

Microsoft Microsoft Microsoft

Microsoft®

Visual C++™ 2.0

for Win32® 大全 (四)

——运行库及io流类库参考手册

[美] Microsoft Corporation 著

丁国良 王嘉桢 张素琴 陈瑞 译



清华大学出版社



**Microsoft®  
Visual C++™ 2.0  
for Win32® 大全(四)  
——运行库及 io 流类库参考手册**

[美] Microsoft Corporation 著

丁国良 王嘉禎 译  
张素琴 陈 瑞

清华大学出版社

**(京)新登字 158 号**

Microsoft® Visual C++™ 2.0 for Win32® 大全(四)——运行库及 io 流类库参考手册  
Microsoft Visual C++ Run-Time Library Reference

Copyright © 1994 by Microsoft Corporation.

Original English Language Edition Copyright © 1994 by Microsoft Corporation.

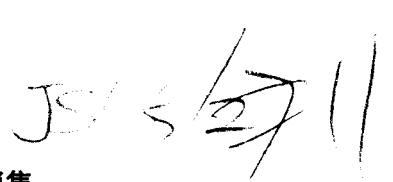
Published by arrangement with the original publisher, Microsoft Press, a division of Microsoft Corporation, Redmond, Washington, U. S. A.

本书中文版由 Microsoft Press 授权清华大学出版社出版。

中华人民共和国国家版权局著作权合同登记章图字：01-95-312 号

版权所有，翻印必究。

本书贴有 Microsoft Press 激光防伪标签，无标签者不得销售。



### 图书在版编目(CIP)数据

Microsoft® Visual C++™ 2.0 for Win32® 大全(四)：运行库及 IO 流类库参考手册/美国微软公司著；丁国良等译。—北京：清华大学出版社，1996  
ISBN 7-302-02106-6

I. M… II. ①美… ②丁… III. ①C语言,C++-程序设计-手册 ②程序库-手册  
IV. TP312C

中国版本图书馆 CIP 数据核字(96)第 02830 号

出版者：清华大学出版社（北京清华大学校内，邮编 100084）

印刷者：清华大学印刷厂

发行者：新华书店总店北京科技发行所

开本：787×1092 1/16 印张：26.5 字数：656 千字

版次：1996 年 6 月第 1 版 1996 年 6 月第 1 次印刷

书号：ISBN 7-302-02106-6/TP·986

印数：0001—3000

定价：52.00 元

0500100

# 目 录

## 第一部分 Microsoft 运行库函数

引言	3
0.1 兼容性	3
0.2 函数和宏之间的选择	4
0.3 类型检查	5
0.4 文档约定	5
<b>第 1 章 运行库例程分类</b>	<b>7</b>
1.1 参数访问例程	7
1.2 缓冲区操作例程	7
1.3 字节分类例程	8
1.4 字符分类例程	9
1.5 数据转换例程	10
1.6 目录控制例程	11
1.7 错误处理例程	11
1.8 异常处理例程	11
1.9 文件处理例程	12
1.10 浮点支持函数	12
1.10.1 长双精度数据类型	14
1.11 输入和输出例程	15
1.11.1 文本和二进制方式的文件读写	15
1.11.2 流 I/O 例程	16
1.11.3 低级 I/O 例程	17
1.11.4 控制台和 I/O 端口例程	18
1.12 国际化例程	19
1.12.1 地区相关例程	19
1.12.2 代码页	20
1.12.3 单字节字符集(SBCS)和多字节字符集(MBCS)	21
1.12.4 Unicode: 宽字符集	22
1.12.5 使用类属文本映射	22
1.13 内存分配例程	26
1.14 进程和环境控制例程	27
1.15 查找和排序例程	29
1.16 串操作例程	29

1.17 系统调用例程 .....	30
1.18 时间管理例程 .....	31
<b>第 2 章 全程变量和标准类型 .....</b>	<b>32</b>
2.1 全程变量 .....	32
2.1.1 - amblksiz .....	32
2.1.2 - daylight, - timezone, - tzname .....	32
2.1.3 - doserrno, errno, sys_errlist, sys_nerr .....	33
2.1.4 - environ, - wenviron .....	34
2.1.5 - fileinfo .....	35
2.1.6 - fmode .....	35
2.1.7 - osver, - winmajor, - winminor, - winver .....	35
2.1.8 - pgmptr, - wpgmptr .....	36
2.2 标准类型 .....	36
<b>第 3 章 按字母顺序排列的运行库函数 .....</b>	<b>39</b>
附录 A 语言和国家串 .....	321
附录 B 类属文本映射 .....	324

## 第二部分 io 流类库

引言 .....	333
0.1 关于本书 .....	333
0.2 文档约定 .....	333
<b>第 1 章 I/O 程序设计 .....</b>	<b>335</b>
1.1 流 .....	335
1.1.1 输入/输出选择 .....	335
1.1.2 io 流类层次 .....	335
1.2 输出流 .....	336
1.2.1 输出流对象的构造 .....	336
1.2.2 插入操作符的使用和格式控制 .....	337
1.2.3 文件输出流成员函数 .....	340
1.2.4 缓冲区的作用 .....	342
1.2.5 二进制输出文件 .....	343
1.2.6 在自定义类中重载 << 操作符 .....	344
1.2.7 无参数自定义操作符 .....	345
1.3 输入流 .....	345
1.3.1 构造输入流对象 .....	346
1.3.2 使用抽取操作符 .....	346
1.3.3 抽取错误测试 .....	346
1.3.4 输入流操纵符 .....	347
1.3.5 输入流成员函数 .....	347

1.3.6	在自定义类中重载 >> 操作符	349
1.4	输入/输出流	350
1.5	带参数的定制操纵符	350
1.5.1	单参数(int 或 long)的输出流操纵符	350
1.5.2	其它的单参数输出流操纵符	351
1.5.3	多参数输出流操纵符	352
1.5.4	输入和输入/输出流的定制操纵符	353
1.5.5	借助派生流类使用操纵符	353
1.6	导出自定义流类	353
1.6.1	streambuf 类	353
1.6.2	为何派生一个定制 streambuf 类	353
1.6.3	streambuf 派生的例子	354
<b>第 2 章</b>	<b>Microsoft io 流类库参考(按字母排序)</b>	<b>360</b>
2.1	io 流类层次图	360
2.2	io 流类一览表	360
2.3	类 filebuf	361
2.3.1	构造与析构——公有成员	361
2.3.2	操作——公有成员	361
2.3.3	状态/信息——公有成员	361
2.3.4	成员函数分述	362
2.4	类 fstream	363
2.4.1	构造与析构——公有成员	364
2.4.2	操作——公有成员	364
2.4.3	状态/信息——公有成员	364
2.4.4	成员函数分述	364
2.5	类 ifstream	367
2.5.1	构造与析构——公有成员	367
2.5.2	操作——公有成员	367
2.5.3	状态/信息——公有成员	367
2.5.4	成员函数分述	368
2.6	类 ios	370
2.6.1	数据成员(静态)——公有成员	370
2.6.2	构造与析构——公有成员	371
2.6.3	标志及格式访问函数——公有成员	371
2.6.4	状态测试函数——公有成员	371
2.6.5	自定义格式标志——公有成员	371
2.6.6	其它函数——公有成员	371
2.6.7	操作符——公有成员	371
2.6.8	ios 操纵符	372

---

2.6.9	参数化操作符	372
2.6.10	成员函数分述	372
2.6.11	操作符分述	377
2.6.12	操纵符分述	378
2.7	类 <code>iostream</code>	379
2.7.1	公有成员	380
2.7.2	保护成员	380
2.7.3	成员函数分述	380
2.8	类 <code>lostream_init</code>	380
2.8.1	公有成员	380
2.8.2	成员函数分述	381
2.9	类 <code>istream</code>	381
2.9.1	构造与析构——公有成员	381
2.9.2	前缀和后缀函数——公有成员	381
2.9.3	输入函数——公有成员	381
2.9.4	其它函数——公有成员	382
2.9.5	操作符——公有成员	382
2.9.6	保护成员	382
2.9.7	操纵符	382
2.9.8	成员函数分述	382
2.9.9	操作符分述	385
2.9.10	操纵符分述	386
2.10	类 <code>istream_withassign</code>	386
2.10.1	构造与析构——公有成员	386
2.10.2	操作符——公有成员	387
2.10.3	成员函数分述	387
2.10.4	操作符分述	387
2.11	类 <code>ostrstream</code>	388
2.11.1	构造与析构——公有成员	388
2.11.2	其它函数——公有成员	388
2.11.3	成员函数分述	388
2.12	类 <code>ofstream</code>	389
2.12.1	构造与析构——公有成员	389
2.12.2	操作——公有成员	389
2.12.3	状态和信息——公有成员	389
2.12.4	成员函数分述	390
2.13	类 <code>ostream</code>	392
2.13.1	构造与析构——公有成员	393
2.13.2	前缀与后缀函数——公有成员	393

2.13.3	无格式输出——公有成员	393
2.13.4	其它函数——公有成员	393
2.13.5	操作符——公有成员	393
2.13.6	操纵符	393
2.13.7	成员函数分述	394
2.13.8	操作符分述	396
2.13.9	操纵符分述	396
2.14	类 ostream_withassign	397
2.14.1	构造与析构——公有成员	397
2.14.2	操作符——公有成员	397
2.14.3	成员函数分述	397
2.14.4	操作符分述	398
2.15	类 ostream	398
2.15.1	构造与析构——公有成员	398
2.15.2	其它函数——公有成员	398
2.15.3	成员函数分述	398
2.16	类 stdiobuf	400
2.16.1	构造与析构——公有成员	400
2.16.2	其它函数——公有成员	400
2.16.3	成员函数分述	400
2.17	类 stdiostream	401
2.17.1	构造与析构——公有成员	401
2.17.2	其它函数——公有成员	401
2.17.3	成员函数分述	401
2.18	类 streambuf	402
2.18.1	字符输入函数——公有成员	402
2.18.2	字符输出函数——公有成员	402
2.18.3	构造与析构——公有成员	402
2.18.4	诊断函数——公有成员	402
2.18.5	虚拟函数——公有成员	403
2.18.6	构造与析构——保护成员	403
2.18.7	其他保护成员函数——保护成员	403
2.18.8	成员函数分述	403
2.19	类 strstream	412
2.19.1	构造与析构——公有成员	412
2.19.2	其它函数——公有成员	412
2.19.3	成员函数分述	412
2.20	类 strstreambuf	413
2.20.1	构造与析构——公有成员	413
2.20.2	其它函数——公有成员	414
2.20.3	成员函数分述	414



# 第一部分 Microsoft 运行库函数



# 引 言

Microsoft 运行库提供了在 Microsoft Windows NT 操作系统下编程所需的例程, 这些例程可以使 C 和 C++ 语言没有提供的一些通用编程任务得以自动实现。

## 0.1 兼容性

Microsoft 运行库支持美国国家标准学会(ANSI) C 和 UNIX C(在本书中, 凡涉及 UNIX 系统的说明也适用于 XENIX、其它类 UNIX 和 Windows NT 中的 POSIX 子系统)。在本书第二部分, 每一个运行库例程的描述包含有同 ANSI、UNIX 和 Win32 应用程序接口(API) 兼容部分的说明, 本运行库包含的所有例程均与 Win32 API 兼容。

### 0.1.1 同 ANSIC 的一致性

本系统中, 所有的 Microsoft 专用标识符(如函数、宏、常量、变量和类型定义)的命名习惯同 ANSI 一致。

Microsoft 专用函数名和全局变量名以单个下划线开头, 在程序中, 这些函数名和全局变量名只能被局部地取代。例如, 当程序中包含有 Microsoft 运行头文件时, 可以把一个局部变量定义为 `_open`, 而取代相同的 Microsoft 专用函数名, 但是不能把该标识符定义成程序中的全程变量或全程函数。

Microsoft 专用宏和常量的命名以两个下划线开头, 或使用一个下划线开头后紧跟一个大写字母的形式。这些标识符的使用范围是绝对的, 不能定义成其它类型的标识符。例如, 不能将 Microsoft 专用标识符 `_UPPER` 定义成其它类型的标识符。

### 0.1.2 UNIX

如果准备将程序移植到 UNIX 系统中, 就需要遵守下列准则:

- 不要删除 SYS 子目录下的头文件, 只有在不准备将程序移植到 UNIX 系统中时, 才可以将 SYS 子目录下的头文件移到他处。
- 许多例程是把表示路径和文件名的字符串作为参数的, 如果准备将程序移植到 UNIX 系统中, 就需要使用与 UNIX 系统兼容的路径分隔符。UNIX 系统在路径中只使用斜杠标记/表示路径分隔符, 而 Windows NT 系统则使用斜杠/和反斜杠\两种标记, 所以本书在 `#include` 语句中使用与 UNIX 系统兼容的斜杠标记/。然而, 对于 Windows NT 的外壳程序 `CMD.EXE` 来说, 不能在命令中使用斜杠标记。
- UNIX 系统对于路径和文件名中的大小写字母是敏感的, 而在 Windows NT 系统中的文件分配表(FAT)对于大小写不敏感。Windows NT 的安装 NT 文件系统(NTFS)在列目录时保持文件名中大小写形式不变, 而文件查找和其它系统操作则不考虑字母的大小写。

9610329

### 0.1.3 向后兼容

编译程序将同时具有旧名和新名的同一个结构看成两个不同的结构类型,不能将旧结构类型复制到新结构类型中。同样,以 **struct** 为指针的旧的原型要在原型中使用旧的 **struct** 名。

为了与 Microsoft C professional development system 6.0 和早期的 Microsoft C 版本相兼容,该产品提供了 OLDNAMES.LIB 库,在编译时,可将原先的名字映射为新名字,例如将 **open** 映射为 **\_open**,当使用下列命令组合编译时,必须显式地连接 OLDNAMES.LIB。

■ /Z1(从目标文件产生缺省库名)和 /Ze(缺省——使用 Microsoft 扩展)

■ /link(连接器控制),/NOD(非缺省——库搜索)和 /Ze

有关编译命令操作的详细信息,请参考 Visual C++ 用户手册。

## 0.2 函数和宏之间的选择

Microsoft 运行库中的大部分例程是函数,它们由编译过的 C 语言代码,或汇编过的 Microsoft 宏汇编代码构成,但有些例程是作为宏来实现的,还有一些例程既有宏版本,又有函数版本。对于既有宏版本又有函数版本的例程,在使用时,就需要有所选择。如果希望使用宏版本,只要在程序中并入含有宏版本的头文件即可,因为例程的宏定义常常出现在函数声明之后,因此宏定义一般具有优先性。如果希望使用函数版本,可以用下述两种方法实现。

■ 可以通过将例程名括在括号内,强制编译程序使用函数版本。

```
#include <ctype.h>
a = toupper(a);          //use macro version of toupper
a = (toupper)(a);       //force compiler to use function version
                        //of toupper
```

■ 使用 **# undef** 命令取消宏定义:

```
#include <ctype.h>
#undef toupper
```

函数和宏是有一些区别的,如果需要在函数和宏之间选择,可以从以下几个方面做一权衡:

■ **速度和代码大小** 使用宏的主要优点是执行速度较快。在预处理期间,宏被扩展(即由其定义所替代)产生内联代码,而函数不管被调用几次,函数定义却只出现一次。与函数相比,宏只是增加了代码大小而没有调用函数时所产生的时间上的开销。

■ **函数求值** 函数可以对一个地址求值,但宏名不能。因此,在需要函数指针的情况下,不能用宏。例如,可以声明一个指向函数的指针,但不能声明指向宏的指针。

■ **宏的副作用** 宏在对其参数值进行多次计算时,可能会错误处理其参数。例如宏 **toupper** 定义如下:

```
#define toupper(c) ((islower(c)) ? _toupper(c):(c))
```

在下例中,宏 `toupper` 会产生副作用。

```
#include <ctype.h>

int a = 'm';
a = toupper(a + +);
```

该定义使用了条件操作符(`?:`)。在条件表达式中,对参数 `c` 计算了二次:一次是检查该参数是否是小程序符,一次是产生结果。该宏将参数 `a + +` 计算了两次,使 `a` 的值增加了 2,而不是 1。结果是, `islower` 操作的值与 `-toupper` 操作的值不同。

■ 类型检查 由于可以声明函数,而不能声明宏,因此,编译程序不能像检查函数参数类型那样,对宏的参数进行类型检查,但是,如果程序中传递给宏的参数数目不正确,编译程序可以检查出来。

### 0.3 类型检查

编译程序对一些参数数目可变的函数进行有限的类型检查。这些函数是:

函数调用	参数类型检查
- <code>cprintf</code> , - <code>cscanf</code> , <code>printf</code> , <code>scanf</code>	第一个参数(格式化字符串)
<code>printf</code> , <code>fscanf</code> , <code>sprintf</code> , <code>sscanf</code>	前两个参数(文件或缓冲区和格式化字符串)
- <code>snprintf</code>	前三个参数(文件或缓冲区,数目,格式化字符串)
- <code>open</code>	前两个参数(路径和 <code>-open</code> 标志)
- <code>sopen</code>	前三个参数(路径, <code>-open</code> 标志和共享方式)
- <code>execl</code> , - <code>execle</code> , - <code>execlp</code> , - <code>execlep</code>	前两个参数(路径和第一个参数指针)
- <code>spawnl</code> , - <code>spawnle</code> , - <code>spawnlp</code> , - <code>spawnlpe</code>	前三个参数(方式标志,路径和第一个参数指针)

编译程序对上述函数对应的宽字符也做同样的有限类型检查。

### 0.4 文档约定

本书使用了下列印刷约定:

例 子	描 述
STUDIO.H	大写字母串表示文件名、段名、寄存器和用于操作系统命令级的术语。
<code>char</code> , - <code>setcolor</code> , - <code>cplusplus</code>	黑体字表示 C 和 C++ 关键字、操作符、语言专用字符,以及库例程名。
<i>expression</i>	在讨论语法时,黑体字表示文本必须按给出的形式输入。 斜体字表示在该位置由使用者提供信息,如文件名。斜体字也偶尔用于正文中的强调部分。
[ <i>option</i> ]	双重方括号内的项是可选的。
# <i>pragma pack</i> {1 2}	花括号和垂直线表示在两个或更多项中选择一项。只要双重方括号 [ ] 没有包围该花括号,就必须选择一项。
# include <io.h>	这种字形用于范例、用户输入、程序输出以及文本形式的错误信息。
CL[ <i>option</i> ... ] <i>file</i> ...	三个点(省略号)跟在一项之后,说明可以出现同一形式的多个项。

续表

例 子	描 述
while() { : }	三点排成一列或一行,表示有意省略了部分程序。
CTRL + ENTER	小号大写字母表示键盘上的键名。当两键名之间有 + 号时,表明必须同时按下这两个键。 回车键,称为 ENTER,有时在键盘上标志为弯箭头。
“argument”	在文本中第一次定义的一个新术语,用引号括起来。
“C string”	有些 C 结构,如串,需要引号。语言本身要求的引号使用“”或‘’形式,而不是“”或‘’。
▶ CEnterDlg;	箭头指向的代码表示这是由前面的例子变换而来,常常表明需要对此进行编辑。
Dynamic-Link Library (DLL)	第一次用到缩写时,将全文拼出。
<b>Microsoft 特定</b> →	某些特性具有专门使用限制时,比如 Microsoft C 实现定义的特性,而不是 ANSI 标准所规定的特性,则以箭头(→)标记,并以 END 结束标记。
<b>END Microsoft 特定</b>	

# 第1章 运行库例程分类

本章按类别列出和描述了 Microsoft 运行库的所有例程,这里的描述只是运行库例程功能的概括介绍,若要详细了解每个例程的作用、语法及使用方法,请参考本书第3章“按字母顺序排列的运行库函数”。

运行库例程的主要类别有:

参数访问例程	字符分类例程	错误处理例程
浮点数支持例程	内存分配例程	串操作例程
缓冲区操作例程	数据转换例程	异常处理例程
输入和输出例程	进程和环境控制例程	系统调用例程
字节分类例程	目录控制例程	文件处理例程
国际化例程	查找和排序例程	时间管理例程

对于多字节字符例程和宽字符例程都同对应的单字节字符例程分在同一组类别中。

## 1.1 参数访问例程

当函数参数的个数可变时,宏 `va_arg`、`va_end` 和 `va_start` 提供了对函数参数的访问功能。在头文件 `STDARG.H` 定义的这些宏同 ANSI C 兼容,在头文件 `VARARG.H` 中定义的同 UNIX V 兼容。

参数访问宏

宏	作用
<code>va_arg</code>	从参数表中检索参数
<code>va_end</code>	指针复位
<code>va_start</code>	将指针指向参数表起始处

## 1.2 缓冲区操作例程

缓冲区操作例程用于以字节为单位的内存区域中的工作。

缓冲区操作例程

例程	作用
<code>- memccpy</code>	将所指定字符及其之前的字符或指定数目的字符从一个缓冲区拷贝到另一个缓冲区。
<code>memchr</code>	在缓冲区里指定数目的字符中搜索特定字符,返回指针指向该字符首次出现的位置。
<code>memcmp</code>	比较两个缓冲区中指定数目的字符。
<code>memcpy</code>	将指定数目的字符从一个缓冲区拷贝到另一个缓冲区中。
<code>- memicmp</code>	不考虑字符的大小写(大写字符与小写字符等价),比较两个缓冲区中指定数目的字符。

续表

例程	作用
<code>memmove</code>	将指定数目的字符从一个缓冲区拷贝到另一个缓冲区。
<code>memset</code>	用指定字符初始化缓冲区中指定数目的字节。
<code>- swab</code>	交换若干字节的数据,并将它们存放在指定位置。

当源数据区和目的区重叠时,只有 `memmove` 函数能够正确地拷贝源数据区的所有内容。

### 1.3 字节分类例程

下列每一个例程测试一个多字节字符中的某一指定字节是否满足测试条件,除了特别说明,否则测试结果取决于当前使用的多字节代码页。

**注意:** 根据定义,ASCII 字符集是所有多字节字符集的一个子集。例如,日语片假名字符集既包括非 ASCII 字符,也包括 ASCII 字符。

下表中的常量均在头文件 `CTYPE.H` 中定义。

#### 多字节字符字节分类例程

例程	字节测试条件
<code>isleadbyte</code>	测试头字节;测试结果取决于当前地区的 <code>IC-CTYPE</code> 类别设置
<code>- ismbbalnum</code>	<code>isalnum</code>    <code>- ismbbkalnum</code>
<code>- ismbbalpha</code>	<code>isalpha</code>    <code>- ismbbkalnum</code>
<code>- ismbbgraph</code>	同例程 <code>- ismbbprint</code> 一样,但例程 <code>- ismbbgraph</code> 不包括空格字符(0x20)
<code>- ismbbkalnum</code>	不同于标点符号的非 ASCII 文本符号。例如,仅在代码页 932 中,该例程测试日文片假名字母
<code>- ismbbkana</code>	日文片假名(0xA1—0xDF),仅在代码页 932 中
<code>- ismbbkprint</code>	非 ASCII 文本或非 ASCII 标点符号。例如,仅在代码页 932 中,该例程测试日文片假名字母或日文片假名标点符号(0xA1—0xDF)
<code>- ismbbkpunct</code>	非 ASCII 标点符号。例如,仅在代码页 932 中,该例程测试日文片假名标点符号
<code>- ismbblead</code>	测试多字节字符的第一个字节。例如,仅在代码页 932 中,有效的范围是 0x81—0x9F,0xE0—0xFC。
<code>- ismbbprint</code>	<code>isprint</code>    <code>- ismbbkprint</code> 。该例程包括空格字符(0x20)
<code>- ismbbpunct</code>	<code>ispunct</code>    <code>- ismbbkpunct</code>
<code>- ismbbtrail</code>	测试多字节字符的第二个字节。例如,仅在代码页 932 中,有效的范围是 0x40—0x7E,0x80—0xEC
<code>- ismslead</code>	测试头字节(在字符串正文中)
<code>- ismbstrail</code>	测试尾字节(在字符串正文中)
<code>- mbctype</code>	根据前一字节返回字节类型
<code>- mbsctype</code>	返回串中的字节类型

`MB_LEN_MAX` 宏(在 `LIMITS.H` 中定义),扩展到任意一个多字节字符可能具有的



最大字节长度。**MB\_CUR\_MAX**(在 **STDLIB.H** 中定义)扩展到当前地区的任意一个多字节字符可能具有的最大字节长度。

## 1.4 字符分类例程

下列例程测试某一指定的单字节字符、宽字符或多字节字符是否满足测试条件(根据定义, ASCII 字符集是所有多字节字符集的一个子集, 例如, 日本片假名字符集包括 ASCII 字符, 也包括非 ASCII 字符)。一般情况下, 这些例程的执行速度要比用户编写的程序要快。例如, 以下代码的执行速度要比 **isalpha(c)** 慢。

```
if ((c >= 'A') && (c <= 'Z')) || ((c >= 'a') && (c <= 'z'))
    return TRUE;
```

### 字符分类例程

例 程	字符测试条件
<b>isalnum, iswalnum, _ismbcalnum</b>	测试字母数字字符
<b>isalpha, iswalpha, _ismbcalpha</b>	测试字母字符
<b>-- isascii, iswascii</b>	测试 ASCII 字符
<b>isctrl, iswctrl</b>	测试控制字符
<b>-- iscsym</b>	测试字母、下划线或数字
<b>-- iscsymf</b>	测试字母或下划线
<b>isdigit, iswdigit, _ismbcdigit</b>	测试十进制数字
<b>isgraph, iswgraph, _ismbcgraph</b>	测试除空格外的可打印字符
<b>islower, iswlower, _ismbclower</b>	测试小写字符
<b>_ismbchira</b>	测试平假名字符
<b>_ismbckata</b>	测试片假名字符
<b>_ismbclegal</b>	测试合法的多字节字符
<b>_ismbc10</b>	测试日本 0 级多字节字符
<b>_ismbc11</b>	测试日本 1 级多字节字符
<b>_ismbc12</b>	测试日本 2 级多字节字符
<b>_ismbsymbol</b>	测试非字母数字的多字节字符
<b>isprint, iswprint, _ismbcprint</b>	测试可打印字符
<b>ispunct, iswpunct, _ismbcpunct</b>	测试标点字符
<b>isspace, iswspace, _ismbcspc</b>	测试空白字符
<b>isupper, iswupper, _ismbcupper</b>	测试大写字符
<b>iswctype</b>	测试由 desc 参数确定的特性
<b>isxdigit, iswxdigit</b>	测试十六进制数字
<b>mblen</b>	返回有效的多字节字符长度, 结果取决于当前地区的 <b>LC_CTYPE</b> 类别设置