**C语言的常用的标准头文件有 ：**

**<ctype.h>** **<time.h>** **<stdio.h>**

**<stdlib.h>** **<math.h>** **<string.h>**

**一. <ctype.h>**

|  |  |
| --- | --- |
| 函数原型 | 功能 |
| int iscntrl(int c) | 判断字符c是否为控制字符。 |
| int isalnum(int c) | 判断字符c是否为字母或数字 |
| int isalpha(int c) | 判断字符c是否为英文字母 |
| int isascii(int c) | 判断字符c是否为ascii码 |
| int isblank(int c) | 判断字符c是否为TAB或空格 |
| int isdigit(int c) | 判断字符c是否为数字 |
| int isgraph(int c) | 判断字符c是否为除空格外的可打印字符 |
| int islower(int c) | 判断字符c是否为小写英文字母 |
| int isprint(int c) | 判断字符c是否为可打印字符（含空格） |
| int ispunct(int c) | 判断字符c是否为标点符号 |
| int isspace(int c) | 判断字符c是否为空白符 |
| int isupper(int c) | 判断字符c是否为大写英文字母 |
| int isxdigit(int c) | 判断字符c是否为十六进制数字 |
| int toascii(int c) | 将字符c转换为ascii码 |
| int tolower(int c) | 将字符c转换为小写英文字母 |
| int toupper(int c); | 将字符c转换为大写英文字母 |

**二. <math.h>**

|  |  |
| --- | --- |
| 函数原型 | 功能 |
| float fabs(float x) | 求浮点数x的绝对值 |
| int abs(int x) | 求整数x的绝对值 |
| float acos(float x) | 求x（弧度表示）的反余弦值 |
| float asin(float x) | 求x（弧度表示）的反正弦值 |
| float atan(float x) | 求x（弧度表示）的反正切值 |
| float atan2(float y, float x) | 求y/x（弧度表示）的反正切值 |
| float ceil(float x) | 求不小于x的最小整数 |
| float cos(float x) | 求x（弧度表示）的余弦值 |
| float cosh(float x) | 求x的双曲余弦值 |
| float exp(float x) | 求e的x次幂 |
| float floor(float x) | 求不大于x的最大整数 |
| float fmod(float x, float y) | 计算x/y的余数 |
| float frexp(float x, int \*exp) | 把浮点数x分解成尾数和指数 |
| float ldexp(float x, int exp) | 返回x\*2^exp的值 |
| float modf(float num, float \*i) | 将浮点数num分解成整数部分和小数部分 |
| float hypot(float x, float y) | 对于给定的直角三角形的两个直角边，求其斜边的长度 |
| float log(float x) | 计算x的自然对数 |
| float log10(float x) | 计算x的常用对数 |
| float pow(float x, float y) | 计算x的y次幂 |
| float pow10(float x) | 计算10的x次幂 |
| float sin(float x) | 计算x（弧度表示）的正弦值 |
| float sinh(float x) | 计算x（弧度表示）的双曲正弦值 |
| float sqrt(float x) | 计算x的平方根 |
| float tan(float x); | 计算x（弧度表示）的正切值 |
| float tanh(float x) | 求x的双曲正切值 |

**三. <stdio.h>**

|  |  |
| --- | --- |
| 函数原型 | 功能 |
| int printf(char \*format...) | 产生格式化输出的函数 |
| int getchar(void) | 从键盘上读取一个键，并返回该键的键值 |
| int putchar(char c) | 在屏幕上显示字符c |
| FILE \*fopen(char \*filename, char \*type) | 打开一个文件 |
| FILE \*freopen(char \*filename, char \*type,FILE \*fp) | 打开一个文件，并将该文件关联到fp指定的流 |
| int fflush(FILE \*stream) | 清除一个流 |
| int fclose(FILE \*stream) | 关闭一个文件 |
| int remove(char \*filename) | 删除一个文件 |
| int rename(char \*oldname, char \*newname) | 重命名文件 |
| FILE \*tmpfile(void) | 以二进制方式打开暂存文件 |
| char \*tmpnam(char \*sptr) | 创建一个唯一的文件名 |
| int setvbuf(FILE \*stream, char \*buf, int type, unsigned size) | 把缓冲区与流相关 |
| int fprintf(FILE \*stream, char \*format[, argument,...]) | 传送格式化输出到一个流中 |
| int scanf(char \*format[,argument,...]) | 执行格式化输入 |
| int fscanf(FILE \*stream, char \*format[,argument...]) | 从一个流中执行格式化输入 |
| int fgetc(FILE \*stream) | 从流中读取字符 |
| char \*fgets(char \*string, int n, FILE \*stream) | 从流中读取一字符串 |
| int fputc(int ch, FILE \*stream) | 送一个字符到一个流中 |
| int fputs(char \*string, FILE \*stream) | 送一个字符到一个流中 |
| int getc(FILE \*stream) | 从流中取字符 |
| int getchar(void) | 从 stdin 流中读字符 |
| char \*gets(char \*string) | 从流中取一字符串 |
| int putchar(int ch) | 在 stdout 上输出字符 |
| int puts(char \*string) | 送一字符串到流中 |
| int ungetc(char c, FILE \*stream) | 把一个字符退回到输入流中 |
| int fread(void \*ptr, int size, int nitems, FILE \*stream) | 从一个流中读数据 |
| int fwrite(void \*ptr, int size, int nitems, FILE \*stream) | 写内容到流中 int fseek |
| (FILE \*stream, long offset, int fromwhere) | 重定位流上的文件指针 |
| long ftell(FILE \*stream) | 返回当前文件指针 |
| int rewind(FILE \*stream) | 将文件指针重新指向一个流的开头 |
| int fgetpos(FILE \*stream) | 取得当前文件的句柄 |
| int fsetpos(FILE \*stream, const fpos\_t \*pos) | 定位流上的文件指针 |
| void clearerr(FILE \*stream) | 复位错误标志 |
| int feof(FILE \*stream) | 检测流上的文件结束符 |
| int ferror(FILE \*stream) | 检测流上的错误 |
| void perror(char \*string) | 系统错误信息 |

**四. <stdlib.h>**

|  |  |
| --- | --- |
| 函数原型 | 功能 |
| char \*itoa (int value, char \*str, int base); value是原数字，str是要保存进的字符串指针，base是指定的数字进制。 | 把整数i转换成字符串 |
| void exit(int retval) | 结束程序 |
| double atof(const char \*s) | 将字符串s转换为double类型 |
| int atoi(const char \*s) | 将字符串s转换为int类型 |
| long atol(const char \*s) | 将字符串s转换为long类型 |
| double strtod (const char\*s,char \*\*endp) | 将字符串s前缀转换为double型 |
| long strtol(const char\*s,char \*\*endp,int base) | 将字符串s前缀转换为long型 |
| unsinged long strtol(const char\*s,char \*\*endp,int base) | 将字符串s前缀转换为 unsinged long型 |
| int rand(void) | 产生一个0~RAND\_MAX之间的伪随机数 |
| void srand(unsigned int seed) | 初始化随机数发生器 |
| void \*calloc(size\_t nelem, size\_t elsize) | 分配主存储器 |
| void \*malloc(unsigned size) | 内存分配函数 |
| void \*realloc(void \*ptr, unsigned newsize) | 重新分配主存 |
| void free(void \*ptr) | 释放已分配的块 |
| void abort(void) | 异常终止一个进程 |
| void exit(int status) | 终止应用程序 |
| int atexit(atexit\_t func) | 注册终止函数 |
| char \*getenv(char \*envvar) | 从环境中取字符串 |
| void \*bsearch(const void \*key, const void \*base, size\_t \*nelem, size\_t width, int(\*fcmp)(const void \*, const \*)) | 二分法搜索函数 |
| void qsort(void \*base, int nelem, int width, int (\*fcmp)()) | 使用快速排序例程进行排序 |
| int abs(int i) | 求整数的绝对值 |
| long labs(long n) | 取长整型绝对值 |
| div\_t div(int number, int denom) | 将两个整数相除 , 返回商和余数 |
| ldiv\_t ldiv(long lnumer, long ldenom) | 两个长整型数相除 , 返回商和余数 |

**五. <time.h>**

|  |  |
| --- | --- |
| 函数原型 | 功能 |
| clock\_t clock(void) | 确定处理器时间函数 |
| time\_t time(time\_t \*tp) | 返回当前日历时间 |
| double difftime(time\_t time2, time\_t time1) | 计算两个时刻之间的时间差 |
| time\_t mktime(struct tm \*tp) | 将分段时间值转换为日历时间值 |
| char \*asctime(const struct tm \*tblock) | 转换日期和时间为ASCII码 |
| char \*ctime(const time\_t \*time) | 把日期和时间转换为字符串 |
| struct tm \*gmtime(const time\_t \*timer) | 把日期和时间转换为格林尼治标准时间 |
| struct tm \*localtime(const time\_t \*timer) | 把日期和时间转变为结构 |
| size\_t strftime(char \*s,size\_t smax,const char \*fmt, const struct tm \*tp) | 根据 fmt 的格式 要求将 \*tp中的日期与时间转换为指定格式 |

**六. <string.h>**

|  |  |
| --- | --- |
| 函数原型 | 功能 |
| int bcmp(const void \*s1, const void \*s2, int n) | 比较字符串s1和s2的前n个字节是否相等 |
| void bcopy(const void \*src, void \*dest, int n) | 将字符串src的前n个字节复制到dest中 |
| void bzero(void \*s, int n) | 置字节字符串s的前n个字节为零 |
| void \*memccpy(void \*dest, void \*src, unsigned char ch, unsigned int count) | 由src所指内存区域复制不多于count个字节到dest所指内存区域，如果遇到字符ch则停止复制 |
| void \*memcpy(void \*dest, void \*src, unsigned int count) | 由src所指内存区域复制count个字节到dest所指内存区域 |
| void \*memchr(void \*buf, char ch, unsigned count) | 从buf所指内存区域的前count个字节查找字符ch |
| int memcmp(void \*buf1, void \*buf2, unsigned int count) | 比较内存区域buf1和buf2的前count个字节 |
| int memicmp(void \*buf1, void \*buf2, unsigned int count) | 比较内存区域buf1和buf2的前count个字节但不区分字母的大小写 |
| void \*memmove(void \*dest, const void \*src, unsigned int count) | 由src所指内存区域复制count个字节到dest所指内存区域 |
| void \*memset(void \*buffer, int c, int count) | 把buffer所指内存区域的前count个字节设置成字符c |
| void setmem(void \*buf, unsigned int count, char ch) | 把buf所指内存区域前count个字节设置成字符ch |
| void movmem(void \*src, void \*dest, unsigned int count) | 由src所指内存区域复制count个字节到dest所指内存区域 |
| char \*stpcpy(char \*dest,char \*src) | 把src所指由NULL结束的字符串复制到dest所指的数组中 |
| char \*strcpy(char \*dest,char \*src) | 把src所指由NULL结束的字符串复制到dest所指的数组中 |
| char \*strcat(char \*dest,char \*src) | 把src所指字符串添加到dest结尾处(覆盖dest结尾处的’\0’)并添加’\0’ |
| char \*strchr(char \*s,char c) | 查找字符串s中首次出现字符c的位置 |
| int strcmp(char \*s1,char \* s2) | 比较字符串s1和s2 |
| int strcmp(char \*s1,char \* s2) | 比较字符串s1和s2 |
| int stricmp(char \*s1,char \* s2) | 比较字符串s1和s2，但不区分字母的大小写 |
| int strcspn(char \*s1,char \*s2) | 在字符串s1中搜寻s2中所出现的字符 |
| char \*strdup(char \*s) | 复制字符串s |
| int strlen(char \*s) | 计算字符串s的长度 |
| char \*strlwr(char \*s) | 将字符串s转换为小写形式 |
| char \*strupr(char \*s) | 将字符串s转换为大写形式 |
| char \*strncat(char \*dest,char \*src,int n) | 把src所指字符串的前n个字符添加到dest结尾处(覆盖dest结尾处的’\0’)并添加’\0’ |
| int strcmp(char \*s1,char \* s2，int n) | 比较字符串s1和s2的前n个字符 |
| int strnicmp(char \*s1,char \* s2，int n) | 比较字符串s1和s2的前n个字符但不区分大小写 |
| char \*strncpy(char \*dest, char \*src, int n) | 把src所指由NULL结束的字符串的前n个字节复制到dest所指的数组中 |
| char \*strpbrk(char \*s1, char \*s2) | 在字符串s1中寻找字符串s2中任何一个字符相匹配的第一个字符的位置，空字符NULL不包括在内 |
| char \*strrev(char \*s) | 把字符串s的所有字符的顺序颠倒过来（不包括空字符NULL） |
| char \*strset(char \*s, char c) | 把字符串s中的所有字符都设置成字符c |
| char \*strstr(char \*haystack, char \*needle) | 从字符串haystack中寻找needle第一次出现的位置（不比较结束符NULL) |
| char \*strtok(char \*s, char \*delim) | 分解字符串为一组标记串。s为要分解的字符串，delim为分隔符字符串 |
| int strnicmp(char \*s1,char \* s2，int n) | 比较字符串s1和s2的前n个字符但不区分大小写 |