

C++常用库函数

1.缓冲区操作函数

函数名: memchr

函数原型: void *memchr(const void *buf, int c, rsize_t count);

参数: buf 缓冲区的指针; c 查找的字符; count 检查的字符个数。

所需头文件: <cstring>

功能: 查找 buf 的前 count 个字节中 c 的第一次出现, 当找到 c 或已检查完 count 个字节时停止。

返回值: 如果成功, 返回 buf 中 c 首次出现的位置的指针; 否则返回 NULL

函数名: memcpy

函数原型: void *memcpy(void *dest, const void *src, rsize_t count);

参数: dest 目的缓冲区; src 源缓冲区; count 拷贝的字符个数。

所需头文件 <cstring>

功能: 从 src 拷贝 count 个字节到 dest。如果源缓冲区和目的缓冲区重叠, 这个函数不能保证正确拷贝; 对于这种情况可使用 memmove 处理。

返回值: 返回 dest 的值。

函数名: memcmp

函数原型: int memcmp(const void *buf1, const void *buf2, unsigned int count);

参数: buf1 第一个缓冲区; buf2 第二个缓冲区; count 字符个数。

所需头文件: <cstring>

功能: 比较两个缓冲区 buf1 和 buf2 的前 count 个字符, 比较过程是大小写无关的。

返回值: buf1 和 buf2 的前 count 个字节之间的关系:

<0: buf1 小于 buf2; =0: buf1 等于 buf2; >0: buf1 大于 buf2

函数名: memmove

函数原型: void *memmove(void *dest, const void *src, rsize_t count);

参数: dest 目的对象; src 源对象; count 拷贝的字符字节个数。

所需头文件: <cstring>

功能: 从 src 拷贝 count 个字节到 dest。如果源区域与目的区域有重叠, memmove 也能确

保正确拷贝。

返回值：返回 `dest` 的值。

函数名：memset

函数原型：void *memset(void *dest, int c, size_t count);

参数：dest 目的指针；c 设置的字符；count 字符个数。

所需头文件：<cstring>

功能：设置 dest 的前 count 个字节为字符 c。

返回值：返回 dest 的值。

函数名：swab

函数原型：void swab(char *src, char *dest, int n);

参数：src 需拷贝和交换的数据；dest 交换结果数据的存储位置；n 拷贝和交换的字节个数。

所需头文件：<cstdlib>

功能：从 src 拷贝 n 个字节，交换每对相邻的字节，并把结果存储在 dest 中。一般用于为转换到使用不同字节次序的机器上而准备二进制数据。

返回值：无

2. 字符分类函数

函数名：isalnum

函数原型：int isalnum(int c);

所需头文件：<cctype>

功能：测试 c 是否字母或数字。

返回值：如果 c 在 A~Z、a~z 或 0~9 的范围内，则返回一个非0值；否则返回0。

函数名：isalpha

函数原型：int isalpha(int c);

所需头文件：<cctype>

功能：测试 c 是否字母。

返回值：如果 c 在 A~Z 或 a~z 的范围内，则返回一个非0值；否则返回0。

函数名：isascii

函数原型：int isascii(int c);

所需头文件：<cctype>

功能：测试 c 是否 ASCII 字符。

返回值：如果 `c` 为一个 `0x00~0x7F` 之间的 ASCII 字符，则返回一个非0值；否则返回0。

函数名：iscntrl

函数原型：int iscntrl(int c);

所需头文件：<cctype>

功能：测试 `c` 是否控制字符，

返回值：如果 `c` 是一个控制字符(`0x00~0x1F` 或 `0x7F`)，则返回一个非0值，否则返回0。

函数名：isalnum

函数原型：int isalnum (int c);

所需头文件：<cctype>

功能：测试 `c` 是否字母、下划线或数字。

返回值：如果 `c` 是一个字母、下划线或数字，则返回一个非0值；否则返回0。

函数名：isalpha

函数原型：int isalpha (int c);

所需头文件：<cctype>

功能：测试是否字母或下划线。

返回值：如果 `c` 是一个字母或下划线，则返回一个非0值；否则返回0。

函数名：isdigit

函数原型：int isdigit(int c);

所需头文件：<cctype>

功能：测试是否十进制数字。

返回值：如果 `c` 是一个十进制数字(`0~9`)，则返回一个非0值；否则返回0。

函数名：isgraph

函数原型：int isgraph(int c);

所需头文件：<cctype>

功能：测试是否空格外的可打印字符。

返回值：如果 `c` 是一个非空格的其它可打印字符，则返回一个非0值；否则返回0。

函数名：islower

函数原型：int islower(int c);

所需头文件：<cctype>

功能：测试是否小写字母。

返回值：如果 **c** 是一个小写字母(**a~z**)

函数名：isprint

函数原型：int isprint(int c);

所需头文件：<cctype>

功能：测试是否可打印字符。

返回值：如果 **c** 是一个可打印字符包括空格字符(**0x20~0x7E**)，则返回一个非0值；否则返回0。

函数名：ispunct

函数原型：int ispunct(int c);

所需头文件：<cctype>

功能：测试是否标点符号。

返回值：如果 **c** 是一个非空格字符并且是 **isalnum** 不为真的字符，则返回一个非0值；否则返回0。

函数名：isspace

函数原型：int isspace(int c);

所需头文件：<cctype>

功能：测试是否空白。

返回值：如果 **c** 是一个空白字符(**0x09~0x0D** 或 **0x20**) 则返回一个非0值；否则返回0。

函数名：isupper

函数原型：int isupper(int c);

所需头文件：<cctype>

功能：测试是否大写字母。

返回值：如果 **c** 是一个大写字母，则返回一个非0值；否则返回0。

函数名：isxdigit

函数原型：int isxdigit(int c);

所需头文件：<cctype>

功能：测试是否十六进制数字。

返回值：如果 **c** 是一个十六进制数字(**A~F**，**a~f** 或 **0~9**)，则返回一个非0值；否则返回0。

3.数据转换函数

函数名: abs

函数原型: int abs(int n);

参数: n 整数值。

所需头文件: <cstdlib>

功能: 求绝对值。

返回值: 返回 n 的绝对值。

函数名: atof, atoi, atol

函数原型: double atof(const char *string);

int atoi(const char *string);

long atol(const char *xstring);

参数: string 要转换的字符串。

所需头文件: <cstdlib>

功能: 将字符串转换成 double(atof)、integer(atoi)或 long(atol)型数据。

返回值: 返回转换后的结果值, 如果输入不能转换成对应类型的值, 返回值为0.0(atof)或0(atoi, atol)。溢出情况下返回值不确定。

函数名: ecvt

函数原型: char *ecvt (double value, int count, int dec, int *sign);

参数: value 被转换的数; count 存储的数字个数; dec 存储的小数点位置; sign 转换的数的符号。

所需头文件: <cstdlib>

功能: 将 double 型浮点数转换成指定长度的字符串,

返回值: 返回数字字符串的一个指针; 没有错误返回

函数名: labs

函数原型: long labs(long n);

参数: n 长整数值。

所需头文件: <cstdlib>

功能: 求 long 整数的绝对值。

返回值: 返回 n 的绝对值; 没有错误返回。

函数名: strtod

函数原型: double strtod(const char *nptr, char **endptr);

参数: `nptr` 要转换的以空字符结尾的字符串; `endptr` 停止扫描的字符的指针。

所需头文件: `<cstdlib>`

功能: 将字符串 `nptr` 转换成 `double` 型数据, 在遇到第一个不能作为数值识别的字符时停止, 这可能是结尾的空字符。

返回值: 返回转换后的结果。如果发生上溢, 函数返回十 / 一 `HUGEVAL`, `HUGEVAL` 的符号与需转换的值符号相同。如果不能进行转换或出现下溢出, 则返回0。

函数名: `strtoul`

函数原型: `long strtoul(const char *nptr, char **endptr, int base);`

参数: `nptr` 要转换的以空字符结尾的字符串; `endptr` 停止扫描的字符的指针;

`base` 使用的基数。

所需头文件: `<cstdlib>`

功能: 将字符串 `nptr` 转换成 `long` 型数据。在遇到第一个不能作为数值识别的字符时停止, 这可能是结尾的空字符, 或者是第一个大于或等于 `base` 的数值字符。

返回值: 返回转换后的结果。如果发生上溢, 函数返回 `LONGMAX` 或 `LONGMIN`。如果不能执行转换, 则返回0。

函数名: `strtoul`

函数原型: `unsigned long strtoul(const char *nptr, char **endptr, int base);`

参数: `nptr` 要转换的以空字符结尾的字符串; `endptr` 停止扫描的字符的指针; `base` 使用的基数。

所需头文件: `<cstdlib>`

功能: 将字符串 `nptr` 转换成 `unsignedlong` 型数据。在读到字符串中第一个不能作为数值识别的字符时停止, 这可能是结尾的空字符或者是大于或等于 `base` 的第一个数值字符。

返回值: 返回转换后的结果。如果发生上溢, 函数返回 `ULONGMAX`。如果不能执行转换, 则返回0。

函数名: `tolower`

函数原型: `int tolower(int c);`

参数: `c` 要转换的字符。

所需头文件: `<cstdlib>`和`<cctype>`

功能: 将字符转换为小写字母。

返回值: 返回转换结果。

函数名: `toupper`

函数原型: `int toupper(int c);`

参数: `c` 要转换的字符。

所需头文件: `<cstdlib>`和`<cctype>`

功能: 将字符转换为大写字母。

返回值: 返回转换结果。

4.数学函数

函数名: `abs`

函数原型: `int abs(int n);`

参数 `n` 需要求绝对值的整数。

所需头文件: `<cstdlib>`或`<cmath>`

功能和返回值: 返回 `n` 的绝对值; 没有错误返回

函数名: `acos`

函数原型: `double acos(double x);`

参数: `x` 是-1到1之间的值。

所需头文件: `<cmath>`

功能和返回值: 计算并返回范围在0到 π 弧度之间的`x`的反余弦值。

函数名: `asin`

函数原型: `double asin(double x);`

参数: `x` 是-1到1之间的值。

所需头文件: `<cmath>`

功能和返回值: 计算并返回范围在 $-\pi/2$ 到 $\pi/2$ 弧度之间的`x`的反正弦值。

函数名: `atan`, `atan2`

函数原型: `double atan(double x);`

`double atan2(double y, double x);`

所需头文件: `<cmath>`

功能: 计算 `x(atan)`或 `y / x(atan2)`的反正切值。

返回值: `atan` 返回 `x` 的反正切值, `atan2`返回 `y / x` 的反正切值。如果 `x` 为0, 则 `atan` 返回0。

如果 `atan2`的两个参数都为0, 该函数返回0。

函数名: `atof`

函数原型: `double atof(const char ustring);`

参数：string 需要转换的字符串。

所需头文件：<cmath>或<cstdlib>

功能和返回值：将字符串转换成 double 值并返回该值。如果 string 不能转换成 double 类型的值，返回值为0.0。

函数名：ceil

函数原型：double ceil(double x);

所需头文件：<cmath>

功能：对 x 向上取整，并以 double 型浮点数形式存储结果。

返回值：返回一个 double 型的大于或等于 x 的最小整数；没有错误返回。

函数名：cos, cosh

函数原型：double cos(double x);

参数：x 弧度值。

所需头文件：<cmath>

功能和返回值：计算并返回 x 的余弦值(cos)或双曲余弦值(cosh)。

函数名：difftime

函数原型：double difftime(timer_t timer1, timer_t timer0);

参数：timer1 终止时间；timer0 开始时间。

所需头文件：<ctime>

功能：计算两个指定时间值之间的差。

返回值：返回从 timer0到 timer1 之间经过的时间

函数名：div

函数原型：div_t div(int numer, int denom);

参数：numer 被除数；denom 除数。

所需头文件：<cstdlib>

功能：用 numer 除以 denom，计算商与余数。如果除数为0，程序输出一个错误消息并终止。

返回值：返回一个 div_t 类型的结构，它由商与余数组成。

函数名：exp

函数原型：double exp(double x);

所需头文件：<cmath>

功能和返回值：计算并返回 e 的 x 次幂。

函数名：fabs

函数原型：double fabs(double x);

所需头文件：<cmath>

功能和返回值：计算并返回浮点参数 x 的绝对值。

函数名：floor

函数原型：double floor(double x);

所需头文件：<cmath>

功能：向下取整，并以 double 型浮点数形式存储结果。

返回值：返回一个 double 型的小于或等于 x 的最大整数；没有错误返回。

函数名：fmod

函数原型：double fmod(double x, double y);

所需头文件：<cmath>

功能和返回值：计算并返回 x / y 的余数，如果 y 值是 0.0，返回一个静止 NaN。

函数名：frexp

函数原型：double frexp(double x, int* expptr);

参数： x 需要求出尾数和指数的浮点数； $expptr$ 指向指数值的指针

所需头文件：<cmath>

功能：取得一个浮点数的尾数和指数。

返回值：返回尾数。如果 x 为 0，尾数和指数都为 0。

函数名：hypot

函数原型：double hypot (double x, double y);

参数：直角三角形的两个直角边长度。

所需头文件：<cmath>

功能和返回值：计算并返回直角三角形的斜边长度(x 与 y 的平方根)，上溢出时返 INF(无穷大)。

函数名：labs

函数原型：long labs(long n)

所需头文件：<stdlib.h>

功能和返回值：返回 long 型参数 n 的绝对值

函数名：ldexp

函数原型：double ldexp(double x, int exp);

参数：x 尾数；exp 指数。

所需头文件：<cmath>

功能和返回值：计算并返回变量x和2的指定乘方的乘积($x*2^{exp}$)。

函数名：ldiv

函数原型：ldiv_t ldiv(long numer, long denom);

参数：numer 被除数；denom 除数。

所需头文件：<stdlib.h>

功能：用 numer 除以 denom，计算商与余数。如果除数为0，程序输出一个错误消息并终止。

返回值：返回一个 ldiv_t 类型的结构，它由商和余数组成，定义在 stdlib 中。

函数名：log

函数原型：double log(double x);

所需头文件：<cmath>

功能和返回值：计算并返回 x 的自然对数。如果 x 是负数，返回值不确定。如果 x 为0，返回 INF(无穷大)。

函数名：log10

函数原型：double log10(double x);

所需头文件：<cmath>

功能和返回值：计算并返回 x 的以10为底的对数。如果 x 是负数，返回值不确定。如果 X 为0，返回 INF(无穷大)。

函数名：logb

函数原型：double logb (double x);

所需头文件：<float.h>

功能和返回值：返回双精度浮点参数 x 的无偏的指数值。

函数名：lrotl, lrotr

函数原型：unsigned long lrot (unsigned long value, int shift)

`unsigned long lrotr (unsigned long value, int shift);`

参数: `value` 需要移位的数值; `shift` 需要移动的位数。

所需头文件: `<cstdlib>`

功能: 循环移动 `value` 值 `shift` 位。

返回值: 返回循环移位后的值。

函数名: `max`

函数原型: `type max (type a, type b);`

参数: `type` 任何数值数据类型; `a` 和 `b` 是参与比较的两个数, 必须是相同类型。

所需头文件: `<cstdlib>`

功能和返回值: 比较 `a` 和 `b` 并返回其中较大者。

函数名: `min`

函数原型: `type min (type a, type b);`

参数: `type` 任何数值数据类型。 `a` 和 `b` 是参与比较的两个数, 必须是相同类型。

所需头文件: `<cstdlib>`

功能和返回值: 比较 `a` 和 `b` 并返回其中较小者。

函数名: `modf`

函数原型: `double modf(double x, double *inptr);`

参数: `x` 需要分解的数; `inptr` 指向分解后整数部分的指针。

所需头文件: `<cmath>`

功能和返回值: 将浮点值 `x` 分解成小数和整数部分, 每个都与 `x` 具有同样的符号。返回 `x` 的带符号的小数部分, 整数部分作为浮点值存储在 `inptr` 处。

函数名: `nextafter`

函数原型: `double nextafter (double x, double y);`

所需头文件: `<float>`

功能和返回值: 返回 `x` 与 `y` 之间, 与 `x` 最邻近的可表示的浮点数。如果 `x=y`, `nextafter` 返回 `x`, 没有异常触发。

函数名: `pow`

函数原型: `double pow(double x, double y);`

所需头文件: `<cmath>`

功能和返回值: 计算并返回 `x` 的 `y` 次幂。

函数名: printf

函数原型: int printf(const char *format[, argument]...);

参数: format 格式控制字符串; argument 待输出的内容, 任选参数。

所需头文件: <stdio>

功能: 格式化并输出一系列字符和数值到标准输出流 stdout。如果有参数 argument 跟随 format 字符串, 该 format 字符串必须包含确定该参数输出格式的格式符。

返回值: 返回输出的字符个数; 如果出现错误, 则返回一个负数。

函数名: rand

函数原型: int rand(void);

所需头文件: <stdlib>

功能和返回值: 返回一个 0 ~ RAND_MAX 的随机数

函数名: scanf

函数原型: int scanf(const char *format[, argument)...);

参数: format 格式控制字符串; argument 可选参数, 表示输入内容的存放地址。

所需头文件: <stdio>

功能: scanf 函数从标准输入流 stdin 读数据并把所读数据写到 argument 指定的位置。每个 argument 必须是对应于 format 中一个类型指示符的类型的变量的一个指针。

返回值: 返回成功转换和赋值的域的个数。

函数名: sin, sinh

函数原型: double sin(double x);

double sinh(double x);

参数: x 弧度值。

所需头文件: <cmath>

功能和返回值: sin 返回 x 的正弦值。sinh 返回 x 的双曲正弦值。

函数名: sqrt

函数原型: double sqrt(double x);

所需头文件: <cmath>

功能和返回值: 计算并返回 x 的平方根。

函数名: srand

函数原型: void srand(unsigned int seed);

参数: seed 产生随机数的种子。

所需头文件: <stdlib>

功能: 为使 rand()产生一序列伪随机整数而设置起始点。使用1作为 seed 参数, 可以重新初始化 rand()。

函数名: tan, tanh

函数原型: double tan(double x);

double tanh(double x);

参数: x 弧度值。

所需头文件: <cmath>

功能和返回值: tan 返回 x 的正切值。tanh 返回 x 的双曲正切值。

5.输入和输出函数

函数名: fclose

函数原型: int fclose(FILE *stream);

参数: stream FILE 结构的指针。

所需头文件: <stdio>

返回值: 如果该流成功关闭, fclose 返回0。如果出错, 则返回 EOF。

功能: 关闭流。

函数名: ferror

函数原型: int ferror(FILE *stream);

参数: stream FILE 结构的指针。

所需头文件: <stdio>

功能: 测试与 stream 关联的文件上的读写错误。如果出现一个错误, 该函数将设置该流的错误指示符并且保留到该流被关闭、重绕或者后面调用了 clearerr 函数。

返回值: 如果 stream 上没有出现错误, ferror 返回0; 否则返回一个非0值。

函数名: fprintf

函数原型: int fprintf(FILE *stream, const char *format[, argument]...);

参数: stream FILE 结构的指针; format 格式控制字符串; argument 需要输出的内容。

所需头文件: <stdio>

功能: 格式化并输出一系列字符和数值到输出流 stream 中。每个参数 argument(如果存在)根据 format 中对应的格式规范转换和输出。format 参量与 printf 中的该参量具有相同的语

法和用途。

返回值：返回所写的字节数，当出现错误时函数返回一个负数。

函数名：freopen

函数原型：FILE *freopen(const char *path, const char *mode, FILE *stream);

参数：path 新文件的路径；mode 文件访问许可；stream FILE 结构的指针。

所需头文件：<stdio.h>

功能：关闭当前与 stream 关联的文件，并将 stream 重新赋给由 path 指定的文件。

返回值：返回最新打开的文件的指针。如果出现错误，最初的文件被关闭并返回 NULL 指针值。

函数名：fscanf

函数原型：int fscanf(FILE *stream, const char *format[, argument]...);

参数：stream FILE 结构的指针；format 格式控制字符串。format 控制对输入内容的格式转换，它与 scanf 中的 format 参量具有同样的格式和功能。每个 argument 必须是对应于 format 中一个类型指示符的类型的指针，作为读入数据的存放地址，是可选参量。

所需头文件：<stdio.h>

功能：从 stream 的当前位置读数据到 argument 值定的位置(如果有)。

返回值：返回成功转换和存储的域个数，返回值不包括被读但没有分配存储位置的域。返回值 0 指出所有域都没有分配存储位置。如果出现错误，或者在转换之前到达文件流末尾，则返回值为 EOF。

函数名：getchar

函数原型：int getchar(void);

所需头文件：<stdio.h>

功能和返回值：从 stdin 读取一个字符并返回所读字符，当出现读错误或遇到文件结尾时返回 EOF。

函数名：gets

函数原型：char *gets(char *buffer);

参数：buffer 输入字符串的存储位置。

所需头文件：<stdio.h>

功能：从标准输入流 stdin 读取一行，并存储在 buffer 中。该行由直到第一个换行符('\n')的所有字符组成，并包括该第一个换行符，然后 gets 在返回该行之前用空字符('\0')代替换行符。

返回值：如果成功，返回 **buffers** 如果有错误或遇到文件结尾则返回 **NULL** 指针。

函数名：printf

函数原型：int printf(eonst char *format[, argument]...);

参数：format 格式控制字符串；argument 待输出的内容，任选参数。

所需头文件：<stdio. h>

功能：格式化并输出一系列字符和数值到标准输出流 **stdout**。如果有参数 **argument** 跟随 **format** 字符串，该 **format** 字符串必须包含确定该参数输出格式的格式符。

返回值：返回输出的字符个数，如果出现错误则返回一个负数。

函数名：putc

函数原型：int putc(int c, FILE *stream);

参数：c 要写的字符；streamFILE 结构指针。

所需头文件：<cstdio>

功能：写一个字符到流 **stream** 中。

返回值：返回所写的字符；如果出现错误，返回 **EOF**。

函数名：putchar

函数原型：int putchar(int c);

参数：c 要写的字符。

所需头文件：<cstdio>

功能：写一个字符到 **stdout** 中。

返回值：返回所写的字符；如果出现错误，返回 **EOF**。

函数名：puts

函数原型：int puts(const char *string);

参数：string 要输出的字符串。

所需头文件：<cstdio>

功能：将 **string** 写到标准输出流 **stdout**，在输出流中用换行符（‘\n’）代替字符串的结尾的空字符（‘\0’）。

返回值：如果成功，返回一个非负值；如果失败，返回 **EOF**。

函数名：scanf

函数原型：int scanf(const char *format[, argument]...);

参数：format 格式控制字符串，控制对输入内容的格式转换。每个 **argument** 必须是对应于

format 中一个类型指示符的类型的指针，作为读入数据的存放地址，是可选参量。

所需头文件：<cstdlib.h>

功能：从标准输入流 **stdin** 读数据，并把所读数据写到 **argument** 给定的位置

返回值：返回成功转换和存储的域的个数。返回值不包括已读但未存储的域

出所有域都没有分配存储位置。错误时返回值为 **EOF**。

函数名：sprintf

函数原型：int sprintf(char *buffer, const char *format[, Argument] ...);

参数：buffer 要写入数据的目标地; format 格式控制字符串; argument 要格式化并写入 buffer 的数据项，是任选参数。

所需头文件：<stdio.h>

功能：将数据格式化后写到字符串中：将每个 **argument** 按照 **format** 指定的格式转换成字符串并存储在从 **buffer** 开始的内存中。这里的格式符与 **printf** 中 **format** 参数具有同样的格式和功能。如果被格式化和存储的字符串与目的字符串之间有重叠，则此函数的执行效果是不确定的。

返回值：返回存储在 **buffer** 中的字节数，不包含尾部的空字符。

函数名：sscanf

函数原型：int sscanf(const char *buffer, const char *format[, Argument] ...);

参数：buffer 存储要被读取并转换的数据。format 格式控制字符串。每个 **argument** 必须是对应于 **format** 中一个类型指示符的类型的指针，作为读入数据的存放地址，是可选参量。

所需头文件：<stdio.h>

功能：按 **format** 指定的格式，由 **buffer** 读取字符数据并转换后存储到每个 **argument** 指定的位置中。每个 **argument** 必须是与 **format** 中的类型指示符对应的类型变量的指针。format 与 **scanf** 函数的 **format** 参数具有同样的格式和功能。

返回值：返回成功转换和存储的数据个数。返回的值不包括已读但未存储的域。返回值0指出所有域都没有分配存储位置。如果出现错误或在第一个转换之前到达字符串结尾，则返回值是 **EOF**。

5.进程控制函数

函数名：exit

函数原型：void exit(int status);

参数：status 退出状态。

所需头文件：<stdlib.h>

功能：终止进程。

函数名: system

函数原型: int system(const char*command)

参数: command 要执行的命令。

所需头文件: <cstdlib>

功能: 把 command 传给命令解释器, 像执行操作系统命令那样执行该字符串。

返回值: 返回该命令解释器所返回的值, 且当该命令解释器返回0时它返回0。返回值-1指出一个错误。

6.字符串操作函数

函数名: strcat

函数原型: char *strcat(char*strDestination, const char *strSource);

参数: strDestination 以空字符结尾的目的字符串 strSource 以空字符结尾的源字符串。

所需头文件: <cstring>

功能: 将 strSource 添加到 strDestination, 并用一个空字符结束该结果字符串。用 strSource 的首字符覆盖 strDestination 的结尾空字符。当字符串被拷贝或添加时不执行上溢出检测。

如果源和目的字符串重叠, strcat 的行为是不确定的。

返回值: 返回目的字符串。

函数名: strchr

函数原型: char*strchr(constchar*string, int c);

参数: string 以空字符结尾的源字符串 c 要查找的字符。

所需头文件: <cstring>

功能: 查找 string 中 c 的第一次出现, 在查找中包括结尾的空字符。

返回值: 返回 string 中第一次出现的指针; 如果 c 未找到, 则返回 NULL。

函数名: strcmp

函数原型: int strcmp(const char*string1, constchar*string2)

参数: string1, string2被比较的以空字符结尾的字符串。

所需头文件: <cstring>

功能: 按词典顺序比较 string1 和 string2, 并返回一个值指出它们之间的关系。

返回值: 返回值<0, string1 小于 string2; 返回值=0, string1 等于 string2; 返回值>0, string1 大于 string2。

函数名: strcpy

函数原型: char*strcpy(char*strDestination, constchar*strSource)

参数: strDestination 目的字符串;strSource 以空字符结尾的源字符串。

所需头文件: <cstring>

功能: 把源字符串 strSource(包括结尾的空字符)拷贝到 strDestination 所指的位置。在字符串被拷贝或添加时不执行上溢出检测。如果源和目的字符串重叠, strcpy 的行为是不确定的。

返回值: 返回目的字符串, 没有用于指出错误的返回值

函数名: strcmp

函数原型: int strcmp(const char*string1, const char *string2);

参数: string1, string2要比较的以空字符结尾的字符串。

所需头文件: <cstring>

功能: 忽略大小写来比较两个字符串。_stricmp 函数以词典次序比较 string1 和 string2的小写版本, 并返回一个值指出它们之间的关系。

返回值: 返回值<0, string1 小于 string2; 返回值=0, string1 等于 string2; 返回值>0, string1 大于 string2。

函数名: strlen

函数原型: sizet strlen(const char*string);

参数: string 以空字符结尾的字符串。

所需头文件: <cstring>

功能和返回值: 返回 string 中的字符个数, 不包括尾部 NULL。没有指出错误的返回值。

函数名: strlwr

函数原型: char* strlwr(char *string);

参数: string 需要转换成小写的以空字符结尾的字符串。

所需头文件: <cstring>

功能: 将 string 中的任何大写字母转换成小写, 其它字符不受影响。

返回值: 返回转换后的字符串的指针。因为不修改位置的指针相同。没有返回值指出错误。

函数名: strncmp

函数原型: int strncmp(constchar*string1, constchar*string2, size_t count)

参数: string1, string2比较的字符串; count 比较的字符的个数。

所需头文件: <cstring>

功能: 按词典顺序比较 string1 和 string2的前 count 个字符, 并返回一个值指出串之间的关系。大小写敏感。

返回值: <0, string1 串小于 string2串。=0, string1 串等于 string2串; >0, string1 大于 string2 串。

函数名: strncpy

函数原型: char*strncpy(char*strDest, const char*strSource, size_t count)

参数: strDest 目的字符串; strSource 源字符串; count 拷贝的字符个数。

所需头文件: <cstring>

功能: 将 strSource 的前 count 个字符拷贝到 strDest 中并返回 strDest。如果 count 小于或等于 strSource 的长度, 空字符不自动添加到拷贝的字符串中。如果 count 大于 strSource 的长度, 目的字符串用空字符填充直到 count 的长度。如果源和目的字符串重叠, 则 strncpy 的行为是不确定的。

返回值: 返回 strDest。没有返回值则表明出错。

函数名: _strnset

函数原型: char *_strnset(char*string, int c, size_t count);

参数: string 需要改变的字符串; c 设置字符; count 设置的字符个数

所需头文件: <string. h>

功能: 将 string 的前 count 个字符设置为 c(转换为 char)。如果 count 大于 string 的长度,用 string 的长度代替 count。

返回值: 返回一个改变后的字符串的指针。

函数名:strrev

函数原型: char *strrev(char *string);

参数: string 要逆转的以空字符结尾的字符串。

所需头文件: <cstring>

功能: 将 string 中字符反序排列。结尾的空字符保留在原位置。

返回值: 返回改变后的字符串的指针。没有返回值则说明出错。

函数名: strstr

函数原型: char*strstr(constchar*string, constchar*strCharSet);

参数: string 要在其中进行查找的以空字符结尾的字符串;strCharSet 要查找的以空字符结尾的字符串。

所需头文件: <cstring>

功能和返回值: 返回 strCharSet 在 string 中第一次出现的起始地址, 如果 strCharSet 不在 string 中出现, 则返回 NULL。