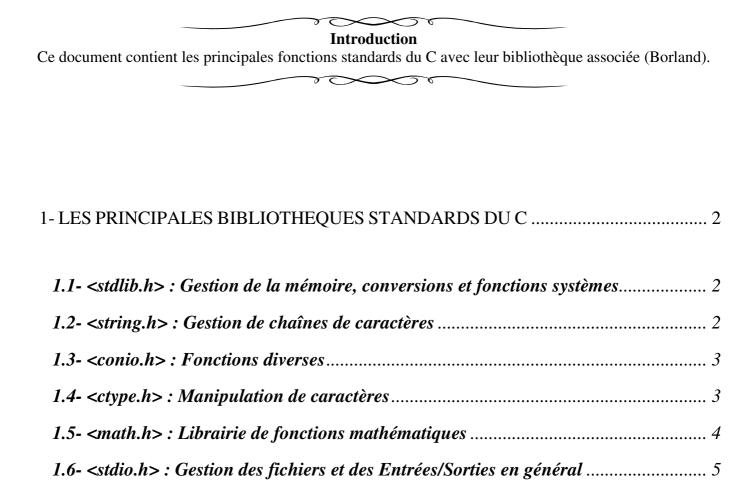
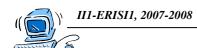
Principales bibliothèques du langage C





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	
ALLOCAT		ION DYNAMIQUE DE LA MEMOIRE	
void *calloc(int nbelt, unsigned int si	ze)	Renvoi pointeur sur <i>nbelt</i> (init. A 0), de <i>size</i> octets ; si échec :NULL	
<pre>void *malloc(unsigned int size)</pre>		Retourne un pointeur sur <i>size</i> octets ; si échec : NULL.	
Void *realloc(void *ptr, unsigned int	size)	Change taille de zone pointée par ptr à size octets ; si échec :NULL	
void free(void *ptr)		Libère les octets pointés par ptr.	
CONV	CONVERSIONS DE CHAINES DE CARACTERES		
Double atof(char *s)	Retou	rne 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 s et en retour, résultat de la conversion de l'entier val. $Base$ (2 à 36) est la base de numération.		
FONCTION SYSTEME			
void exit(int status)		ne un programme avec le code d'erreur <i>status</i> : 0 (EXIT_SUCCESS) est une naison normale ; sinon EXIT_FAILURE.	
Void abort(void)	Interre	ompt exécution et message « Abnormal program termination ».	
int system(const char *command)	Exécu	te 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 s1 et s2 lexicographiquement.
	Renvoi nombre<0 si s1 précède s2, 0 si =, un nombre>0 si s1 suit s2
int strncmp(char *s1,char *s2,int n)	Compare les n premiers caractères de $s1$ et de $s2$.
	Renvoi nombre<0 si s1 précède s2, 0 si =, un nombe >0 si s1 suit s2
int strlen(char *s)	Retourne la longueur de s1, sans compter '\0'.
Char *strcat(char *s1,char *s2)	Concatène s2 à s1 avec un zéro terminal. Retourne s1.
Char *strncat(char *s1,char *s2,int n)	Concatène au plus les n premiers caractères de $s2$ à $s1$. Retourne $s1$.
Char *strcpy(char *s1,char *s2)	Copie s2 dans s1, en incluant '\0'. Retourne s1.
Char *strncpy(char *s1,char *s2,int n)	Copie au plus les n premiers caractères de $s2$ dans $s1$. Retourne $s1$.
Char *strdup(char *s1)	Duplique s1 en mémoire dynamique.
	Retourne pointeur sur nouvelle zone mémoire ; si échec : NULL.



RECHERCHES D'OCCURRENCES	
char *strchr(char *s,int c)	Renvoi l'adresse du premier caractère c dans $*s1$; si non trouvé : NULL
<pre>char *strpbrk(char *s1 , char *s2)</pre>	Renvoi adresse du 1° carac. De s1 contenu dans s2; si non trouvé: NULL
char *strrchr(char *s,int c)	Retourne l'adresse du dernier caractère c dans $s1$; si non trouvé : NULL
char *strstr(char *s1, char *s2)	Cherche s2 dans s1.
Char *strtok(char *s1,char *s2)	Identifie des mots dans s1 séparés par la chaîne s2.
	Retourne adresse sur un délimiteur s2 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)	<i>Macro</i> teste si c est carac alphanumérique (lettre, isalpha() ou chiffre, isdigit()).		
	Retourne une valeur non nulle si test positif.		
int isalpha(int c)	Macro qui teste si c est une lettre : 'a''z','A''Z'.		
	Retourne une valeur non nulle si test positif.		
int islower(int c)	<i>Macro</i> 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 c est une lettre majuscule.		
	Retourne une valeur non nulle si test positif.		
int isascii(int c)	Macro qui teste si c est un caractère ASCII.		
	Retourne une valeur non nulle si test positif.		
int isdigit(int c)	<i>Macro</i> qui teste si c est un chiffre : '0''9'.		
	Retourne une valeur non nulle si test positif.		
int isxdigit(int c)	Macro qui teste si c est un chiffre héxadécimal : '0''9', 'a''f', 'A'.'F'.		
	Retourne une valeur non nulle si test positif.		
int ispunct(int c)	Macro qui teste si c n'est ni isalnum(), iscntrl() ou isspace().		
	Retourne une valeur non nulle si test positif.		
int iscntrl(int c)	<i>Macro</i> qui teste si c est un caractère de contrôle : ASCII 031,127.		
	Retourne une valeur non nulle si test positif.		
int isgraph(int c)	<i>Macro</i> 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 c est un caractère imprimable : ASCII 32 à 126.		
	Retourne une valeur non nulle si test positif.		
int isspace(int c)	Macro teste si c 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 c au format ASCII.		
int _tolower(int c)	Macro convertit c en minuscule. L'utilisateur doit être sûr de la validité du carac.	
	Retourne c ou le résultat de la conversion.	
int tolower(int c)	nt c) Fonction qui convertit c en minuscule.	
	Retourne c ou le résultat de la conversion.	
int _toupper(int c)		
Retourne c ou le résultat de la conversion.		
int toupper(int c)	Fonction qui convertit c en majuscule.	
Retourne c 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 x (rad) ; si échec : 0.
double acosh(double x)	Retourne l'arcosinus hyperbolique de x (rad) ; si échec : 0.
double cos(double x)	Retourne le cosinus de x (rad) ; si échec : 0.
double cosh(double x)	Retourne le cosinus hyperbolique de x (rad) ; si échec : 0.
double asin(double x)	Retourne l'arcsinus de x (rad) ; si échec : 0.
double asinh(double x)	Retourne l'arcsinus hyperbolique de x (rad) ; si échec : 0.
double sin(double x)	Retourne le sinus de x (rad) ; si échec : 0.
double sinh(double x)	Retourne le sinus hyperbolique de x (rad) ; si échec : 0.
double atan(double x)	Retourne l'arctangente de x (rad) ; si échec : 0.
double atanh(double x)	Retourne l'arctangente hyperbolique de x (rad) ; si échec : 0.
double tan(double x)	Retourne la tangente de x (rad) ; si échec : 0.
double tanh(double x)	Retourne la tangente hyperbolique de x (rad) ; si échec : 0.

FONCTIONS ARITHMETIQUES

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

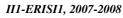
1.6- <stdio.h> : Gestion des fichiers et des Entrées/Sorties en général

PROTOTYPE	RÔLE

OUVERTURE/FERMETURE FICHIERS	
FILE *fopen(const char *name,const char	Ouvre le fichier <i>name</i> dans le mode <i>type</i> .
*type)	Retourne le pointeur sur fichier ; si échec : NULL.
FILE *freopen(const char *name,const char	Ouvre le fichier <i>name</i> dans le mode <i>type</i> et l'associe à <i>stream</i> .
*type,FILE *stream)	
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.	
<pre>void rewind(FILE *stream)</pre>	Positionnement en début du fichier stream.	

LEG	CTURE 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 *buffer 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.





char *fgets(char *s,int n,FILE *stream)	Lit 1 ligne (n-1 carac max ou RC) dans fichier associé à stream.
	Retourne la ligne dans *s et en retour de fonction, si échec : NULL.
int fread(void *buffer,int size,int nitems,FILE	Lit <i>nitems</i> blocs de <i>size</i> octets à la position courante du fichier <i>stream</i> .
*stream)	Mise à jour du pointeur de fichier.
	Retourne le résultat dans *buffer, ainsi que le nombre d'éléments lus.
<pre>int fscanf(FILE *stream,const char *format,)</pre>	Lit les valeurs formatées par *format dans le fichier stream.
	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			
int printf(const char *format,)	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.		
int puts(const char *s)	Ecrit la chaîne *s et un retour à la ligne sur la sortie standard STDOUT. Retourne nombre de caractères imprimés ; si échec : EOF.		
int putchar(int c)	Macro qui écrit l'octet c (convertit en unsigned char) sur la sortie standard STDOUT. Retourne le caractère écrit ; si erreur : EOF.		
int fputchar(int c)	Fonction qui écrit l'octet c (convertit en unsigned char) sur la sortie standard STDOUT. Retourne le caractère écrit ; si erreur : EOF.		
int putc(int c,FILE *stream)	Macro qui écrit l'octet c (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.		
int fputc(int c,FILE *stream)	Fonction qui écrit l'octet c (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.		
int putw(int w,FILE *stream)	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.		
int fputs(const char *s,FILE *stream)	Ecrit la chaîne *s dans le fichier stream. Retourne le dernier caractère écrit ; si échec : EOF.		
int fwrite(const void *buffer,int size,int nitems,FILE *stream)	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.		
int fprintf(FILE *stream,const char *format,)	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		
int fflush(FILE *stream)	Vide le buffer associé au flux <i>stream</i> (si fichier : transfert du buffer vers le fichier, sinon : initialisation du buffer).		
<pre>int sprintf(char *stream, const char *format[,arguments])</pre>	Ecrit une chaîne formatée dans <i>stream</i> ; <i>arguments</i> sont les variables numériques éventuelles. Si erreur, retourne EOF.		