**C语言库函数总结**

#include<stdio.h>

库宏

Stdin ——标准输入（键盘）

Stdout ——标准输出（屏幕）

Stderr ——标准错误

函数

int scanf(const char \*format, &……)

如果成功，该函数返回成功匹配和赋值的个数。如果到达文件末尾或发生读错误，则返回 EOF。

format 说明符形式为: [ % [\*] [width] [modifiers] type ]

|  |  |
| --- | --- |
| **参数** | **描述** |
| \* | 这是一个可选的星号，表示数据是从流 stream 中读取的，但是可以被忽视，即它不存储在对应的参数中。 |
| width | 这指定了在当前读取操作中读取的最大字符数。 |
| modifiers | 为对应的附加参数所指向的数据指定一个不同于整型（针对 d、i 和 n）、无符号整型（针对 o、u 和 x）或浮点型（针对 e、f 和 g）的大小： h ：短整型（针对 d、i 和 n），或无符号短整型（针对 o、u 和 x） l ：长整型（针对 d、i 和 n），或无符号长整型（针对 o、u 和 x），或双精度型（针对 e、f 和 g） L ：长双精度型（针对 e、f 和 g） |
| type | 一个字符，指定了要被读取的数据类型以及数据读取方式。具体参见下一个表格。 |

**scanf 类型说明符：**

|  |  |  |
| --- | --- | --- |
| **类型** | **合格的输入** | **参数的类型** |
| %a、%A | 读入一个浮点值(仅 C99 有效)。 | float \* |
| %c | 单个字符：读取下一个字符。如果指定了一个不为 1 的宽度 width，函数会读取 width 个字符，并通过参数传递，把它们存储在数组中连续位置。在末尾不会追加空字符。 | char \* |
| %d | 十进制整数：数字前面的 + 或 - 号是可选的。 | int \* |
| %e、%E、%f、%F、%g、%G | 浮点数：包含了一个小数点、一个可选的前置符号 + 或 -、一个可选的后置字符 e 或 E，以及一个十进制数字。两个有效的实例 -732.103 和 7.12e4 | float \* |
| %i | 读入十进制，八进制，十六进制整数 。 | int \* |
| %o | 八进制整数。 | int \* |
| %s | 字符串。这将读取连续字符，直到遇到一个空格字符（空格字符可以是空白、换行和制表符）。 | char \* |
| %u | 无符号的十进制整数。 | unsigned int \* |
| %x、%X | 十六进制整数。 | int \* |
| %p | 读入一个指针 。 |  |
| %[] | 扫描字符集合 。 |  |
| %% | 读 % 符号。 |  |

int sscanf(const char \*str, const char \*format, &……)

如果成功，该函数返回成功匹配和赋值的个数。如果到达文件末尾或发生读错误，则返回 EOF。

**str -- 这是 C 字符串，是函数检索数据的源。**

**format -- 这是 C 字符串，包含了以下各项中的一个或多个：空格字符、非空格字符 和 format 说明符。**

**format 说明符形式为 [ % [\*] [width] [modifiers] type ]，具体讲解如下：**

|  |  |
| --- | --- |
| **参数** | **描述** |
| \* | 这是一个可选的星号，表示数据是从流 stream 中读取的，但是可以被忽视，即它不存储在对应的参数中。 |
| width | 这指定了在当前读取操作中读取的最大字符数。 |
| modifiers | 为对应的附加参数所指向的数据指定一个不同于整型（针对 d、i 和 n）、无符号整型（针对 o、u 和 x）或浮点型（针对 e、f 和 g）的大小： h ：短整型（针对 d、i 和 n），或无符号短整型（针对 o、u 和 x） l ：长整型（针对 d、i 和 n），或无符号长整型（针对 o、u 和 x），或双精度型（针对 e、f 和 g） L ：长双精度型（针对 e、f 和 g） |
| type | 一个字符，指定了要被读取的数据类型以及数据读取方式。具体参见下一个表格。 |
|  |  |

**sscanf 类型说明符：**

|  |  |  |
| --- | --- | --- |
| **类型** | **合格的输入** | **参数的类型** |
| c | 单个字符：读取下一个字符。如果指定了一个不为 1 的宽度 width，函数会读取 width 个字符，并通过参数传递，把它们存储在数组中连续位置。在末尾不会追加空字符。 | char \* |
| d | 十进制整数：数字前面的 + 或 - 号是可选的。 | int \* |
| e,E,f,g,G | 浮点数：包含了一个小数点、一个可选的前置符号 + 或 -、一个可选的后置字符 e 或 E，以及一个十进制数字。两个有效的实例 -732.103 和 7.12e4 | float \* |
| o | 八进制整数。 | int \* |
| s | 字符串。这将读取连续字符，直到遇到一个空格字符（空格字符可以是空白、换行和制表符）。 | char \* |
| u | 无符号的十进制整数。 | unsigned int \* |
| x,X | 十六进制整数。 | int \* |

int printf(const char \*format, ...)

* format 标签属性是 **%[flags][width][.precision][length]specifier**，具体讲解如下：

|  |  |
| --- | --- |
| **格式字符** | **意义** |
| a, A | 以十六进制形式输出浮点数(C99 新增)。 |
|  |  |
| d | 以十进制形式输出带符号整数(正数不输出符号) |
| o | 以八进制形式输出无符号整数(不输出前缀0) |
| x,X | 以十六进制形式输出无符号整数(不输出前缀Ox) |
| u | 以十进制形式输出无符号整数 |
| f | 以小数形式输出单、双精度实数 |
| e,E | 以指数形式输出单、双精度实数 |
| g,G | 以%f或%e中较短的输出宽度输出单、双精度实数 |
| c | 输出单个字符 |
| s | 输出字符串 |
| p | 输出指针地址 |
| lu | 32位无符号整数 |
| llu | 64位无符号整数 |

|  |  |
| --- | --- |
| **flags（标识）** | **描述** |
| - | 在给定的字段宽度内左对齐，默认是右对齐（参见 width 子说明符）。 |
| + | 强制在结果之前显示加号或减号（+ 或 -），即正数前面会显示 + 号。默认情况下，只有负数前面会显示一个 - 号。 |
| 空格 | 如果没有写入任何符号，则在该值前面插入一个空格。 |
| # | 与 o、x 或 X 说明符一起使用时，非零值前面会分别显示 0、0x 或 0X。 与 e、E 和 f 一起使用时，会强制输出包含一个小数点，即使后边没有数字时也会显示小数点。默认情况下，如果后边没有数字时候，不会显示显示小数点。 与 g 或 G 一起使用时，结果与使用 e 或 E 时相同，但是尾部的零不会被移除。 |
| 0 | 在指定填充 padding 的数字左边放置零（0），而不是空格（参见 width 子说明符）。 |

|  |  |
| --- | --- |
| **width（宽度）** | **描述** |
| (number) | 要输出的字符的最小数目。如果输出的值短于该数，结果会用空格填充。如果输出的值长于该数，结果不会被截断。 |
| \* | 宽度在 format 字符串中未指定，但是会作为附加整数值参数放置于要被格式化的参数之前。 |

|  |  |
| --- | --- |
| **.precision（精度）** | **描述** |
| .number | 对于整数说明符（d、i、o、u、x、X）：precision 指定了要写入的数字的最小位数。如果写入的值短于该数，结果会用前导零来填充。如果写入的值长于该数，结果不会被截断。精度为 0 意味着不写入任何字符。 对于 e、E 和 f 说明符：要在小数点后输出的小数位数。 对于 g 和 G 说明符：要输出的最大有效位数。 对于 s: 要输出的最大字符数。默认情况下，所有字符都会被输出，直到遇到末尾的空字符。 对于 c 类型：没有任何影响。 当未指定任何精度时，默认为 1。如果指定时不带有一个显式值，则假定为 0。 |
| .\* | 精度在 format 字符串中未指定，但是会作为附加整数值参数放置于要被格式化的参数之前。 |

|  |  |
| --- | --- |
| **length（长度）** | **描述** |
| h | 参数被解释为短整型或无符号短整型（仅适用于整数说明符：i、d、o、u、x 和 X）。 |
| l | 参数被解释为长整型或无符号长整型，适用于整数说明符（i、d、o、u、x 和 X）及说明符 c（表示一个宽字符）和 s（表示宽字符字符串）。 |
| L | 参数被解释为长双精度型（仅适用于浮点数说明符：e、E、f、g 和 G）。 |

int getchar(void)

该函数以无符号 char 强制转换为 int 的形式返回读取的字符，如果到达文件末尾或发生读错误，则返回 EOF。

int getc(FILE \*stream)

该函数以无符号 char 强制转换为 int 的形式返回读取的字符，如果到达文件末尾或发生读错误，则返回 EOF。

int fgetc(FILE \*stream)

该函数以无符号 char 强制转换为 int 的形式返回读取的字符，如果到达文件末尾或发生读错误，则返回 EOF。

char \*gets(char \*str)

如果成功，该函数返回 str。如果发生错误或者到达文件末尾时还未读取任何字符，则返回 NULL。

当读取到换行符时，读取并丢弃。

char \*fgets(char \*str, int n, FILE \*stream)

如果成功，该函数返回相同的 str 参数。如果到达文件末尾或者没有读取到任何字符，str 的内容保持不变，并返回一个空指针。如果发生错误，返回一个空指针。

当读取到换行符时，读取并丢弃。

str -- 这是指向一个字符数组的指针，该数组存储了要读取的字符串。

n -- 这是要读取的最大字符数（包括最后的空字符）。通常是使用以 str 传递的数组长度。

stream -- 这是指向 FILE 对象的指针，该 FILE 对象标识了要从中读取字符的流。

int snprintf ( char \* str, size\_t size, const char \* format, ... );

返回值是输出到 str 缓冲区中的字符数，不包括字符串结尾的空字符 \0。如果 snprintf() 输出的字符数超过了 size 参数指定的缓冲区大小，则输出的结果会被截断，只有 size - 1 个字符被写入缓冲区，最后一个字符为字符串结尾的空字符 \0。

需要注意的是，snprintf() 函数返回的字符数并不包括字符串结尾的空字符 \0。

str -- 目标字符串，用于存储格式化后的字符串的字符数组的指针。

size -- 字符数组的大小。

format -- 格式化字符串。

... -- 可变参数，可变数量的参数根据 format 中的格式化指令进行格式化。

#include<math.h>

函数

double exp(double x)

该函数返回 e 的 x 次幂。

double ldexp(double x, int exponent)

该函数返回 x \* (2^exp)。

double log10(double x)

返回 x 的常用对数（基数为 10 的对数），x 的值大于 0。

double log(double x)

返回 x 的自然对数（基数为 e 的对数）。

double fmod(double x, double y)

该函数返回 x/y 的余数。

double pow(double x, double y)

该函数返回 x 的 y 次幂的结果。

double sqrt(double x)

该函数返回 x 的平方根。

double ceil(double x)

该函数返回不小于 x 的最小整数值。

double floor(double x)

该函数返回不大于 x 的最大整数值。

double fabs(double x);

float fabsf(float x);

long double fabsl(long double x);

如果 x 是正数，则返回 x，如果 x 是负数，则返回它的相反数，即 -x。如果 x 是 0，则返回 0。

#include<string.h>

函数

void \*memchr(const void \*str, int c, size\_t n)

在参数 str 所指向的字符串的前 n 个字节中搜索第一次出现字符 c的位置。

该函数返回一个指向匹配字节的指针，如果在给定的内存区域未出现字符，则返回 NULL。

str -- 指向要执行搜索的内存块。

c -- 以 int 形式传递的值，但是函数在每次字节搜索时是使用该值的无符号字符形式。

n -- 要被分析的字节数。

int memcmp(const void \*str1, const void \*str2, size\_t n)

如果返回值 < 0，则表示 str1 小于 str2。

如果返回值 > 0，则表示 str1 大于 str2。

如果返回值 = 0，则表示 str1 等于 str2。

str1 -- 指向内存块的指针。

str2 -- 指向内存块的指针。

n -- 要被比较的字节数

void \*memcpy(void \*str1, const void \*str2, size\_t n)

从存储区 str2 复制 n 个字节到存储区 str1。

该函数返回一个指向目标存储区 str1 的指针。

str1 -- 指向用于存储复制内容的目标数组，类型强制转换为 void\* 指针。

str2 -- 指向要复制的数据源，类型强制转换为 void\* 指针。

n -- 要被复制的字节数。

void \*memmove(void \*str1, const void \*str2, size\_t n)

从 str2 复制 n 个字符到 str1，但是在重叠内存块这方面，memmove是比 memcpy更安全的方法。

该函数返回一个指向目标存储区 str1 的指针。

str1 -- 指向用于存储复制内容的目标数组，类型强制转换为 void\* 指针。

str2 -- 指向要复制的数据源，类型强制转换为 void\* 指针。

n -- 要被复制的字节数。

void \*memset(void \*str, int c, size\_t n)

复制字符 c到参数 str 所指向的字符串的前 n 个字符。

该值返回一个指向存储区 str 的指针。

str -- 指向要填充的内存块。

c -- 要被设置的值。该值以 int 形式传递，但是函数在填充内存块时是使用该值的无符号字符形式。

n -- 要被设置为该值的字符数。

char \*strcat(char \*dest, const char \*src)

该函数返回一个指向最终的目标字符串 dest 的指针。

dest -- 指向目标数组，该数组包含了一个 C 字符串，且足够容纳追加后的字符串。

src -- 指向要追加的字符串，该字符串不会覆盖目标字符串。

char \*strncat(char \*dest, const char \*src, size\_t n)

把 src 所指向的字符串追加到 dest 所指向的字符串的结尾，直到 n 字符长度为止。

该函数返回一个指向最终的目标字符串 dest 的指针。

dest -- 指向目标数组，该数组包含了一个字符串，且足够容纳追加后的字符串，包括额外的空字符。

src -- 要追加的字符串。

n -- 要追加的最大字符数。

char \*strchr(const char \*str, int c)

查找字符串中的一个字符，并返回该字符在字符串中第一次出现的位置。

找到字符 c，则函数返回指向该字符的指针，如果未找到该字符则返回 NULL。

str -- 要查找的字符串。

c -- 要查找的字符。

char \*strrchr(const char \*str, int c)

参数 str 所指向的字符串中搜索最后一次出现字符 c的位置。

如果找到字符，它将返回一个指向该字符的指针，否则返回 NULL。

str -- C 字符串。

c -- 要搜索的字符，通常以整数形式传递（ASCII 值），但是最终会转换回 char 形式。

char \*strstr(const char \*haystack, const char \*needle)

字符串 haystack 中查找第一次出现字符串 needle 的位置，不包含终止符 '\0'。

该函数返回在 haystack 中第一次出现 needle 字符串的位置，如果未找到则返回 null。

haystack -- 要被检索的 C 字符串。

needle -- 在 haystack 字符串内要搜索的小字符串。

char \*strpbrk(const char \*str1, const char \*str2)

检索字符串 str1 中第一个匹配字符串 str2 中字符的字符

该函数返回 str1 中第一个匹配字符串 str2 中字符的字符，如果未找到字符则返回 NULL。

str1 -- 要被检索的 C 字符串。

str2 -- 该字符串包含了要在 str1 中进行匹配的字符列表。

int strcmp(const char \*str1, const char \*str2)

如果返回值小于 0，则表示 str1 小于 str2。

如果返回值大于 0，则表示 str1 大于 str2。

如果返回值等于 0，则表示 str1 等于 str2。

str1 -- 要进行比较的第一个字符串。

str2 -- 要进行比较的第二个字符串。

int strncmp(const char \*str1, const char \*str2, size\_t n)

如果返回值 < 0，则表示 str1 小于 str2。

如果返回值 > 0，则表示 str1 大于 str2。

如果返回值 = 0，则表示 str1 等于 str2。

str1 -- 要进行比较的第一个字符串。

str2 -- 要进行比较的第二个字符串。

n -- 要比较的最大字符数。

int strcoll(const char \*str1, const char \*str2)

结果取决于 LC\_COLLATE 的位置设置。

如果返回值 < 0，则表示 str1 小于 str2。

如果返回值 > 0，则表示 str2 小于 str1。

如果返回值 = 0，则表示 str1 等于 str2。

str1 -- 要进行比较的第一个字符串。

str2 -- 要进行比较的第二个字符串。

char \*strcpy(char \*dest, const char \*src)

该函数返回一个指向最终的目标字符串 dest 的指针。

dest -- 指向用于存储复制内容的目标数组。

src -- 要复制的字符串。

char \*strncpy(char \*dest, const char \*src, size\_t n)

该函数返回最终复制的字符串。strncpy 不自动加上终止符的，需要手动添加。

dest -- 指向用于存储复制内容的目标数组。

src -- 要复制的字符串。

n -- 要从源中复制的字符数。

size\_t strcspn(const char \*str1, const char \*str2)

该函数返回 str1 开头连续都不含字符串 str2 中字符的字符数。

str1 -- 要被检索的 C 字符串。

str2 -- 该字符串包含了要在 str1 中进行匹配的字符列表。

size\_t strspn(const char \*str1, const char \*str2)

该函数返回 str1 中第一个不在字符串 str2 中出现的字符下标。

str1 -- 要被检索的 C 字符串。

str2 -- 该字符串包含了要在 str1 中进行匹配的字符列表。

size\_t strlen(const char \*str)

该函数返回字符串的长度。

char \*strtok(char \*str, const char \*delim)

分解字符串 str 为一组字符串，delim 为分隔符。原字符串的改动是切分符原位置均更改为 '\0'。

该函数返回被分解的第一个子字符串，如果没有可检索的字符串，则返回一个空指针。

str -- 要被分解成一组小字符串的字符串。

delim -- 包含分隔符的 C 字符串。

size\_t strxfrm(char \*dest, const char \*src, size\_t n)

根据程序当前的区域选项中的 LC\_COLLATE 来转换字符串 src 的前 n 个字符，并把它们放置在字符串 dest 中。

该函数返回被转换字符串的长度，不包括空结束字符。

dest -- 指向存储内容的目标数组的指针，如果参数 n 为 0，则它是一个空指针。

src -- 要被转换为当前区域设置的 C 字符串。

n -- 被复制到 str1 的最大字符数。

#include<ctype.h>

函数

int isalnum(int c);

如果 c 是一个数字或一个字母，则该函数返回非零值，否则返回 0。

int isalpha(int c);

如果 c 是一个字母，则该函数返回非零值，否则返回 0。

int iscntrl(int c);

如果 c 是一个控制字符，则该函数返回非零值，否则返回 0。

int isdigit(int c);

如果 c 是一个数字，则该函数返回非零值，否则返回 0。

int isdigit(int c);

如果 c 是一个数字，则该函数返回非零值，否则返回 0。

int isgraph(int c);

如果 c 有图形表示法，则该函数返回非零值，否则返回 0。

int islower(int c);

如果 c 是一个小写字母，则该函数返回非零值（true），否则返回 0（false）。

int isupper(int c);

如果 c 是一个大写字母，则该函数返回非零值（true），否则返回 0（false）。

int isprint(int c);

如果 c 是一个可打印的字符，则该函数返回非零值（true），否则返回 0（false）。

int ispunct(int c);

如果 c 是一个标点符号字符，则该函数返回非零值（true），否则返回 0（false）

int isspace(int c);

如果 c 是一个空白字符，则该函数返回非零值（true），否则返回 0（false）。

标准的空白字符包括：

' ' (0x20)    space (SPC) 空格符

'\t'    (0x09)    horizontal tab (TAB) 水平制表符

'\n'    (0x0a)    newline (LF) 换行符

'\v'    (0x0b)    vertical tab (VT) 垂直制表符

'\f'    (0x0c)    feed (FF) 换页符

'\r'    (0x0d)    carriage return (CR) 回车符

int tolower(int c);

如果 c 有相对应的小写字母，则该函数返回 c 的小写字母，否则 c 保持不变。返回值是一个可被隐式转换为 char 类型的 int 值。

int toupper(int c);

如果 c 有相对应的大写字母，则该函数返回 c 的大写字母，否则 c 保持不变。返回值是一个可被隐式转换为 char 类型的 int 值。

***#include<stdlib.h>***

函数

double atof(const char \*str)

函数返回转换后的双精度浮点数，如果没有执行有效的转换，则返回零（0.0）。

int atoi(const char \*str)

该函数返回转换后的长整数，如果没有执行有效的转换，则返回零。

long int atol(const char \*str)

该函数返回转换后的长整数，如果没有执行有效的转换，则返回零。

long long int atoll (const char \* str)

该函数返回其long long整数值，如果没有执行有效的转换，则返回零。

double strtod(const char \*str, char \*\*endptr)

该函数返回转换后的双精度浮点数，如果没有执行有效的转换，则返回零（0.0）。

long int strtol(const char \*str, char \*\*endptr, int base)

函数返回被转换的long型整数值。如果输入字符串不符合数字格式，将返回 0。

如果转换结果超出了long整数的表示范围，那么将产生溢出。

# 对于str指向的字符串，其开头和结尾处的空格被忽视，字符串中间的空格被视为非法字符。

str -- 要转换为长整数的字符串。

endptr -- 对类型为 char\* 的对象的引用，其值由函数设置为 str 中数值后的下一个字符。

base -- 基数，必须介于 2 和 36（包含）之间，或者是特殊值 0。如果 base 为 0，则会根据字符串的前缀来判断进制：如果字符串以 '0x' 或 '0X' 开头，则将其视为十六进制；如果字符串以 '0' 开头，则将其视为八进制；否则将其视为十进制。

unsigned long int strtoul(const char \*str, char \*\*endptr, int base)

该函数返回转换后的长整数，如果没有执行有效的转换，则返回一个零值。

long long int strtoll(const char \*str, char\*\* endptr, int base)

函数返回被转换的long long型整数值。如果输入字符串不符合数字格式，将返回 0。

unsigned long long strtoull(const char \*str, char\*\* endptr, int base);

函数返回被转换的unsigned long long型整数值。如果输入字符串不符合数字格式，将返回 0。

char \*itoa(int value, char \*string, int radix)

将整型值转换为字符串。

value：要转换的数据。

string：目标字符串的地址。

radix：转换后的进制数，可以是10进制、16进制等，范围必须在 2-36。

# itoa并不是一个标准的C函数，它是Windows特有的

char \*ltoa(long value,char \*string,int radix)

将长整型值转换为字符串。

# ltoa并不是一个标准的C函数，它是Windows特有的

char \*ultoa(long value,char \*string,int radix)

将无符号长整型值转换为字符串。

# ultoa并不是一个标准的C函数，它是Windows特有的

gcvt()

将浮点型数转换为字符串，取四舍五入。

ecvt()

将双精度浮点型值转换为字符串，转换结果中不包含十进制小数点。

fcvt()

指定位数为转换精度，其余同ecvt()。

void \*calloc(size\_t nitems, size\_t size)

calloc() 函数将分配的内存全部初始化为零（初始化）。

该函数返回一个指针，指向已分配的内存。如果请求失败，则返回 NULL。

nitems -- 要被分配的元素个数。

size -- 元素的大小。

Malloc(size\_t size)

malloc和calloc 之间的不同点是，malloc不会设置内存为零（不初始化）。

该函数返回一个指针，指向已分配的内存。如果请求失败，则返回 NULL。

void \*realloc(void \*ptr, size\_t size)

该函数返回一个指针 ，指向重新分配大小的内存。如果请求失败，则返回 NULL。

realloc 分配内存时，如果为空指针，则会分配一个新的内存块，且函数返回一个指向它的指针。

ptr -- 指针指向一个要重新分配内存的内存块，该内存块之前是通过调用 malloc、calloc 或 size -- 内存块的新的大小，以字节为单位。如果大小为 0，且 ptr 指向一个已存在的内存块，则 ptr 所指向的内存块会被释放，并返回一个空指针。

void free(void \*ptr)

该函数不返回任何值。

ptr -- 指针指向一个要释放内存的内存块，该内存块之前是通过调用 malloc、calloc 或 realloc 进行分配内存的。如果传递的参数是一个空指针，则不会执行任何动作。

int abs(int x)

返回整数 x 的绝对值。

long int labs(long int x)

返回 x 的绝对值

div\_t div(int numer, int denom)

需先定义结构体div\_t变量（for example: div\_t output ; ）

该函数返回定义在 <cstdlib> 中结构div\_t的值，该结构有两个int成员quot（商）、rem（余数）

numer -- 分子。

denom -- 分母。

ldiv\_t ldiv(long int numer, long int denom)

需先定义结构体ldiv\_t变量（for example: div\_t output ; ）

该函数返回定义在 <cstdlib> 中结构div\_t的值，该结构有两个long int成员quot（商）、rem（余数）

Tip:

ldiv()函数的用途是在需要同时计算商和余数时，避免重复计算的情况，因为在计算商的同时，也可以得到余数。

此外，当被除数或除数是负数时，ldiv()的结果将符合向零舍入的规则。

void \*bsearch

(const void \*key, const void \*base, size\_t nitems, size\_t size, int (\*compar)(const void \*, const void \*))

如果查找成功，该函数返回一个指向数组中匹配元素的指针，否则返回空指针。

key -- 指向要查找的元素的指针，类型转换为 void\*。

base -- 指向进行查找的数组的第一个对象的指针，类型转换为 void\*。

nitems -- base 所指向的数组中元素的个数。

size -- 数组中每个元素的大小，以字节为单位。

compar -- 用来比较两个元素的函数。

void qsort(void \*base, size\_t nitems, size\_t size, int (\*compar)(const void \*, const void\*))

该函数不返回任何值。

比较函数有两个参数，它们是通用指针，分别指向两个需要比较的数组元素。

比较函数返回一个整数。（升降序样例示范）

// 如果第一个参数指向的数组元素大于第二个参数指向的数组元素，返回正整数 ；

// 如果第一个参数指向的数组元素等于第二个参数指向的数组元素，返回0 ；

// 如果第一个参数指向的数组元素小于第二个参数指向的数组元素，返回负整数 。

//升序（顺序）： int comp(const void \*p,const void \*q)

// {

// return (int)( \* (int \*) p - \* (int \*) q ) ;

// }

//降序（逆序）： int comp(const void \*p,const void \*q)

// {

// return (int)( \* (int \*) q - \* (int \*) p ) ;

// }

比较函数样例

整型数组

int comp\_int(const void\* elem1, const void\* elem2)

{

return \*(int\*)elem1 - \*(int\*)elem2;

}

字符数组

int comp\_char(const void\* elem1, const void\* elem2)

{

return \*(char\*)elem1 - \*(char\*)elem2; //字符计算是计算ASCLL值（int）

}

浮点数组

int comp\_double(const void\* elem1, const void\* elem2)

{

return (int)(\*(double\*)elem1 - \*(double\*)elem2);

}

字符串（首字母）

int comp\_char(const void\* elem1, const void\* elem2)

{

return \*(char\*)elem1 - \*(char\*)elem2;//字符计算是计算ASCLL值

}

字符串（长度）

int comp\_string(const void\* elem1, const void\* elem2)

{

if ( strlen( \* (char\*)elem1) == strlen( \*(char\*)elem2) )

{

return 0 ;

}

else

{

if ( strlen( \* (char\*)elem1) > strlen( \*(char\*)elem2) )

{

return 1 ;

}

else

{

return -1 ;

}

}

}

字符串（常规顺序）

int comp\_string(const void\* elem1, const void\* elem2)

{

return strcmp(\*(char\*)elem1, \*(char\*)elem2);//strcmp比较两个字符串大小

}

结构体

system函数

<http://t.csdnimg.cn/a3TXk>

Windows CMD常用命令

<https://zhuanlan.zhihu.com/p/468515490?utm_psn=1727414850894798848>

c语言 conio.h头文件函数介绍

<http://t.csdnimg.cn/kbv9w>

C语言调整控制台

<https://blog.csdn.net/cjz2005/article/details/104358000?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522170450697116800213040098%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=170450697116800213040098&biz_id=0&u>

C语言控制台光标控制

<https://blog.csdn.net/liujian619/article/details/14055431?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522170485149016800225538171%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=170485149016800225538171&biz_id=0&utm_mediu>

C语言API函数

<https://blog.csdn.net/u010258235/article/details/45666851>

Windows API函数整理

[https://blog.csdn.net/H888001/article/details/](https://blog.csdn.net/H888001/article/details/100546290?ops_request_misc=&request_id=&biz_id=102&utm_term=c%E8%AF%AD%E8%A8%80windows%20API%E5%87%BD%E6%95%B0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-2-100546290.142^v99^)

C语言键盘操作函数

[https://blog.csdn.net/wuwuku123/article/](https://blog.csdn.net/wuwuku123/article/details/103520270?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522170454735916800186545170%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=170454735916800186545170&biz_id=0)