

C 語言標準函式庫

- ◆ C-1 標準輸出輸入函數<[stdio.h](#)>
- ◆ C-2 字元檢查函數<[ctype.h](#)>
- ◆ C-3 字串函數<[string.h](#)>
- ◆ C-4 數學函數<[math.h](#)>
- ◆ C-5 日期/時間函數<[time.h](#)>
- ◆ C-6 工具函數<[stdlib.h](#)>

C-1 標準輸出輸入函數<[stdio.h](#)>

FILE* [fopen](#)(const char* filename, const char* mode) :

使用 mode 模式開啟參數 filename 的檔案，傳回檔案串流，失敗傳回 NULL。

FILE* [freopen](#)(const char* filename, const char* mode, FILE* stream) :

關閉檔案後重新開啟檔案。

int [fflush](#)(FILE* stream) :

清除緩衝區的內容，成功傳回 0，失敗傳回 EOF。

int [fclose](#)(FILE* stream) :

關閉檔案。

int [remove](#)(const char* filename) :

刪除參數的檔案，失敗傳回非零值。

int [rename](#)(const char* oldname, const char* newname) :

將檔案名稱 oldname 改為 newname，失敗傳回非零值。

FILE* [tmpfile](#)() :

建立"wb+"模式的暫存檔案，當結束程式後就會關閉且刪除此檔案。

`char* tmpname(char s[L_tmpnam]) :`

指定暫存檔案的名稱為 s。

`int setvbuf(FILE* stream, char* buf, int mode, size_t size) :`

指定串流暫存區尺寸 size，使用 mode 參數值 `_IOFBF` 為完整暫存區、`_IOLBF` 是線性暫存區或 `_IONBF` 沒有暫存區。

`void setbuf(FILE* stream, char* buf) :`

指定串流的暫存區為參數 buf。

`int fprintf(FILE* stream, const char* format, ...) :`

將格式化字串寫入檔案串流。

`int printf(const char* format, ...) :`

在標準輸出顯示格式化字串。

`int sprintf(char* s, const char* format, ...) :`

將格式化字串輸出到字串 s。

`int fscanf(FILE* stream, const char* format, ...) :`

從檔案串流讀取指定格式的資料。

`int scanf(const char* format, ...) :`

從標準輸入讀取指定格式的資料。

`int sscanf(char* s, const char* format, ...) :`

從字串 s 讀取指定格式的資料。

`int fgetc(FILE* stream) :`

從檔案串流讀取一個字元。

`char* fgets(char* s, int n, FILE* stream) :`

從檔案串流讀取一個字串。

`int fputc(int c, FILE* stream) :`

寫入一個字元到檔案。

`char* fputs(const char* s, FILE* stream) :`

寫入一個字串到檔案。

`int getc(FILE* stream) :`

從檔案串流讀取一個字元。

`int getchar(void) :`

從標準輸入讀取一個字元。

`char* gets(char* s) :`

從標準輸入讀取一個字串。

`int putc(int c, FILE* stream) :`

寫入一個字元到檔案。

`int putchar(int c) :`

在標準輸出顯示一個字元。

`int puts(const char* s) :`

在標準輸出顯示一個字串。

`int ungetc(int c, FILE* stream) :`

將一個字元放回檔案串流。

`size_t fread(void* ptr, size_t size, size_t nobj, FILE* stream) :`

從檔案讀取指定大小的資料。

`size_t fwrite(const void* ptr, size_t size, size_t nobj, FILE* stream) :`

將指定大小的資料寫入檔案。

`int fseek(FILE* stream, long offset, int origin) :`

移動檔案指標到 `offset` 位移量，其方向是 `origin` 參數值 `SEEK_SET` 的檔案開頭、`SEEK_CUR` 是目前位置或 `SEEK_END` 檔尾。

`long ftell(FILE* stream) :`

目前檔案指標的位置。

`void rewind(FILE* stream) :`

重設檔案指標到檔頭。

`int feof(FILE* stream) :`

是否到達檔尾。

int ferror(FILE* stream) :

是否檔案串流產生錯誤。

C-2 字元檢查函數<ctype.h>

int isalnum(int c) :

isalpha(c)或 isdigit(c)的字元。

int isalpha(int c) :

isupper(c)或 islower(c)的字元。

int iscntrl(int c) :

是否是 ASCII 控制字元。

int isdigit(int c) :

是否是數字。

int isgraph(int c) :

是否是顯示字元，不含空白字元。

int islower(int c) :

是否是小寫字元。

int isprint(int c) :

是否是顯示字元 0x20 (' ')到 0x7E ('~')。

int ispunct(int c) :

是否是顯示字元，不包含空白、字母、數字字元。

int isspace(int c) :

是否是空白字元。

int isupper(int c) :

是否是大寫字元。

int isxdigit(int c) :

是否是十六進位字元。

int tolower(int c) :
轉換成小寫字元。

int toupper(int c) :
轉換成大寫字元。

C-3 字串函數<string.h>

char* strcpy(char* s, const char* ct) :
將字串 ct 複製到字串 s。

char* strncpy(char* s, const char* ct, size_t n) :
將字串 ct 前 n 個字元複製到字串 s。

char* strcat(char* s, const char* ct) :
連結字串 ct 到字串 s 之後。

char* strncat(char* s, const char* ct, size_t n) :
連結字串 ct 前 n 個字元到字串 s。

int strcmp(const char* cs, const char* ct) :
比較字串 cs 和 ct。

int strncmp(const char* cs, const char* ct, size_t n) :
比較字串 cs 和 ct 的前 n 個字元。

char* strchr(const char* cs, int c) :
傳回字元 c 第一次出現在字串 cs 位置的指標。

char* strrchr(const char* cs, int c) :
傳回字元 c 最後一次出現在字串 cs 位置的指標。

char* strpbrk(const char* cs, const char* ct) :
傳回字串 ct 任何字元在字串 cs 第一次出現的位置指標。

char* strstr(const char* cs, const char* ct) :
傳回字串 ct 在字串 cs 第一次出現的位置指標。

size_t strlen(const char* cs) :

傳回字串 cs 的長度。

char* strerror(int n) :

傳回指定錯誤代碼的說明文字內容。

char* strtok(char* s, const char* t) :

以字串 t 的任何字元為分隔字元，找尋字串 s 中下一個 token 記號。

void* memcpy(void* s, const void* ct, size_t n) :

從位置 ct 複製 n 個字元到位置 s，傳回 s。

void* memmove(void* s, const void* ct, size_t n) :

從位置 ct 搬移 n 個字元到位置 s，傳回 s。

int memcmp(const void* cs, const void* ct, size_t n) :

比較位置 ct 和位置 cs 的前 n 個字元。

void* memchr(const void* cs, int c, size_t n) :

傳回 cs 位置開始前 n 個字元第一次出現字元 c 的位置指標。

void* memset(void* s, int c, size_t n) :

取代 cs 位置開始前 n 個字元成為字元 c，傳回位置指標 s。

C-4 數學函數<math.h>

double exp(double x) :

自然數的指數 e^x 。

double log(double x) :

自然對數 $\log x$

double log10(double x) :

十為底的對數 $\log_{10}x$ 。

double pow(double x, double y) :

傳回參數 x 為底，參數 y 的次方值 x^y 。

double sqrt(double x) :

參數 x 的平方根。

double ceil(double x) :

傳回大於或等於參數 x 的最小 double 整數。

double floor(double x) :

傳回小於或等於參數 x 的最大 double 整數。

double fabs(double x) :

傳回參數 x 的絕對值。

hypot(double x, double y) :

傳回 (x^2+y^2) 公式的值

double ldexp(double x, int n) :

x 乘以 2 的 n 次方是 $x*2^n$

double frexp(double x, int* exp) :

將參數 x 的浮點數分解成尾數和指標， $x = m*2^{exp}$ ，傳回 m 值的尾數，將指數存入參數 exp。

double modf(double x, double* ip) :

將浮點數 x 分解成整數和小數部分，傳回小數部分，將整數部分存入參數 ip。

double fmod(double x, double y) :

如果 y 為非零值，傳回浮點數 x/y 的餘數。

double sin(double x) :

正弦函數。

double cos(double x) :

餘弦函數。

double tan(double x) :

正切函數。

double asin(double x) :

反正弦函數。

double acos(double x) :

反餘弦函數。

double atan(double x) :

反正切函數。

double atan2(double y, double x) :

參數 y/x 的反正切函數值。

double sinh(double x) :

hyperbolic 正弦函數， $\sinh(x)=(e^x-e^{-x})/2$ 。

double cosh(double x) :

hyperbolic 餘弦函數， $\cosh(x)=(e^x+e^{-x})/2$ 。

double tanh(double x) :

hyperbolic 正切函數， $\tanh(x)=(e^x-e^{-x})/(e^2+e^{-x})$ 。

C-5 日期/時間函數 <time.h>

clock_t clock(void) :

傳回程式開始執行後所使用的 CPU 時間，以 ticks 為單位，除以常數 CLK_TCK 就是秒數。

time_t time(time_t* tp) :

傳回目前的曆法時間 (Calendar Time)，也會指定給參數的 tp 指標，如為無效時間，傳回-1。

double difftime(time_t time2, time_t time1) :

傳回參數 time2 和 time1 的時間差，即 time2-time1。

time_t mktime(struct tm* tp) :

將參數*tp 的當地時間改為曆法時間，如果不能轉換傳回-1。

char* asctime(const struct tm* tp) :

傳回參數 tm 結構指標轉換成日期/時間格式的字串，字串最後有換行字元\n。

char* ctime(const time_t* tp) :

傳回參數 time_t 指標轉換成當地日期/時間的字串，字串最後有換行字元\n。

`struct tm* gmtime(const time_t* tp) :`

傳回將參數的 `time_t` 指標轉換成 UTC (Coordinated Universal Time) 日期/時間的 `tm` 結構指標。

`struct tm* localtime(const time_t* tp) :`

傳回將參數的 `time_t` 指標轉換成當地日期/時間的 `tm` 結構指標。

`size_t strftime(char* s, size_t smax, const char* fmt, const struct tm* tp) :`

將參數 `tp` 的日期/時間以格式化字串 `fmt` 輸出到字串 `s` , `s` 最多儲存 `smax` 個字元。

C-6 工具函數 <stdlib.h>

`int abs(int n)`、`long labs(long n) :`

傳回整數 `n` 的絕對值。

`double atof(const char* s) :`

將參數字串 `s` 轉換成浮點數，如果字串不能轉換傳回 0.0。

`int atoi(const char* s) :`

將參數字串 `s` 轉換成整數，如果字串不能轉換傳回 0。

`long atol(const char* s) :`

將參數字串 `s` 轉換成長整數，如果字串不能轉換傳回 0。

`double strtod(const char* s, char** endp) :`

函數忽略字串 `s` 前的空白字元，將數字部分轉換成浮點數，如果尚有未轉換的部分字串，則設成參數 `endp` 指標。

`long strtol(const char* s, char** endp, int base) :`

函數忽略字串 `s` 前的空白字元，將數字部分轉換成長整數，如果尚有未轉換的部分字串，則設成參數 `endp` 指標。

`unsigned long strtoul(const char* s, char** endp, int base) :`

如同 `strtol` 函數，其傳回值是無符號長整數。

`void* calloc(size_t nobj, size_t size) :`

傳回一塊參數 `nobj` 陣列大小的記憶體指標，`nobj` 元素大小為 `size` 初值為 0，錯誤傳回 `NULL`。

void* malloc(size_t size) :

傳回大小 size 記憶體指標，沒有指定初值，錯誤傳回 NULL。

void* realloc(void* p, size_t size) :

將指標 p 的記憶體改為 size 大小，不會更改原記憶體的值，多配置部分初值為 0，錯誤傳回 NULL。

void free(void* p) :

釋放參數 p 指標的記憶體空間。

void abort() :

強迫程式以不正常方式結束，如同呼叫 raise(SIGABRT)函數。

void exit(int status) :

程式以正常方式結束，傳回系統環境狀態值，0 表示正常結束。

int system(const char* s) :

將字串 s 的指令傳給環境來執行，也就是執行 MS-DOS 的指令。

char* getenv(const char* name) :

傳回參數 name 的環境字串，如果沒有傳回 NULL。

void* bsearch(const void* key, const void* base, size_t n, size_t size, int (*cmp)(const void* keyval, const void* datum)) :

陣列基礎的二元搜尋函數，陣列是參數 base，鍵值是參數 key，n 是陣列大小，size 是每個元素的大小，最後的參數是指向函數的指標，這是比較元素大小的函數，找到傳回該元素指標，沒有找到傳回 NULL。

void qsort(void* base, size_t n, size_t size, int (*cmp)(const void*, const void*)) :

陣列基礎的快速排序法函數，陣列是參數 base，n 是陣列大小，size 是每個元素的大小，最後的參數是指向函數的指標，這是比較元素大小的函數。

int rand(void) :

傳回亂數的整數值，其值的範圍是 0 到 RAND_MAX 常數，其值為 0x7FFF。

void srand(unsigned int seed) :

指定亂數的種子數，參數是無符號整數，如果沒有指定，預設的種子數為 1。