



Petit dictionnaire du langage C

Introduction

Ce document contient la traduction en C des instructions vues dans les Algorithmes, les principales fonctions standards du C avec leur bibliothèque associée et le tableau des codes ASCII.

1- FICHIER SOURCE EN LANGAGE C	2
2- TRADUCTION EN C DES INSTRUCTIONS VUES DANS LES ALGOS	3
2.1- <i>Types de données</i>	3
2.2- <i>Déclaration de variables</i>	3
2.3- <i>Déclaration de constantes</i>	3
2.4- <i>Affichage écran</i>	4
2.5- <i>Saisie clavier</i>	4
2.6- <i>Opérateurs de calculs</i>	5
2.7- <i>Instructions SI</i>	5
2.8- <i>Instruction CHOIX</i>	6
2.9- <i>Boucle POUR</i>	6
2.10- <i>Boucles TANTQUE</i>	7
2.11- <i>Déclaration et utilisation d'un tableau à une dimension</i>	7
3- LES PRINCIPALES BIBLIOTHEQUES STANDARDS DU C	8
3.1- <i><stdlib.h> : Gestion de la mémoire, conversions et fonctions systèmes</i>	8
3.2- <i><string.h> : Gestion de chaînes de caractères</i>	8
3.3- <i><conio.h> : Fonctions diverses</i>	9
3.4- <i><ctype.h> : Manipulation de caractères</i>	9
3.5- <i><math.h> : Librairie de fonctions mathématiques</i>	10
3.6- <i><stdio.h> : Gestion des fichiers et des Entrées/Sorties en général</i>	10
4- LES CODES ASCII	12
4.1- <i>Code ASCII standard (sur 7 bits)</i>	13
4.2- <i>Un Code ASCII étendu (sur 8 bits) :</i>	15



1- FICHIER SOURCE EN LANGAGE C

```
/*  Auteur : nom de l'auteur du programme
    Nom du fichier : exemple.c
    Rôle : Mettre en évidence les éléments d'un programme en C
    Date : 03/01/2017 */
```

```
# include <stdio.h>
```

```
main()
{
    float prixHT ;
    /* Saisie du prix HT */
    printf("Entrez le prix HT : ");
    scanf("%f",&prixHT) ;

    return(0) ;
}
```

Fonction principale du programme ou main()

*ici débute l'exécution du programme
cette fonction est obligatoire*

Commentaire de type en-tête

Permet de documenter le programme avec 4 rubriques importantes : Auteur, Nom du fichier sur le disque dur, rôle du programme, date

Directives pour le pré-processeur

Ici, une inclusion de bibliothèque standard : pour utiliser une fonction standard, il faut inclure sa bibliothèque au début du programme

Dans la fonction main() :

- ✓ Accolade de début de main() ;
- ✓ **La partie Données :** Déclaration de variables et éventuellement de constantes locales au main() ;
- ✓ **La partie Instructions :**
 - Commentaires (au moins au début de chaque bloc) ;
 - Instructions et indentations associées (la plupart des instructions se terminent par ;).
- ✓ Accolade fermant le main().



2- TRADUCTION en C des INSTRUCTIONS vues dans les ALGOS

2.1- Types de données

Type dans les algorithmes	Type en langage C
carac ([0,255])	char
ensc [0,65 535]	unsigned short
esc [-32 768,32 767]	short
ensl [0,4 294 967 295]	unsigned long ou unsigned int
esl [-2 147 483 648,2 147 483 647]	long ou int
rsp : +- [3.4 10 ⁻³⁸ , 3.4 10 ³⁸]	float
rdp : +- [1.7 10 ⁻³⁰⁸ , 1.7 10 ³⁰⁸]	double

2.2- Déclaration de variables

Instruction algorithmique	Instruction en langage C
ALGO nomAlgo VAR type nomVariable DEBUT	main() { type nomVariable ;
ALGO nomAlgo VAR type nomTableau[5] DEBUT	main() { type nomTableau[5] ;

2.3- Déclaration de constantes

Instruction algorithmique	Instruction en langage C
ALGO nomAlgo CONST type NOMCONSTANTE= valeur DEBUT	<u><i>Juste après les #include :</i></u> 1) # define NOMCONSTANTE valeur 2) #define NOMCONSTANTE valeurTYPE avec, TYPE = ✓ L : entier signé long ✓ UL : entier non signé long ✓ U : entier non signé court ✓ F : réel simple précision ✓ LF : réel double précision <i>Remarque, définition d'une constante hexa :</i> # define NOMCONSTANTE OXF1D2
ALGO nomAlgo CONST type NOMCONSTANTE= valeur DEBUT	<u><i>Au début du main() :</i></u> main() { const type NOMCONSTANTE = valeur ; avec, type = types classiques (int, double, ...)



2.4- Affichage écran

Instruction algorithmique	Instruction en langage C
Afficher(nomVariable1,nomVariable2)	<pre>#include <stdio.h> /* Bibliothèque à inclure pour utiliser la fonction printf(), <u>AVANT LE MAIN()</u>*/ printf("%Format1%Format2",nomVariable1,nomVariable2);</pre> <p>Voir les chaînes de formatage dans le tableau suivant : elles précisent les types des variables affichées</p>
Afficher("message")	<pre>printf("message") ;</pre>
Afficher("résultats : ",nomVariable,".")	<pre>printf("résultats : %Format.",nomVariable) ;</pre>
Sauter 1 ligne	<pre>printf("\n") ;</pre>
Indenter 1 fois	<pre>printf("\t") ;</pre>
Aller à (colonne,ligne)	<pre>#include <conio.h> /* Bibliothèque pour la fonction gotoxy()*/ gotoxy(colonne,ligne) ;</pre>
Effacer écran	<pre>#include <conio.h> /* Bibliothèque où se trouve la fonction clrscr()*/ clrscr() ;</pre>

Quelques chaînes de formatages pour le PRINTF() :

Chaîne	Signification
%c	Caractère (char)
%s	Chaîne de caractères
%hu	Entier non signé court affiché en décimal (unsigned short)
%hx	Entier non signé court affiché en Hexadécimal (unsigned short)
%hd	Entier signé court affiché en décimal (short)
%lu	Entier non signé long affiché en décimal (unsigned long ou unsigned int)
%lx	Entier non signé long affiché en Hexadécimal (unsigned long ou unsigned int)
%ld ou %d	Entier signé long affiché en décimal (long ou int)
%f	Réel simple précision (float)
%.4f	float avec au maximum 4 décimales
%lf	Réel double précision (double)
%.2lf	double avec au maximum 2 décimales

2.5- Saisie clavier

Instruction algorithmique	Instruction en langage C
Saisir(&nomVariable)	<pre>#include <stdio.h> /* Bibliothèque pour scanf()*/ scanf("%Format",&nomVariable) ;</pre> <p>Voir les chaînes de formatage dans le tableau suivant : elles précisent dans quel type convertir les valeurs saisies</p>
Saisir(&nomVar1,&nomVar2)	<pre>scanf("%Format1%Format2",&nomVar1,&nomVar2) ;</pre>

**Les chaînes de formatages pour le SCANF() :**

Chaîne de formatage	Type
%c	char
%hu	unsigned short
%hd	short
%lu	unsigned long ou unsigned int
%ld	long ou int
%f	float
%lf	double

2.6- Opérateurs de calculs

Opérateur en algorithmique	Opérateur en langage C
Arithmétiques	
+	+
-	-
X	*
/	/
%	%
Logiques de comparaison	
==	==
≠	!=
>	>
<	<
≤	<=
≥	>=
Logiques booléens	
OU	
ET	&&
NON	!
Binaires	
+	
.	&
a	~a
⊕	^
Affectation :	=

2.7- Instructions SI

Instruction algorithmique	Instruction en langage C
SI (condition logique) ALORS instructions 1 SINON instructions 2 /*optionnel*/ FSINON	if (condition logique) // PAS DE ; ICI { instructions 1 ; } else /*optionnel*/ // PAS DE ; ICI { instructions 2 ; }



SI (condition logique 1) ALORS instructions 1 SINON SI (condition logique 2) instructions 2 SINON SI (condition logique 3) instructions 3 SINON /*optionnel*/ instructions 4 FSINON	if (condition logique 1) // PAS DE ; ICI { instructions 1 ; } else if (condition logique 2) // PAS DE ; ICI { instructions 2 ; } else if (condition logique 3) // PAS DE ; ICI { instructions 3 ; } else /*optionnel*/ // PAS DE ; ICI { instructions 4 ; }
---	--

2.8- Instruction CHOIX

Instruction algorithmique	Instruction en langage C
CHOIX (nomVariable) CAS val1 : instructions 1 SORTIR /*optionnel*/ CAS val2 : instructions 2 SORTIR /*optionnel*/ DEFAULT : instructions 3 /*optionnel*/ FCHOIX	switch (nomVariable) // PAS DE ; ICI { case val1 : { instructions 1 ; break ; /*optionnel*/ } case val2 : { instructions 2 ; break ; /*optionnel*/ } default : /*optionnel*/ { instructions 3 ; } }

2.9- Boucle POUR

Instruction algorithmique	Instruction en langage C
<u>Avec un pas positif :</u> POUR i de valInit à valFin, PAS de x instructions FPOUR	for (i=valInit ; i<=valFin ; i=i+x) // PAS DE ; ICI { instructions ; }
POUR i de valInit à valFin // PAS de 1 instructions FPOUR	for (i=valInit ; i<=valFin ; i++) // PAS DE ; ICI { instructions ; }
<u>Avec un pas négatif :</u> POUR i de valInit à valFin, PAS de x instructions FPOUR	for (i=valInit ; i>=valFin ; i=i-x) // PAS DE ; ICI { instructions ; }
POUR i de valInit à valFin, PAS de -1 instructions FPOUR	for (i=valInit ; i>=valFin ; i--) // PAS DE ; ICI { instructions ; }

**2.10- Boucles TANTQUE**

Instruction algorithmique	Instruction en langage C
FAIRE instructions TANTQUE (condition logique)	do <u>// PAS DE ; ICI</u> { instructions ; } while (condition logique) ;
TANTQUE (condition logique) instructions FTANTQUE	while (condition logique) <u>// PAS DE ; ICI</u> { instructions ; }

2.11- Déclaration et utilisation d'un tableau à une dimension

Instruction algorithmique	Instruction en langage C
ALGO NomAlgo VAR typeElémentTableau nomTableau[10] DEBUT	main() { 1) const unsigned short NBELT= 10 typeElementTableau nomTableau [NBELT] ; 2) typeElementTableau nomTableau [10] ;
ALGO NomAlgo VAR typeEltTableau tab[7]= {0,0,0,1,2,0,0} DEBUT	main() { typeEltTableau tab[7]= {0,0,0,1,2,0,0} ;
POUR i de 0 à (NBELT – 1) tab[i] = 0 FPOUR	for (i=0 ; i<=(NBELT – 1) ; i++) { tab[i] = 0 ; }
POUR i de 0 à (NBELT – 1) Afficher ("Elément d'indice ",i, " : ") Saisir (&tabEntier[i]) Sauter 1 ligne FPOUR	for (i=0 ; i<=(NBELT – 1) ; i++) { printf("Elément d'indice %hu : ",i); scanf("%ld",&tabEntier[i]); }
POUR i de 0 à (NBELT – 1) Afficher ("Elément n° ",i, " : ",tabEntier[i]) Sauter 1 ligne FPOUR	for (i=0 ; i<=(NBELT – 1) ; i++) { printf("Elément n°%hu: %ld\n",i,tabEntier[i]); }
sommeCumul= 0 POUR i de 0 à (NBELT – 1) sommeCumul= sommeCumul + tab[i] FPOUR	sommeCumul= 0 ; for (i=0 ; i<=(NBELT – 1) ; i++) { sommeCumul= sommeCumul + tab[i] ; }
produitCumul= 1 POUR i de 0 à (NBELT – 1) produitCumul= produitCumul x tab[i] FPOUR	produitCumul= 1 ; for (i=0 ; i<=(NBELT – 1) ; i++) { produitCumul= produitCumul * tab[i] ; }



3- 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.

3.1- <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 <i>val</i> . <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.

3.2- <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 *s ; si non trouvé : NULL
char *strpbrk(char *s1, char *s2)	Renvoie adresse du 1° 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>s1</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.



3.3- <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.

3.4- <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)	Macro 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)	Fonction qui convertit <i>c</i> en minuscule. Retourne <i>c</i> ou le résultat de la conversion.
int _toupper(int c)	Macro 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)	Fonction qui convertit <i>c</i> en majuscule. Retourne <i>c</i> ou le résultat de la conversion.



3.5- <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 x .
double pow(double x,double y)	Renvoie le résultat de x puissance y ; si échec : HUGE_VAL.
double sqrt(double x)	Retourne la racine carrée de x ; si échec : 0.
double fmod(double x,double y)	Renvoie 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 x par y .
double exp(double x)	Retourne l'exponentielle de x ; 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 x avec le signe de y .
double fabs(double x)	Retourne la valeur absolue de x .

3.6- <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)	<i>Macro</i> 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)	<i>Fonction</i> 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)	<i>Macro</i> 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)	<i>Fonction</i> 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 (<i>n</i> -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.
int fread(void *buffer, int size, int nitems, FILE *stream)	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.
int fscanf(FILE *stream, const char *format,...)	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, FILE *stream)</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.

4- LES CODES ASCII

American Standard Code for Information Interchange est un standard international normalisant 128 caractères, en les codant sur 7 bits. Le domaine a été étendu aux 256 caractères d'un octet, selon diverses normes de codage. Elles incluent la plupart les 128 premiers codes du code ASCII, mais diffèrent dans leur usage des 128 codes restant disponibles. Parmi ces normes, la norme dite **iso-latin-1** permet de représenter les caractères accentués présents dans les langues latines dont le français.



4.1- Code ASCII standard (sur 7 bits)

Les caractères de contrôle sont les 32 premiers caractères du code ASCII standard. Ils correspondent à des caractères de commande ou "caractères non imprimables", mais utiles tels que le "Retour Chariot" (touche Entrée) de code 13.

Caractères de contrôle (de 0 à 31) :

décimal	hexadécimal	aperçu	Caractère	signification
0	00	^@	NUL	Null, idle
1	01	^A	SOH	Start Of Heading
2	02	^B	STX	Start Of Text
3	03	^C	ETX	End Of Text
4	04	^D	EOT	End Of Transmission
5	05	^E	ENQ	ENQuiry
6	06	^F	ACK	ACKnowledge
7	07	^G	BEL	sonnerie
8	08	^H	BS	BackSpace
9	09	^I	HT	tabulation horizontale
10	0A	^J	LF	LineFeed (saut de ligne)
11	0B	^K	VT	tabualtion verticale, home
12	0C	^L	FF	Form Feed
13	0D	^M	CR	Carriage Return
14	0E	^N	SO	Shift Out
15	0F	^O	SI	Shift In
16	10	^P	DLE	Data Link Escape
17	11	^Q	DC1	Device Control 1, XON
18	12	^R	DC2	Device Control 2
19	13	^S	DC3	Device Control 3, XOFF
20	14	^T	DC4	Device Control 4
21	15	^U	NAK	Negative Acknowledge
22	16	^V	SYN	Synchronous Idle
23	17	^W	ETB	End Transmission Block
24	18	^X	CAN	Cancel
25	19	^Y	EM	End of Medium
26	1A	^Z	SUB	Substitute
27	1B	^[ESC	Escape sequence
28	1C	^\	FS	Cursor Right
29	1D	^]	GS	Cursor Left
30	1E	^^	RS	Cursor Up
31	1F	^_	US	Cursor Down



Caractères imprimables (de 32 à 127) :

décimal	hexadécimal	Caractère
32	20	Espace
33	21	!
34	22	"
35	23	#
36	24	\$
37	25	%
38	26	&
39	27	'
40	28	(
41	29)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

décimal	hexadécimal	caractère
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D]
94	5E	^
95	5F	_
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	
125	7D	}
126	7E	~
127	7F	DEL

**4.2- Un Code ASCII étendu (sur 8 bits) :****Caractères iso-latin-1 (de 128 à 255) :**

décimal	hexa	caractère
128	80	€
129	81	
130	82	,
131	83	f
132	84	„
133	85	...
134	86	†
135	87	‡
136	88	^
137	89	‰
138	8A	Š
139	8B	‹
140	8C	Œ
141	8D	
142	8E	Ž
143	8F	
144	90	
145	91	‘
146	92	’
147	93	“
148	94	”
149	95	•
150	96	—
151	97	—
152	98	~
153	99	™
154	9A	š
155	9B	›
156	9C	œ
157	9D	
158	9E	ž
159	9F	Ÿ
160	A0	espace
161	A1	ı
162	A2	ç
163	A3	£
164	A4	¤
165	A5	¥
166	A6	¦
167	A7	§
168	A8	¨
169	A9	©
170	AA	ª

décimal	hexa	caractère
171	AB	«
172	AC	¬
173	AD	
174	AE	®
175	AF	¯
176	B0	°
177	B1	±
178	B2	²
179	B3	³
180	B4	´
181	B5	µ
182	B6	¶
183	B7	·
184	B8	,
185	B9	¹
186	BA	º
187	BB	»
188	BC	¼
189	BD	½
190	BE	¾
191	BF	¿
192	C0	À
193	C1	Á
194	C2	Â
195	C3	Ã
196	C4	Ä
197	C5	Å
198	C6	Æ
199	C7	Ç
200	C8	È
201	C9	É
202	CA	Ê
203	CB	Ë
204	CC	Ì
205	CD	Í
206	CE	Î
207	CF	Ï
208	D0	Ð
209	D1	Ñ
210	D2	Ò
211	D3	Ó
212	D4	Ô
213	D5	Õ

décimal	hexa	caractère
214	D6	Ö
215	D7	×
216	D8	Ø
217	D9	Ù
218	DA	Ú
219	DB	Û
220	DC	Ü
221	DD	Ý
222	DE	Þ
223	DF	ß
224	E0	à
225	E1	á
226	E2	â
227	E3	ã
228	E4	ä
229	E5	å
230	E6	æ
231	E7	ç
232	E8	è
233	E9	é
234	EA	ê
235	EB	ë
236	EC	ì
237	ED	í
238	EE	î
239	EF	ï
240	F0	ð
241	F1	ñ
242	F2	ò
243	F3	ó
244	F4	ô
245	F5	õ
246	F6	ö
247	F7	÷
248	F8	ø
249	F9	ù
250	FA	ú
251	FB	û
252	FC	ü
253	FD	ý
254	FE	þ
255	FF	ÿ