Keil c51 函数参考

一、数学函数<math.h >

(1)cabs

原型：char cabs(char val);

功能：cabs 函数取 val 的绝对值。

参数：一字节数 val。

返回：cabs 返回 val 的绝对值。

(2)abs

原型：Int abs(int val);

功能：求绝对值

参数：val 整型数

返回：val 的绝对值

(3)labs

原型：long labs(long val);

功能：labs 函数确定长整数 val 的绝对值。

返回：val 的绝对值。

(4)fabs

原型：float fabs(float val);

功能：fabs 函数确定浮点数 val 的绝对值。

参数：

返回：fabs 返回 val 的绝对值。

(5)sqrt

原型：float sprt(float x);

功能：sqrt 函数计算 x 的平方根。

返回：sqrt 函数返回 x 的正平方根。

(6)exp

原型：float exp(float x);

功能：exp 函数计算自然对数中 e 的 x 次幂。e≈2.71828182845953581496，是无限循环小数。

返回：e^x 的值。

(7)log

原型：float log(float val);

功能：log 函数计算浮点数 val 的自然对数。自然对数基数为 e。

返回：val 的浮点自然对数。

(8)log10

原型：float log10(float val);

功能：logl0 函数计算浮点数 val 的常用对数。常用对数为基数 10。

返回：val 的浮点常用对数。

(9)sin

原型：float sin(float x);

功能：sin 函数计算浮点数 x 的正弦值。

参数：x 必须在-65535～65535 之间，或产生一个 NaN 错误。

返回：sin 函数返回 x 的正弦。

(10)cos

原型：float cos(float x);

功能：COS 函数计算浮点数 X 的余弦。

参数：X 的值必须在-65535～65535 之间，或产生一个 NaN 错误。

返回：COS 函数返回 X 的余弦。

(11)tan

原型：float tan(float x);

功能：tan 函数计算浮点数 x 的正切值。

参数：x 必须在-65535～65535 之间，或错误值 NaN。

返回：tan 函数返回 x 的正切。

(12)asin

原型：float asin(float x);

功能：求反正弦

参数：浮点数 x，取值必须在-1～1 之间。

返回：X 的反正弦，值在-π /2～π /2 之间。

(13)acos

原型：float acos(float x);

功能：求反余弦

参数：浮点数 x，取值必须在-1～1 之间。

返回：x 的反余弦，值在 0～π 之间。

(14)atan

原型：float atan(float x);

功能：求反正切

参数：浮点数 x，取值必须在-1～1 之间。

返回：X 的反正切，值在-π /2～π /2 之间。

(15)sinh

原型：float sinh(float x);

功能：sinh 函数计算浮点数 X 的双曲正弦。

参数：x 必须在-65535～65535 之间，或产生一个 NaN 错误。

返回：sinh 函数返回 x 的双曲正弦。

(16)cosh

原型：float cosh(float x);

功能：cosh 函数计算浮点数 X 的双曲余弦。

参数：

返回：cosh 函数返回 X 的双曲余弦。

(17)tanh

原型：float tanh(float x);

功能：tanh 函数计算浮点数 x 的双曲正切。

返回：tanh 函数返回 x 的双曲正切。

(18)atan2

原型：float atan2(float y,float x);

功能：计算浮点数 y/x 的反正切。

参数：浮点数 y，浮点数 x。

返回：反正切值，值在-π ～π 之间。x 和 y 的符号确定返回值的象限。

(19)ceil

原型：float ceil(float val)

功能：ceil 函数计算大于或等于 val 的最小整数值（收尾取整）。

参数：要化为整数的数。

返回：ceil 函数返回不小于 val 的最小 float 整数值。

(20)floor

原型：float floor(float val);

功能：取整。

返回：floor 函数返回不大于 val 的最大整数值。

(21)fmod

原型：float fmod(float x,float y);

功能：取模。

返回：x/y 的浮点余数。

(22)modf

原型：float modf(float val,float *ip);

功能：modf 函数把浮点数 val 分成整数和小数部分。

返回：modf 函数返回带符号小数部分 val。整数部分保存在浮点数 ip 中。

(23)pow

原型：float pow(float x,float y);

功能：pow 函数计算 x 的 y 次幂。

返回：pow 函数返回值 x^y。如果 x≠0 和 y=O，pow 返回值 1；如果 x=0 和 y≤0，pow 返回 NaN。如果 x<O 和 y 不是一个整数，pow 返回 NaN。

二、空操作，左右位移等内嵌代码<intrins.h>

(1)_nop_

原型：void _nop_(void);

功能：_nop_插入一个 8051NOP 空操作指令到程序，用来停顿 1 个 CPU 周期。本程序是固有函数，代码要求内嵌而不是调用。

返回：无。

(2)_testbit_

原型：bit _testbit_(bit b);

功能：_testbit_程序在生成的代码中用 JBC 指令来测试位 b，并清零。

参数：本程序只能用在直接寻址位变量，对任何类型的表达式无效。固有函数，代码要求内嵌，而非调用。

返回：_testbit_程序返回值 b

(3)_cror_

原型：unsigned char _cror_(unsigned char c, unsigned char b);

功能：_cror_程序字符 c 循环右移 b 位。固有函数，代码要求内嵌，而不是调用。

参数：

返回：右移的结果

(4)_iror_

原型：unsigned int _iror_(unsigned int i, unsigned char b);

功能：_iror_程序将整数 i 循环右移 b 位。固有函数，代码要求内嵌而不是被调用。

参数：i 右移的整数，b 右移的次数。

返回：_iror_程序返回右移后的值。

(5)_lror_

原型：unsigned long _lror_(unsigned long l,unsigned char b);

功能：_lror_程序将长整数 l 循环右移 b 位。固有函数代码，要求内嵌而不是被调用。

参数：l 要右移的数，b 要右移的位数。

返回：返回右移后的值。摘要：#include<intrins_h>。

(6)_crol_

原型：unsigned char _crol_(unsigned char c, unsigned char b);

功能：_crol_程序字符 c 循环左移 b 位。固有函数，代码要求内嵌，而不是调用。

参数：

返回：左移的结果

(7)_irol_

原型：unsigned int _irol_(unsigned int i, unsigned char b);

功能：_irol_程序将整数 i 循环左移 b 位。固有函数，代码要求内嵌而不是被调用。

参数：i 左移的整数，b 左移的次数。

返回：_irol_程序返回左移后的值。

(8)_lrol_

原型：unsigned long _lrol_(unsigned long l,unsigned char b);

功能：_lrol_程序将长整数 l 循环左移 b 位。固有函数，代码要求内嵌而不是被调用。

参数：l 要左移的数，b 要左移的位数。

返回：返回左移后的值。

(9)_chkfloat_

原型：unsigned char _chkfloat_(float val);

功能：检查浮点数的状态。

参数：浮点型变量。

返回：0，标准浮点数；1，浮点数 0；2，正溢出；3，负溢出；4，NaN(不是一个数)错误状态。

(10)_push_

原型：void _push_(unsigned char _sfr);

功能：将特殊功能寄存器_sfr 压入堆栈。

(11)_pop_

原型：void _pop_(unsigned char _sfr);

功能：将堆栈中的数据弹出到特殊功能寄存器_sfr。

三、字串转数字，随机数，存储池管理<stdlib.h>

(1)atof

原型：float atof(void *string);

功能：将浮点数格式的字符串转换为浮点数。如果 string 的第一个字符不能转换成数字，就停止处理。

参数：格式为，[{+|-}]数字[.数字][{e|E}[{+|-}]数字]。如，-12.345e+67

返回：atof 函数返回 string 的浮点值。

(2)atoi

原型：int atoi(void *string);

功能：atoi 函数转换 string 为一个整数值。string 是一个字符序列，可以解释为一个整数。如果 string 的第一个字符不能转换成数字，就停止处理。

参数：atoi 函数要求 string 有这样的格式：[空格][{+|-}]数字，如”123456”。

返回：atoi 函数返回 string 的整数值。

(3)atol

原型：long atol(void *string);

功能：atol 函数转换 string 为一个长整数值。string 是一个字符序列，可以解释为一个长整数。如果 string 的第一个字符不能转换成数字，就停止处理。

参数：atoi 函数要求 string 有这样的格式：[空格][{+|-}]数字，如”1234567890”。

返回：atol 函数返回 string 的长整数值。

(4)rand

原型：int rand(void);

功能：rand 函数产生一个 0～32767 之间的虚拟随机数。

返回：rand 函数返回一个虚拟随机数。

(5)srand

原型：void srand(int seed);

功能：srand 函数设置 rand 函数所用的虚拟随机数发生器的起始值 seed，随机数发生器对任何确定值 seed 产生相同的虚拟随机数序列。

返回：无。

(6)strtod

原型：unsigned long strtod(const char *string, char **ptr);

功能：strtod 函数将一个浮点数格式的字符串 string 转换为一个浮点数。字符串开头的空白字符被忽略。

参数：要求 string 有下面的格式：

[{+|-}]digits[.digits][{e|E}[{+|-}]digits]

digits 可能是一个或多个十进制数。

ptr 的值设置指针到 string 中转换部分的第一个字符。如果 ptr 是 NULL，没有值和 ptr 关联。如果不能转换，则 prt 就设为 string 的值，strtod 返回 0。

返回：strtod 函数返回由 string 生成的浮点数。

(7)strtol

原型：long strtol(const char *string, char **ptr, unsigned char base);

功能：strtol 函数将一个数字字串 string 转换为一个 long 值。

参数：输入 string 是一个字符序列，可以解释为一个整数。字符串开头的空白字符被忽略，符号可选。要求 string 有下面的格式：

[whitespace][{+|-}]digits

digits 可能是一个或多个十进制数。

如果 base 是零，数值应该有一个十进制常数、八进制常数或十六进制常数的格式。数值的基数从格式推出。如果 base 在 2～36 之间，数值必须是一个字母或数字的非零序列，表示指定基数的一个整数。字母 a～z(或 A～Z)分别表示值 10～36。只有小于 base 的字母表示的值是允许的。如果 base 是 16，数值可能以 0x 或 0X 开头，0x 或 0x 被忽略。

prt 的值设置指针指向 string 中转换部分的第一个字符。如果 prt 是 NULL，没有值和 ptr 关联。如果不能转换，prt 设置为 string 的值，strtol 返回 0。

返回：strtol 函数返回 string 生成的整数值。如溢出则返回 LONG_MIN 或 LONG_MAX。

(8)strtoul

原型：unsigned long strtoul(const char *string, char **ptr, unsigned char base);

功能：strtoul 函数转换 string 为一个 unsigned long 值。

参数：与 strtol 函数类似。

返回：strtoul 函数返回 string 生成的整数值。如溢出则返回 ULONG_MAX。

(9)init_mempool

原型：void inti_mempool(void xdata *p, unsigned int size);

功能：init_mempool 函数初始化存储管理程序，提供存储池的开始地址和大小。本函数必须在任何其他的存储管理函数(calloc，free，malloc，realloc)被调用前设置存储池，只在程序的开头调用一次。可以修改源程序以适合硬件环境。

参数：p 参数指向一个 xdata 的存储区，用 calloc，free，malloc 和 realloc 库函数管理。size 参数指定存储池所用的字节数。

返回：无。

(10)malloc

原型：void xdata *malloc(unsigned int size);

功能：malloc 函数从存储池分配 size 字节的存储块。

返回：malloc 返回一个指向所分配的存储块的指针，如果没有足够的空间，则返回一个 NULL 指针。

(11)free

原型：void free(void xdata *p);

功能：free 函数返回一个存储块到存储池。p 参数指向用 calloc、malloc 或 realloc 函数分配的存储块。一旦块返回到存储就可被再分配。如果 p 是一个 NULL 指针，被忽略。本程序的源代码在\KEIL\C51\LIB 目录中，可以修改源程序，根据硬件来定制本程序。

参数：

返回：无。

(12)realloc

原型：void xdata *realloc(void xdata *p,unsigned int size);

功能：realloc 函数改变已分配的存储块的大小。本程序的源代码在目录 KEIL\C5I\LIB 中，可以根据硬件环境定制本函数。

参数：P 参数指向已分配块，size 参数指定新块的大小。原块的内容复制到新块，新块中的任何其他区，如果是一个更大的块不初始化。

返回：realloc 返回一个指向新块的指针。如果存储池没有足够的存储区，返回一个 NULL 指针，存储块不受影响。

(13)calloc

原型：void xdata *calloc(unsigned int num,unsigned int len);

功能：calloc 函数从一个数组分配 num 个元素的存储区。每个元素占用 len 字节，并清 0。字节总数为 num*len。在 LIB 目录提供程序的源代码。可以修改源程序，为硬件定制本函数。

参数：num 为元素数目，len 为每个元素的长度。

返回：calloc 函数返回一个指针，指向分配的存储区，如果不能分配，则返回一个 NULL 指针。

四、流输入输出<stdio.h>

(1)_getkey

原型：char _getkey(void);

功能：_getkey 函数等待从串口接收字符。_getkey 和 putchar 函数的源代码可以修改，提供针对硬件的字符级的 I/O。

返回：接收到的字符

(2)getchar

原型：char getchar(void);

功能：getchar 函数用_getkey 函数从输入流读一个字符。所读的字符用 putchar 函数显示。本函数基于_getkey 或 putchar 函数的操作。这些函数，在标准库中提供，用 8051 的串口读和写字符。定制函数可以用别的 I/O 设备。

返回：所读的字符。

(3)ungetchar

原型：char ungetchar(char c);

功能：ungetchar 函数把字符 c 放回到输入流。子程序被 getchar 和别的返回 c 的流输入函数调用。getchar 在调用时只能传递一个字符给 ungetehar。

参数：

返回：如果成功，ungetchar 函数返回字符 c。如果调用者在读输入流时调用 ungetchar 多次，返回 EOF 表示一个错误条件。

(4)putchar

原型：char putchr(char c);

功能：putchar 函数用 8051 的串口输出字符 c。本程序指定执行，功能可能有变。因提供了_getkey 和 putchar 函数的源程序，可以根据任何硬件环境修改以提供字符级的 I/O。

参数：

返回：putchar 函数返回输出的字符 c。

(5)printf

原型：int printf(const char *fmtstr [,arguments]…);

功能：printf 函数格式化一系列的字符串和数值，生成一个字符串用 putchar 写到输出流。

参数：fmtstr 参数是一个格式化字符串，可能是字符、转义系列和格式标识符。普通的字符和转义系列按说明的顺序复制到流。格式标识符通常以百分号(%)开头，要求在函数调用中包含附加的参数 Arguments。

格式字符串从左向右读。第一个格式标识符使用 fmtstr 后的第一个参数，用格式标识符转换和输出。第二个格式标识符访问 fmtstr 后的第二个参数。如果参数比格式标识符多，多出的参数被忽略。如果参数不够，结果是不可预料的。

格式标识符用下面的格式：

%[flags][width][.precision][{b|B|l|L}]type

格式标识符中的每个域可以是一个字符或数字

type 域是一个字符，指定参数是否解释为一个字符、字符串、数字或指针。如下表所示：

字　符	参　数	参 数 类 型
D	int	带符号十进制数
U	unsigned int	不带符号十进制数
O	unsigned int	不带符号八进制数
x	unsigned int	不带符号十六进制数，用“0123456789abcdef”
X	unsigned int	不带符号十六进制数，用“0123456789ABCDEF”
f	float	浮点数用格式[-]dddd.dddd
e	float	浮点数用格式[-]d.ddde[-]dd
E	float	浮点数用格式[-]d.ddddE[-]dd
g	float	浮点数用 e 或 f 格式，无论哪个对指定的值或精度更简洁
G	float	和 g 格式一样除了（可能）指数前为 E 而不是 e
c	char	单个字符
s	通用 *	用 NULL 字符结尾的字符串
p	通用 *	指针，用 t:aaaa 格式，这里 t 是指针索引的存储类型（c:code，i:data/idata，x:xdata，p:pdata），aaaa 是十六进制地址

可选的字符 b 或 B 和 l 和 L 可直接放在类型字符前，分别指定整数类型 d、i、u、o、x 和 X 的 char 或 long 版本。

flags 域是单个字符，用来对齐、输出和打印+/-号、空白、小数点、八进制和十六进制的前缀。如下表所示：

标　记	意　义
-	对指定的域宽度，左对齐输出
+	如果输出是一个带符号类型，用+或-符号作为输出值的前缀
空白（' '）	如果是一个带符号正值，输出值的前缀是空格。否则没有空格
#	当用 o、x 和 X 域类型时，对非零输出值分别用 0、0x 或 0X 为前缀 当用 e、E、f、g 和 G 域类型时，#标记强迫输出值包含一个小数点。在其他的情况#标记被忽略
*	忽略格式标识符

width 域是一个非负数字，指定显示的最小字符数。如果输出值的字符数小于 width，空白会加到左边或右边(当指定了一个标记)以达到最小的宽度。如果 width 用一个’0’作前缀，则填充的是零而不是空白。width 域不会截短一个域。如果输出值的长度超过指定宽度，则输出所有的字符。

width 域可能是星号(*)，在这种情况下，参数列表的一个 int 参数提供宽度值。如果参数使用的是 unsigned char，在星号标识符前指定一个’b’。

precision 域是非负数字，指定显示的字符数、小数位数或有效位。precison 域可能使输出值切断或舍入。

类　型	意　义
d,u,o,x,X	precision 域用来指定输出值的数字的最小数目。如果参数中的数字数目超过 precision 域定义的，数字就不会被切断。如果参数的数字的数目小于 precision 域，输出值在左边用零填充
f	precision 域用来指定小数点后面的数字的位数，最后一位舍入
e,E	precision 域用来指定小数点后面的数字的位数，最后一位舍入
g,G	precision 域用来指定输出值的有效位的最大数目
c,p	precision 域无效
S	precision 域指定输出值的最多字符数，超过的不输出

precision 域可能是星号(*)，在这种情况，参数列表的一个 int 参数提供宽度值。如果参数使用的是 unsigned char，在星号标识符前指定一个'b'。

本函数指定执行基于 putchar 函数的操作。本函数作为标准库提供，用 805l 的串口写字符，用别的 I/O 设备可以定制函数。

必须确保参数类型和指定的格式匹配。可用类型映射确保正确的类型传递到 printf。可传递给 printf 的总的字节数受到 8051 的存储区的限制。SMALL 模式和 COMPACT 模式最多 15 字节，LARGE 模式最多 40 字节。

返回：printf 函数返回实际写到输出流的字符数。

(6)sprintf

原型：int sprintf(char *buffer,const char *fmtstr [,arguments]…);

功能：sprintf 函数格式化一系列的字符串和数值，并保存结果字符串在 buffer fintstr。

参数：参数是一个格式字符串，和 printf 函数指定的要求相同。

返回：sprintf 函数返回实际写到 buffer 的字符数。

(7)vprintf

原型：void vprintf(const char *fmtstr, char *argptr);

功能：vprintf 函数格式化一系列字符串和数字值，并建立一个用 puschar 函数写到输出流的字符串，函数类似于 printf 的副本，但使用参数列表的指针，而不是一个参数列表。本函数是指定执行的，基于 putchar 函数的操作。本函数作为标准库提供，用 8051 的串口写字符。别的 I/O 设备可以定制函数。

参数：fmtstr 参数是一个指向一个格式字符串的指针，和 printf 函数的 fmtstr 参数有相同的形式和功能。argptr 参数指向一系列参数，根据格式中指定的对应格式转换和输出。

返回：vprintf 函数返回实际写到输出流的字符数。

(8)vsprimf

原型：void vsprintf(char *buffer, const char *fmtstr, char *argptr);

功能：vsprintf 函数格式化一系列字符串和数字值，并保存字符串在 Buffer 中。函数类似于 sprintf 的副本，但使用参数列表的指针，而不是一个参数列表。

参数：fmtstr 参数是一个指向一个格式字符串的指针，和 printf 函数的 fmtstr 参数有相同的形式和功能。argptr 参数指向一系列参数，根据格式中指定的对应格式转换和输出。

返回：vsprintf 函数返回实际写到输出流的字符数。

(9)gets

原型：char *gets(char *string, int len);

功能：gets 函数调用 getchar 函数读一行字符到 string。这行包括所有的字符和换行符('\n')。在 string 中换行符被一个 NULL 字符('\n')替代。len 参数指定可读的最多字符数。如果长度超过 len，gets 函数用 NULL 字符终止 string 并返回。本函数指定执行基于_getkey 或 putchar 函数的操作。这些函数，在标准库中提供，用 8051 的串口读写。对别的 I/O 设备可以定制。

参数：string 要读的字符串，len 最多字符数。

返回：gets 函数返回 string

(10)scanf

原型：int scanf(sonst char *fmtstr [,argument]…);

功能：scanf 函数用 getchar 程序读数据。输入的数据保存在由 argument 根据格式字符串 fmtstr 指定的位置。

参数：每个 argument 必须是一个指针，指向一个变量，对应 fmtstr 定义的类型，fmtstr 控制解释输入的数据，fmtstr 参数由一个或单个空白字符、非空白字符和下面定义的格式标识符组成。

- 空白字符，空白(' ')，制表('\t')或换行('\n')，使 scanf 跳过输入流中的空白字符。格式字符串中的单个的空白字符匹配输入流的 0 或多个空白字符。
- 非空白字符，除了百分号('%')，使 scanf 从输入流读但不保存一个匹配字符。如果输入流的下一个字符和指定的非空白字符不匹配，scanf 函数终止。
- 格式标识符以百分号('%')开头，使 scanf 从输入流读字符，并转换字符到指定的类型值。转换后的值保存在参数列表的 argument 中。百分号后面的字符不被认为是一个格式标识符，只作为一个普通字符。例如%%匹配输入流的一个百分号。

格式字符串从左向右读，不是格式标识符的字符必须和输入流的字符匹配。这些字符从输入流读入，但不保存，如果输入流的一个字符和格式字符串冲突，scanf 终止。任何冲突的字符仍保留在输入流中。

在格式字符串中的第一个格式标识符引用 fmtstr 后面的第一个参数，并转化输入字符，用格式标识符保存值。第二个格式标识符访问 fmtstr 后面的第二个参数，等等。如果参数比格式标识符多，多出的参数被忽略。如果没有足够的参数匹配格式标识符，结果是不可预料的。

输入流中的值被输入域调用，用空白字符隔开。在转换输入域时，scanf 遇到一个空白字符就结束一个参数的转换，而且任何当前格式标识符不认识的字符会结束一个域转换。

格式标识符的格式：

%[*][width][{b|h|l}]type

格式标识符中的每个域可以是单个字符或数字，用来指定一个特殊的格式选项。

type 域是单个字符，指定输入字符是否解释为一个字符、字符串或数字。本域可以是下表中的任何值。

字 符	参 数 类 型	输 入 格 式
d	int*	带符号十进制数
l	int*	带符号十进制十六进制或八进制整数
u	unsigned int*	不带符号十进制数
o	unsigned int*	不带符号八进制数
x	unsigned int*	不带符号十六进制数
e	float *	浮点数
f	float *	浮点数
g	float *	浮点数
c	char *	单个字符
s	char *	一个字符串以空白结尾

以星号(*)作为格式标识符的第一个字符，会使输入域被扫描但不保存。星号禁止和格式标识符关联。

width 域是一个非负数，指定从输入流读入的最多字符数。从输入流读入的字符不超过 width，并根据相应的 argument 转换。然而，如果一个空白字符或一个不认识字符先遇到，则读入的字符数小于 width。

可选字符 b，h 和 l 直接放在类型字符前面，分别指定整数类型 d，i，u，o 和 x 的 char，short 或 long 版本。

本函数指定执行基于_getkey 或 putchar 函数的操作。这些函数，作为标准库提供，用 805l 的串口读写。可对别的 I/O 设备定制函数。

可以传递给 scanf 的字节数受 8051 存储区的限制。SMALL 模式或 COMPACT 模式最多为 15 字节。LARGE 模式最多为 40 字节。

返回：scanf 函数返回成功转换的输入域的数目。如果有错误则返回 EOF。

(11)sscanf

原型：int sscanf(char *buffer,const char *fmtstr [,argument]…);

功能：sscanf 函数从 buffer 读字符串。

参数：输入的数据保存在由 argument 根据格式字符串 fmtstr 指定的位置。每个 argument 必须是指向变量的指针，对应定义在 fmtstr 的类型，控制输入数据的解释。fmtstr 参数由一个或多个空白字符、非空白字符和格式标识符组成，如同 scanf 函数所定义。

返回：sscanf 函数返回成功转换的输入域的数目，如果出现错误则返回 EOF。

(12)puts

原型：int puts(const char *string);

功能：puts 函数用 putchar 函数写 string 和换行符\n 到输出流。本函数指定执行基于 putchar 函数的操作。本函数作为标准库提供，写字符到 8051 的串口。用别的 I/O 口可以定制函数。

参数：输出的字符串。

返回：如果出现错误，puts 函数返回 EOF，如果没有则返回 0。

五、字符测试<ctype.h>

(1)isalpha

原型：bit isalpha(char c);

功能：isalpha 函数测试参数 c，确定是否是一个字母('A'～'Z'，'a'～'z')。

返回：如果 c 是一个字母，isalpha 函数返回 1(真)，否则返回 0(假)。

(2)isalnum

原型：bit isalnum(char c);

功能：isalnum 函数测试参数 c，确定是否是一个字母或数字字符('A'～'Z'，'a'～'z'，'0'～'9')。

返回：如果 c 是一个字母或数字字符，isalnum 函数返回 1(真)，否则返回 0(假)。

(3)iscntrl

原型：bit iscntrl (char c);

功能：iscntrl 函数测试参数 c，确定是否是一个控制字符(0x00~0x1F 或 0x7F)。

返回：如果 c 是一个控制字符，iscntrl 函数返回 1(真)，否则返回 0(假)。

(4)isdigit

原型：bit isdigit(char c);

功能：isdigit 函数测试参数 c，确定是否是一个十进制数('0'～'9')。

返回：如果 c 是一个十进制数，isdigit 函数返回 1(真)，否则返回 0(假)。

(5)isgraph

原型：bit isgraph(char c);

功能：isgraph 函数测试参数 c，确定是否是一个可打印字符(0x21～0x7E，不包括空格)。

返回：如果 c 是一个可打印字符，isgraph 函数返回 1(真)，否则返回 0(假)。

(6)isprint

原型：bit isprint(char c);

功能：isprint 函数测试参数 c，确定是否是一个可打印字符(0x20～0x7E)。

返回：如果 c 是一个可打印字符，isprint 函数返回 1(真)，否则返回 0(假)。

(7)ispunct

原型：bit ispunct(char c);

功能：ispunct 函数测试参数 c，确定是否是一个标点符号字符(! , . : ; ? " # $ % & ' ` ( ) < > [ ] { } * + - = / | \ @ ^ _ ~)。

返回：如果 c 是一个标点符号字符，ispunct 函数返回 1(真)，否则返回 0(假)。

(8)islower

原型：bit islower(char c);

功能：islower 函数测试参数 c，确定是否是一个小写字母字符('a'～'z')。

返回：如果 c 是一个小写字母字符，islower 函数返回 1(真)，否则返回 0(假)。

(9)isupper

原型：bit isupper(char c);

功能：isupper 函数测试参数 c，确定是否是一个大写字母字符('A'～'Z')。

返回：如果 c 是一个大写字母字符，isupper 函数返回 1(真)，否则返回 0(假)。

(10)isspace

原型：bit isspace(char c);

功能：isspace 函数测试参数 c，确定是否是一个空白字符(0x09～0x0D 或 0x20)。

返回：如果 c 是一个空白字符，isspace 函数返回 1(真)，否则返回 0(假)。

(11)isxdigit

原型：bit isalnum(char c);

功能：isalnum 函数测试参数 c，确定是否是一个十六进制数('A'～'F'，'a'～'f'，'0'～'9')。

返回：如果 c 是一个十六进制数，isalnum 函数返回 1(真)，否则返回 0(假)。

(12)tolower

原型：char tolower(char c);

功能：tolower 函数转换 c 为一个小写字符。如果 c 不是一个字母，tolower 函数无效。

(13)toupper

原型：char toupper(char c);

功能：toupper 函数转换 c 为一个大写字符。如果 c 表示一个字母，toupper 函数无效。

参数：

返回：toupper 宏返回 c 的大写。

(14)toint

原型：char toint(char c);

功能：toint 函数解释 c 为十六进制值。ASCII 字符'0'～'9'生成值 0～9。ASCII 字符'A'～'F'和'a'～'f'生成值 10～15。如果 c 表示一个十六进制数，函数返回-1。

返回：toint 宏返回 c 的十六进制 ASCII 值。

(15)_tolower

原型：#define _tolower(c) ((c)-'A'+'a')

功能：_tolower 宏是在已知 c 是一个大写字符的情况下可用的 lower 的一个版本。

返回：_tolower 宏返回 c 的小写。

(16)_toupper

原型：#define _toupper(c) ((c)-'a'+'A')

功能：_toupper 宏是在已知 c 是一个小写字符的情况下可用的 toupper 的一个版本。

返回：_toupper 宏返回 c 的大写。

(17)toascii

原型：#define toascii(c) ((c) & 0x7F)

功能：toascii 宏转换 c 为一个 7 位 ASCII 字符。宏只转换变量 c 的低 7 位。

返回：toascii 宏返回 c 的 7 位 ASCII 字符。

六、跳转<setjmp.h>

(1)setjmp

原型：volatile int setjmp(jmp_buf env);

功能：setjmp 函数保存当前 CPU 的状态在 env，该状态可以调用 longjmp 函数来恢复。

参数：当同时使用时，setjmp 和 longjmp 函数提供一种方法实行非局部跳转。setjmp 函数保存当前指令地址和别的 CPU 寄存器。一个 longjmp 的并发调用恢复指令指针和寄存器，在 setjmp 调用后面恢复运行。只有声明了 volatile 属性的局部变量和函数参数被恢复。

返回：当 CPU 的当前状态被复制到 env，setjmp 函数返回一个 0。一个非零值表示执行了 longjmp 函数来返回 setjmp 函数调用。在这种情况下，返回值是传递给 longjmp 函数的值。

(2)longjmp

原型：volatile void longjmp(jmp_bufenv,int retval);

功能：longjmp 函数恢复用 setjmp 函数保存在 env 的状态。retval 参数指定从 setjmp 函数调用返回值。longjmp 和 setjmp 函数可以用来执行非局部跳转，通常用来控制一个错误恢复程序。只有用 volatile 属性声明的局部变量和函数参数被恢复。

七、字符串操作<string.h>

(1)strcat

原型：char *strcat(char *s1,char *s2);

功能：strcat 函数连接或添加 s2 到 s1，并用 NULL 字符终止 s1。

参数：s1 目标字符串，s2 源字符串。

返回：s1。

(2)strncat

原型：char *strncat(char *s1,char *s2,int len);

功能：strncat 函数从 s2 添加最多 len 个字符到 s1，并用 NULL 结束。如果 s2 的长度小于 len，s2 连带 NULL 全部复制。

参数：s1 目标字符串，s2 源字符串，len 连接的最多字符数。

返回：strncat 函数返回 s1。

(3)strcmp

原型：char strcmp(char *s1,char *s2);

功能：strcmp 函数比较字串 s1 和 s2 的内容，并返回一个值表示它们的关系。

返回：若 s1<s2 返回负数；若 s1=s2 返回 0；若 s1>s2 返回正数。

(4)strncmp

原型：char *strncmp(char *s1,char *s2,int len);

功能：strncmp 函数比较 s1 的前 len 字节和 s2，返回一个值表示它们的关系。

参数：s1,s2 为字串，len 为比较的长度。

返回：若 s1<s2 返回负数；若 s1=s2 返回 0；若 s1>s2 返回正数。

(5)strcpy

原型：char *strcpy(char *s1, char *s2);

功能：strcpy 函数复制字符串 s2 到字符串 s1，并用 NULL 字符结束 s1。

参数：s1 目标字符串，s2 源字符串。

返回：字符串 s1。

(6)strncpy

原型：char *strncpy(char *dest, char *s2, int len);

功能：strncpy 函数从字符串 s2 复制最多 len 个字符到字符串 s1。

返回：字符串 s1。

(7)strlen

原型：int strlen(char *s);

功能：strlen 函数计算字符串 s 的字节数，不包括 NULL 结束符。

参数：s 要测试长度的字串。

返回：字符串 s 的长度。

(8)strchr

原型：char *strchr(const char *s, char c);

功能：strchr 函数搜索字符串 s 中第一个出现的 c。s 中的 NULL 字符终止搜索。

参数：s 被搜索的字符串，c 要查找的字符。

返回：字符串 s 中指向 c 的指针，如没有发现则返回一个 NULL 指针。

(9)strops

原型：int strpos(const char *s, char c);

功能：strpos 函数查找字符串 s 中 c 的第一次出现，包括 s 的 NULL 结束符。

参数：s 被搜索的字符串，c 要查找的字符。

返回：s 中和 c 匹配的字符的索引。如没匹配则返回-1。s 中第一个字符的索引是 0。

(10)strrchr

原型：char *strrchr(const char *s, char c);

功能：strrchr 函数查找字符串 s 中 c 的最后一次出现，包括 s 的 NULL 结束符。

参数：s 被搜索的字符串，c 要查找的字符。

返回：strrchr 函数返回 s 中和 c 匹配的字符的指针，如没匹配则返回 NULL。

(11)strrpos

原型：int strrpos(const char *s, char c);

功能：strrpos 函数查找字符串 s 中 c 的最后一次出现，包括 s 的 NULL 结束符。

参数：s 被搜索的字符串，c 要查找的字符。

返回：s 中和 c 匹配的最后字符的索引。如没匹配则返回-1，s 中第一个字符的索引是 0。

(12)strcspn

原型：int strcspn(char *s, char *set);

功能：在字符串 s 中查找字符串 set 中的任何字符。

参数：s 源字串，set 查找的字串。

返回：strcspn 函数返回 s 中和 set 匹配的第一个字符的索引。如果 s 的第一个字符和 set 中的一个字符匹配，返回 0。如果 s 中没有字符匹配，返回字符串的长度。

(13)strpbrk

原型：char *strpbrk(char *s, char *set);

功能：查找字符串 s 中第一个出现的 set 中的任何字符，不包括 NULL 结束符。

参数：s 源字串，set 查找的字串。

返回：s 匹配的字符的指针。如果 s 没有字符和 set 匹配，返回一个 NULL 指针。

(14)strrpbrk

原型：char *strrpbrk(char *s, char *set);

功能：查找字符串 s 中最后一个出现的 set 中的任何字符，不包括 NULL 结束符。

返回：s 最后匹配的字符的指针。如果 s 没有字符和 set 匹配，返回一个 NULL 指针。

(15)strspn

原型：int strspn(char *s, char *set);

功能：查找字符串 s 中 set 没有的字符。

返回：strspn 函数返回 s 第一个和 set 不匹配的字符的索引。如果 s 中的第一个字符和 set 中的字符不匹配，返回 0。如果 s 中的所有字符 set 中都有，返回 string 的长度。

(16)strstr

原型：char *strstr(const char *s, char *sub);

功能：在字符串 s 中搜索子串 sub。

返回：strstr 函数返回子字符串 sub 在字符串 s 中第一次出现的位置的指针。指针指向第一次出现的开头。如果 s 中不存在 sub，则返回一个 NULL 指针。

(17)memcmp

原型：char memcmp(void *buf1,void *buf2,int len);

功能：memcmp 函数比较两个缓冲区 buf1 和 bur2 长度为 len 的字节，并返回一个值。

返回：若返回 0，则 bufl=buf2；若返回负数，则 buf1<but2；若返回正数，则 buf1>bul2。

(18)memcpy

原型：void *memcpy(void *s1,void *s2,int len);

功能：memcpy 函数从字符串 s2 复制 len 字节到字符串 s1。如果存储缓冲区重叠，memcpy 函数不能保证 s2 中的那个字节在被覆盖前复制到 s1。如果缓冲区重叠用 memmove 函数。

参数：s1 目标缓冲区，s2 源缓冲区，len 拷贝的字节数。

返回：s1。

(19)memchr

原型：void *memchr(void *buf,char c,int len);

功能：memchr 函数在 buf 中的前 len 个字节中查找字符 c。

返回：memchr 函数返回字符 c 在 buf 中的指针，如没有则返回一个 NULL 指针。

(20)memccpy

原型：void *memccpy(void *dest,void *src,char c,int len);

功能：memccpy 函数从 src 到 dest 复制 0 或更多的字符，直到字符 c 被复制或 len 字节被复制，哪个条件先遇到就执行哪个条件。

返回：memccpy 函数返回一个指针，指向 dest 最后一个复制的字符的后一个字节。如果最后一个字符是 c，则返回一个 NULL 指针。

(21)memmove

原型：void *memmove(void *dest,void *src,int len);

功能：memmove 函数从 src 复制 len 字节到 dest。如果存储缓冲区重叠，memmove 函数保证 src 中的那个字节在被覆盖前复制到 dest。

参数：dest 目标缓冲区，src 源缓冲区，len 移动的字节数。

返回：memmove 函数返回 dest

(22)memset

原型：void *memset(void *buf,char c,int len);

功能：memset 函数设置 buf 的前 len 字节为 c。

参数：buf 要初始化的缓冲区，c 要设值的值，len 缓冲区长度。

返回：memset 函数返回 dest。

八、可变参数<stdarg.h>

(1)va_arg

原型：type va_arg(argptr, type);

功能：va_arg 宏用来从一个可变长度参数列表索引 argptr 提取并列参数。type 参数指定提取参数的数据类型。本宏对每个参数只能调用一次，且必须根据参数列表中的参数顺序调用。第一次调用 va_arg 返回 va_start 宏中指定的 prevparm 参数后的第一个参数。后来对 va_arg 的调用依次返回余下的参数。

参数：

返回：va_arg 宏返回指定参数类型的值。

(2)va_end

原型：void va_end(argptr);

功能：va_end 宏用来终止可变长度参数列表指针 argptr 的使用，argptr 用 va_start 宏初始化。

参数：

返回：无。

(3)va_start

原型：void va_start(argptr, prevparm);

功能：va_start 宏用在一个可变长度参数列表的函数中时，用 va_arg 和 va_end 宏初始化 argptr。prevparm 参数必须是用三点号(...)指定的可选参数前紧挨的函数参数。此函

数必须在用 va_arg 宏访问前初始化可变长度参数列表指针。

参数：

返回：无。