



Principales bibliothèques du langage C

Introduction

Ce document contient les principales fonctions standards du C avec leur bibliothèque associée (Borland).

1- LES PRINCIPALES BIBLIOTHEQUES STANDARDS DU C	2
1.1- <i><stdlib.h> : Gestion de la mémoire, conversions et fonctions systèmes</i>	2
1.2- <i><string.h> : Gestion de chaînes de caractères</i>	2
1.3- <i><conio.h> : Fonctions diverses</i>	3
1.4- <i><ctype.h> : Manipulation de caractères</i>	3
1.5- <i><math.h> : Librairie de fonctions mathématiques</i>	4
1.6- <i><stdio.h> : Gestion des fichiers et des Entrées/Sorties en général</i>	5



1- LES PRINCIPALES BIBLIOTHEQUES STANDARDS DU C

Nom	Rôle
stdio.h	Gestion des E/S.
stdlib.h	Gestion de la mémoire, conversions et fonctions systèmes.
string.h	Gestion des chaînes de caractères.
conio.h	Gestion de l'écran.
ctype.h	Manipulation de caractères.
math.h	Fonctions mathématiques.

1.1- <stdlib.h> : Gestion de la mémoire, conversions et fonctions systèmes

PROTOTYPE	RÔLE
ALLOCATION DYNAMIQUE DE LA MEMOIRE	
void *calloc(int nbelt, unsigned int size)	Renvoie pointeur sur <i>nbelt</i> (init. A 0), de <i>size</i> octets ; si échec : NULL
void *malloc(unsigned int size)	Retourne un pointeur sur <i>size</i> octets ; si échec : NULL.
void *realloc(void *ptr, unsigned int size)	Change taille de zone pointée par <i>ptr</i> à <i>size</i> octets ; si échec : NULL
void free(void *ptr)	Libère les octets pointés par <i>ptr</i> .
CONVERSIONS DE CHAINES DE CARACTERES	
Double atof(char *s)	Retourne un réel, résultat de la conversion de *s ; si échec : 0.
Int atoi(char *s)	Retourne un entier, résultat de la conversion de *s ; si échec : 0.
Long atol(char *s)	Retourne un entier long, résultat de la conversion de *s ; si échec : 0.
Char *itoa(int val, char *s, int base)	Retourne une chaîne, dans <i>s</i> et en retour, résultat de la conversion de l'entier val. <i>Base</i> (2 à 36) est la base de numération.
FONCTION SYSTEME	
void exit(int status)	Termine un programme avec le code d'erreur <i>status</i> : 0 (EXIT_SUCCESS) est une terminaison normale ; sinon EXIT_FAILURE.
Void abort(void)	Interrompt exécution et message « Abnormal program termination ».
int system(const char *command)	Exécute l'instruction MS-DOS <i>command</i> . Si exéc OK : renvoi 0, sinon : -1.

1.2- <string.h> : Gestion de chaînes de caractères

PROTOTYPE	RÔLE
MANIPULATION DE CHAINES	
int strcmp(char *s1, char *s2)	Compare <i>s1</i> et <i>s2</i> lexicographiquement. Renvoie nombre < 0 si <i>s1</i> précède <i>s2</i> , 0 si =, un nombre > 0 si <i>s1</i> suit <i>s2</i>
int strncmp(char *s1, char *s2, int n)	Compare les <i>n</i> premiers caractères de <i>s1</i> et de <i>s2</i> . Renvoie nombre < 0 si <i>s1</i> précède <i>s2</i> , 0 si =, un nombre > 0 si <i>s1</i> suit <i>s2</i>
int strlen(char *s)	Retourne la longueur de <i>s1</i> , sans compter '\0'.
Char *strcat(char *s1, char *s2)	Concatène <i>s2</i> à <i>s1</i> avec un zéro terminal. Retourne <i>s1</i> .
Char *strncat(char *s1, char *s2, int n)	Concatène au plus les <i>n</i> premiers caractères de <i>s2</i> à <i>s1</i> . Retourne <i>s1</i> .
Char *strcpy(char *s1, char *s2)	Copie <i>s2</i> dans <i>s1</i> , en incluant '\0'. Retourne <i>s1</i> .
Char *strncpy(char *s1, char *s2, int n)	Copie au plus les <i>n</i> premiers caractères de <i>s2</i> dans <i>s1</i> . Retourne <i>s1</i> .
Char *strdup(char *s1)	Duplique <i>s1</i> en mémoire dynamique. Retourne pointeur sur nouvelle zone mémoire ; si échec : NULL.



RECHERCHES D'OCCURRENCES	
char *strchr(char *s,int c)	Renvoie l'adresse du premier caractère <i>c</i> dans <i>s</i> ; si non trouvé : NULL
char *strpbrk(char *s1 , char *s2)	Renvoie adresse du 1 ^o carac. De <i>s1</i> contenu dans <i>s2</i> ; si non trouvé :NULL
char *strrchr(char *s,int c)	Retourne l'adresse du dernier caractère <i>c</i> dans <i>s</i> ; si non trouvé : NULL
char *strstr(char *s1, char *s2)	Cherche <i>s2</i> dans <i>s1</i> .
Char *strtok(char *s1,char *s2)	Identifie des mots dans <i>s1</i> séparés par la chaîne <i>s2</i> . Retourne adresse sur un délimiteur <i>s2</i> trouvé.

1.3- <conio.h> : Fonctions diverses

PROTOTYPE	RÔLE
int getch(void)	Lit un caractère au clavier. Retourne le caractère lu ; si touche de fonction ou flèche : 0.
int getche(void)	Comme getch() avec écho à l'écran.
void gotoxy(int colonne,int ligne)	Place le curseur écran au point (colonne, ligne) L'origine est (1,1) en haut à gauche de l'écran. Si les coordonnées sont incorrectes, la fonction n'est pas exécutée
void clrscr(void)	Efface la fenêtre en mode texte.

1.4- <ctype.h> : Manipulation de caractères

PROTOTYPE	RÔLE
TESTS DE CARACTERES	
int isalnum(int c)	Macro teste si <i>c</i> est carac alphanumérique (lettre, isalpha() ou chiffre, isdigit()). Retourne une valeur non nulle si test positif.
int isalpha(int c)	Macro qui teste si <i>c</i> est une lettre : 'a'..'z','A'..'Z'. Retourne une valeur non nulle si test positif.
int islower(int c)	Macro qui teste si <i>c</i> est une lettre minuscule. Retourne une valeur non nulle si test positif.
int isupper(int c)	Macro qui teste si <i>c</i> est une lettre majuscule. Retourne une valeur non nulle si test positif.
int isascii(int c)	Macro qui teste si <i>c</i> est un caractère ASCII. Retourne une valeur non nulle si test positif.
int isdigit(int c)	Macro qui teste si <i>c</i> est un chiffre : '0'..'9'. Retourne une valeur non nulle si test positif.
int isxdigit(int c)	Macro qui teste si <i>c</i> est un chiffre hexadécimal : '0'..'9', 'a'..'f', 'A'..'F'. Retourne une valeur non nulle si test positif.
int ispunct(int c)	Macro qui teste si <i>c</i> n'est ni isalnum(), iscntrl() ou isspace(). Retourne une valeur non nulle si test positif.
int iscntrl(int c)	Macro qui teste si <i>c</i> est un caractère de contrôle : ASCII 0..31,127. Retourne une valeur non nulle si test positif.
int isgraph(int c)	Macro qui teste si <i>c</i> est un caractère imprimable (sauf l'espace). Retourne une valeur non nulle si test positif.
int isprint(int c)	Macro qui teste si <i>c</i> est un caractère imprimable : ASCII 32 à 126. Retourne une valeur non nulle si test positif.
int isspace(int c)	Macro teste si <i>c</i> est 1 carac séparateur: espace, tab, saut page, RC, saut ligne. Retourne une valeur non nulle si test positif.



CONVERSIONS DE CARACTERES	
int toascii(int c)	Convertit <i>c</i> au format ASCII.
int _tolower(int c)	<i>Macro</i> convertit <i>c</i> en minuscule. L'utilisateur doit être sûr de la validité du carac. Retourne <i>c</i> ou le résultat de la conversion.
int tolower(int c)	<i>Fonction</i> qui convertit <i>c</i> en minuscule. Retourne <i>c</i> ou le résultat de la conversion.
int _toupper(int c)	<i>Macro</i> convertit <i>c</i> en majuscule. L'utilisateur doit être sûr de la validité du carac. Retourne <i>c</i> ou le résultat de la conversion.
int toupper(int c)	<i>Fonction</i> qui convertit <i>c</i> en majuscule. Retourne <i>c</i> ou le résultat de la conversion.

1.5- <math.h> : Librairie de fonctions mathématiques

PROTOTYPE	RÔLE
-----------	------

FONCTIONS TRIGONOMETRIQUES

double acos(double x)	Retourne l'arcosinus de <i>x</i> (rad) ; si échec : 0.
double acosh(double x)	Retourne l'arcosinus hyperbolique de <i>x</i> (rad) ; si échec : 0.
double cos(double x)	Retourne le cosinus de <i>x</i> (rad) ; si échec : 0.
double cosh(double x)	Retourne le cosinus hyperbolique de <i>x</i> (rad) ; si échec : 0.
double asin(double x)	Retourne l'arcsinus de <i>x</i> (rad) ; si échec : 0.
double asinh(double x)	Retourne l'arcsinus hyperbolique de <i>x</i> (rad) ; si échec : 0.
double sin(double x)	Retourne le sinus de <i>x</i> (rad) ; si échec : 0.
double sinh(double x)	Retourne le sinus hyperbolique de <i>x</i> (rad) ; si échec : 0.
double atan(double x)	Retourne l'arctangente de <i>x</i> (rad) ; si échec : 0.
double atanh(double x)	Retourne l'arctangente hyperbolique de <i>x</i> (rad) ; si échec : 0.
double tan(double x)	Retourne la tangente de <i>x</i> (rad) ; si échec : 0.
double tanh(double x)	Retourne la tangente hyperbolique de <i>x</i> (rad) ; si échec : 0.

FONCTIONS ARITHMETIQUES

double cbrt(double x)	Retourne la racine cubique de <i>x</i> .
double pow(double x,double y)	Renvoie le résultat de <i>x</i> puissance <i>y</i> ; si échec : HUGE_VAL.
double sqrt(double x)	Retourne la racine carrée de <i>x</i> ; si échec : 0.
double fmod(double x,double y)	Renvoie le reste réel de la division entière de <i>x</i> par <i>y</i> , avec le signe de <i>x</i> .
double remainder(double x,double y)	Retourne le reste de la division entière de <i>x</i> par <i>y</i> .
double exp(double x)	Retourne l'exponentielle de <i>x</i> ; si échec : HUGE_VAL.
double log(double x)	Retourne le logarithme népérien de <i>x</i> ; si échec : HUGE_VAL.
double log10(double x)	Retourne le logarithme décimal de <i>x</i> ; si échec : HUGE_VAL.
double rint(double x)	Retourne la valeur entière la plus proche de <i>x</i> .
double ceil(double x)	Retourne le plus petit entier supérieur à <i>x</i> .
double floor(double x)	Retourne le plus grand entier inférieur à <i>x</i> .
double copysign(double x,double y)	Retourne <i>x</i> avec le signe de <i>y</i> .
double fabs(double x)	Retourne la valeur absolue de <i>x</i> .



1.6- <stdio.h> : Gestion des fichiers et des Entrées/Sorties en général

PROTOTYPE	RÔLE
-----------	------

OUVERTURE/FERMETURE FICHIERS	
FILE *fopen(const char *name,const char *type)	Ouvre le fichier <i>name</i> dans le mode <i>type</i> . Retourne le pointeur sur fichier ; si échec : NULL.
FILE *freopen(const char *name,const char *type,FILE *stream)	Ouvre le fichier <i>name</i> dans le mode <i>type</i> et l'associe à <i>stream</i> .
int fclose(FILE *stream)	Ferme le fichier associé au flux <i>stream</i> , ouvert avec fopen(). Retourne 0 ; si échec : EOF.

DEPLACEMENT DANS LES FICHIERS	
int fseek(FILE *stream,long offset,int origin)	Change la position courante dans le fichier <i>stream</i> de <i>offset</i> octets depuis <i>origin</i> (SEEK_SET -début fic.-, SEEK_CUR -pos. cour.-, SEEK_END -fin-).
long ftell(FILE *stream)	Retourne la position courante dans le fichier <i>stream</i> (en octets depuis le début du fichier).
int feof(FILE *stream)	Retourne une valeur non nulle si <i>stream</i> est à la fin du fichier.
void rewind(FILE *stream)	Positionnement en début du fichier <i>stream</i> .

LECTURE A PARTIR D'UN FLUX	
int scanf(const char *format,...)	Lit des valeurs formatées sur l'entrée standard STDIN et les affecte aux adresses fournies en paramètres variables. Retourne nombre de variables lues correctement.
int sscanf(const char *buffer,const char *format,...)	Lit dans <i>buffer</i> des valeurs formatées et les affecte aux adresses fournies en paramètres variables. Retourne nombre de variables lues correctement.
char *gets(char *buffer)	Lit une ligne terminée par RC sur l'entrée standard STDIN. Retourne le résultat dans <i>buffer</i> et en retour de fonction, en remplaçant RC par '\0' ; si échec : NULL.
int getchar(void)	Macro qui lit un octet (correspondant à un unsigned char) sur l'entrée standard ; attend un retour à la ligne. Retourne le caractère lu ; si échec : EOF.
int fgetchar(void)	Fonction qui lit un octet (correspondant à un unsigned char) sur l'entrée standard ; attend un retour à la ligne. Retourne le caractère lu ; si échec : EOF.
int getc(FILE *stream)	Macro qui lit un octet (correspondant à un unsigned char) à la position courante du flux <i>stream</i> . Incrémente la position courante du pointeur de fichier. Retourne le caractère lu ; si échec ou fin de fichier : EOF.
int fgetc(FILE *stream)	Fonction qui lit un octet (correspondant à un unsigned char) à la position courante du flux <i>stream</i> . Incrémente la position courante du pointeur de fichier. Retourne le caractère lu ; si échec ou fin de fichier : EOF.
int getw(FILE *stream)	Lit un mot à la position courante du fichier <i>stream</i> . Incrémente la position courante du pointeur de fichier de 2 octets. Retourne le mot lu ; si échec ou fin de fichier : EOF.



<code>char *fgets(char *s,int n,FILE *stream)</code>	Lit 1 ligne ($n-1$ carac max ou RC) dans fichier associé à <i>stream</i> . Retourne la ligne dans <i>*s</i> et en retour de fonction, si échec : NULL.
<code>int fread(void *buffer,int size,int nitems,FILE *stream)</code>	Lit <i>nitems</i> blocs de <i>size</i> octets à la position courante du fichier <i>stream</i> . Mise à jour du pointeur de fichier. Retourne le résultat dans <i>*buffer</i> , ainsi que le nombre d'éléments lus.
<code>int fscanf(FILE *stream,const char *format,...)</code>	Lit les valeurs formatées par <i>*format</i> dans le fichier <i>stream</i> . Retourne les valeurs lues aux adresses des paramètres variables, ainsi que le nombre de données lues ; si erreur ou fin de fichier : EOF.

ECRITURE DANS UN FLUX

<code>int printf(const char *format,...)</code>	Convertit les données fournies en paramètres variables en une chaîne de caractères et les écrit sur la sortie standard STDOUT. Retourne nombre de caractères imprimés ; si échec : EOF.
<code>int puts(const char *s)</code>	Ecrit la chaîne <i>*s</i> et un retour à la ligne sur la sortie standard STDOUT. Retourne nombre de caractères imprimés ; si échec : EOF.
<code>int putchar(int c)</code>	<i>Macro</i> qui écrit l'octet <i>c</i> (convertit en unsigned char) sur la sortie standard STDOUT. Retourne le caractère écrit ; si erreur : EOF.
<code>int fputc(int c)</code>	<i>Fonction</i> qui écrit l'octet <i>c</i> (convertit en unsigned char) sur la sortie standard STDOUT. Retourne le caractère écrit ; si erreur : EOF.
<code>int putc(int c,FILE *stream)</code>	<i>Macro</i> qui écrit l'octet <i>c</i> (convertit en unsigned char) à la position courante du flux <i>stream</i> . Pointeur sur fichier incrémenté de 1 carac. Retourne le caractère écrit ; si erreur : EOF.
<code>int fputc(int c,FILE *stream)</code>	<i>Fonction</i> qui écrit l'octet <i>c</i> (convertit en unsigned char) à la position courante du flux <i>stream</i> . Pointeur sur fichier incrémenté de 1 carac. Retourne le caractère écrit ; si erreur : EOF.
<code>int putw(int w,FILE *stream)</code>	Ecrit le mot <i>w</i> (fourni en binaire) à la position courante du fichier binaire associé à <i>stream</i> . Retourne la valeur écrite ; si échec : EOF.
<code>int fputs(const char *s,FILE *stream)</code>	Ecrit la chaîne <i>*s</i> dans le fichier <i>stream</i> . Retourne le dernier caractère écrit ; si échec : EOF.
<code>int fwrite(const void *buffer,int size,int nitems,FILE *stream)</code>	Ecrit <i>nitems</i> blocs de <i>size</i> octets, stockés dans <i>*buffer</i> , à la position courante du fichier <i>stream</i> . Mise à jour du pointeur de fichier. Retourne le nombre d'éléments écrits.
<code>int fprintf(FILE *stream,const char *format,...)</code>	Convertit des données fournies en paramètres variables en une chaîne de caractères et les écrit dans le flux <i>stream</i> . Retourne nombre de caractères imprimés ; si échec : EOF.

DIVERS

<code>int fflush(FILE *stream)</code>	Vide le buffer associé au flux <i>stream</i> (si fichier : transfert du buffer vers le fichier, sinon : initialisation du buffer).
<code>int sprintf(char *stream, const char *format[,arguments])</code>	Ecrit une chaîne formatée dans <i>stream</i> ; <i>arguments</i> sont les variables numériques éventuelles. Si erreur, retourne EOF.