节。

```
integer a
a = FileOpen("c: \ Windows \ data")
FileSeek(a, -14, FromEnd!)
```

例 2. 将文件指针从当前位置朝文件尾移动 14 字节。在这种情况下,如果在 FileOpen 后没有其他调用对文件指针产生影响,则指定 FromCurrent! 和 FromBeginning! 结果相同。

```
integer a
a = FileOpen("data")
FileSeek(li _ FileNum, 14, FromCurrent!)
```

FileWrite()

语法:FileWrite(fileno,variable)

功能:写数据到指定文件。

参数:fileno 整型变量。

文件句柄(由 FileOpen()函数得到)

variable string 或 blob 类型。

写入 fileno 参数指定文件的数据。

用法:该函数从当前文件指针开始写指定数据,随后将文件指针移动到刚刚写入数据的下一个字节位置。

写入方式在打开文件时设置。

如果文件以 Replace! 方式打开,起始文件指针位于文件头;如果文件以 Append! 方式打开,起始文件指针位于文件尾。

操作模式是 FileOpen( ) 函数打开该文件时设置。

如果文件以行模式打开,调用该函数时,该函数在每次写入数据后面自动加上回车换行符,同时将文件指针移动到回车换行符后。如果文件以流模式打开,该函数最多一次写入 32,765 个字节。

如果 variable 中数据长度超过 32,765 个字节,则该函数仅向文件写入前 32,765 个字节且返回 32,765

返回值:整型变量。

函数成功执行则返回写入文件的字符或字节数,否则返回 -1。

如果任何参数的值为 NULL,则该函数返回 NULL。

参照:FileClose( ),FileLength( ),FileOpen( ),FileRead( ),FileSeek( )

范例:下例打开 data.txt 文件且在文件尾写入字符串 New Staffs。

变量 a 保存被打开的文件号。

```
integer a
a = FileOpen("C:\rc\data.txt", & LineMode!, Write!, LockWrite!, Append!)
FileWrite(a, "New Staffs")
```

GetFileOpenName( )

语法:GetFileOpenName(title,pathname,filename{,extension},filter{})

功能:打开文件对话框,由用户选择想要打开的文件。

参数:extension string 类型。

(可选项)用 1 - 3 个字符指定缺省的扩展文件名。

如果用户没有键入文件名后缀,则该参数自动在文件名后加上后缀。

filename string 类型。

保存该对话框返回的文件名。

filter string 类型。

(可选项)其值为文件名掩码,给出文件类型过滤器。

该参数可以选择屏蔽显示列表框中的文件,缩小选择范围。

filter 参数的格式为:

"description, *.txt"

缺省值为"All Files(*.*), *.*"

其中,description 说明扩展名的意义,可根据需要指定在打开文件对话框中显示的文件名类型(如果需给出许多文件类型时,用逗号分隔各类型)。

pathname string 类型。

保存该对话框返回的文件路径及文件名。

title string 类型。

给出对话框的标题。

用法:该函数仅得到一个文件名,并没有打开文件。

打开文件仍需要调用 FileOpen( )函数。

返回值:整型变量。

函数成功执行返回 1,否则返回 -1。

如果用户单击对话框上的"cancel"按钮则函数返回 0。

如果任何参数值均为 NULL,则该函数返回 NULL。

参照:GetFileSaveName()

范例:下例显示打开的文件窗口。如果 GetFileOpenName 成功执行,则打开用户选择的文件。文件类型是 TXT 和 DOC。

```
string dname, name
integer value
value = GetFileOpenName("选择文件", &+ dname, name, "DOC", &
+ "Text Files (*.TXT), *.TXT," &+ "Doc Files (*.DOC), *.DOC")
IF value = 1 THEN FileOpen(dname)
```

GetFileSaveName()

语法:GetFileSaveName(title,pathname,filename{,extension{,filter}})

功能:通过保存文件对话框选择文件名。

参数:extension string 类型。

(可选项)用 1-3 个字符指定缺省的扩展文件名。

如果用户没有键入文件名后缀,则该参数自动在文件名后加上后缀。

filename string 类型。

保存该对话框返回的文件名。

filter string 类型。

(可选项)其值为文件名掩码,给出文件类型过滤器。

该参数可以选择屏蔽显示列表框中的文件,缩小选择范围。

filter 参数的格式为:

"description, *.txt"

缺省值为"All Files(*.*), *.*"

其中,description 说明扩展名的意义,可根据需要指定在打开文件对话框中显示的文件名类型(如果需给出许多文件类型时,用逗号分隔各类型)。

pathname string 类型。

保存该对话框返回的文件路径及文件名。

title string 类型。

给出对话框的标题。

用法:该函数仅得到一个文件名,并没有打开文件。

打开文件仍需要调用 FileOpen()函数。

返回值:整型变量。

函数成功执行则返回 1;否则返回 -1。

如果用户单击对话框上的"cancel"按钮,则返回 0。

如果任何参数的值均为 NULL,则该函数返回 NULL。

参照:GetFileOpenName()

范例：下例通过存储文件对话框选择文件名，同时向该文件写数据。

```
string dname, name
string data = "hello"
integer value, resp
resp = GetFileSaveName("存储文件", name, dname, "txt", "text &
file( * .txt), * .txt, exe file, * .exe")
IF resp = 1 THEN
    MessageBox ("存储","将存储的文件:" + dname + & "完整路径名:" + name,
                Information!)
    ELSEIF resp = 0 THEN
        MessageBox ("取消","按下对话框的取消按钮!", stopsign!)
        RETURN
    ELSE
        MessageBox("错误","操作错误", stopsign!)
        RETURN
ENDIF
value = FileOpen( name, LineMode!, Write!, LockWrite!)
resp = FileWrite( value, data)
FileClose( value)
```

1.2 数组函数

利用数组函数可以得到数组的上界。

列表如下：

函　数	功　能
LowerBound( )	按给定参数获得数组第 n 维的下界。
UpperBound( )	按给顶参数获得数组第 n 维的上界。

下面分别详细介绍。

LowerBound( )

语法：LowerBound(Array{, n})

功能：按给定参数获得数组第 n 维的下界。

参数：array　数组类型，数组名。

n　数值类型。

(可选项)给出将获取哪一维的下界(缺省值 1)。

用法：该函数给出数组的某维的下界。

如果是可变数组，该函数返回已经分配空间的最低下界的值。如果没有分配空间，

则返回 1。

如果是固定数组,该函数获得声明时描述的某维下界值。

返回值:Long。

函数成功执行则返回 array 数组第 n 维的下界。

如果 n 的值超出数组的最大维数,则该函数返回 -1。

如果任何参数的值均为 NULL,则该函数返回 NULL。

参照:UpperBound()

范例:下例表明该函数报告内存分配前后固定数组和可变数组下界的情况。

```
integer a[5], b[2,5]
LowerBound(a)
  (返回 1)
LowerBound(a, 1)
  (返回 1)
LowerBound(a, 2)
  (返回 -1, a 只有 1 维)
LowerBound(b, 2)
  (返回 1)
integer c[ ]
LowerBound(c)
  (返回 1)
c[50] = 900
LowerBound(c)
  (返回 1)
integer d[-10, 50]
LowerBound(d)
  (返回 10)
```

UpperBound()

语法:UpperBound(Array{,n})

功能:按给定参数获得数组第 n 维的上界。

参数:array 数值类型,数组名

n 数值类型。

(可选项)给出要获得数组哪一维的上界。(缺省值 1)

用法:该函数给出数组的某维的上界。

如果是可变数组,该函数返回已经分配空间的最高上界的值。如果没有分配空间,返回 0。

如果是固定数组,该函数获得声明时描述的某维上界值。

返回值:Long。

函数成功执行返回 array 数组第 n 维的上界。

如果 n 的值超出数组最大维数，该函数返回 -1。

如果任何参数的值均为 NULL，该函数返回 NULL。

参照：LowerBound()

范例：下例表明该函数报告内存分配前后固定数组和可变数组上界的情况。

```
integer a[5]
UpperBound(a)
  (返回 5)
UpperBound(a,1)
  (返回 5)
UpperBound(a,2)
  (返回 -1,没有 2 维)
integer b[10,20]
UpperBound(b,1)
  (返回 10)
UpperBound(b,2)
  (返回 20)
integer c[ ]
UpperBound(c)
  (返回 0,没有分配内存)
c[50] = 900
UpperBound(c)
  (返回 50)
c[60] = 800
UpperBound(c)
  (返回 60)
c[60] = 800
c[50] = 700
UpperBound(c)
  (返回 60)
integer d[10 to 50]
UpperBound(d)
  (返回 50)
```

1.3 Blob 函数

Blob(大二进制对象)函数用于获得 Blob 数据类型变量的信息。此外它们可以实现数据类型转换和对 Blob 类型数据的操作。

列表如下：