LANGAGE C – FICHE 1
Types – Entrées/sorties standard

Structure du programme en langage C

Un programme simple en C doit avoir la structure suivante :

Un programme que l'on souhaite conserver et qui sera ré-utilisé commence par un en-tête en commentaires donnant les informations sur le programme.

Commentaires :
/* Commentaire sur
Plusieurs lignes */
Ou bien
// Commentaire sur une ligne possible

Les espaces et indentations ne sont pas significatifs en C mais sont indispensables pour la lisibilité du code.

Les différentes parties du code doivent être clairement délimitées.

La première partie comprend les importations de bibliothèques et modules.
Pour l'importation des bibliothèques C, il faut faire appel au préprocesseur par l'intermédiaire de la commande :
include
On fait ensuite suivre le nom de la bibliothèque C avec son extension (.h, comme header), entre <nom.h>.

Le programme principal est toujours contenu dans la fonction main() :
On précise le type de variable retournée (ici, un entier : int puis on écrit main().

Les instructions du programme principal sont entre accolades. En C, les accolades permettent de délimiter les blocs d'instructions.

En C, la première partie du programme principal contient la déclaration des variables :
type nomDeLaVariable ;

Cette partie est indispensable afin de réserver dans l'espace mémoire du programme (RAM) l'espace pour stocker les variables.

En fin de chaque ligne, on trouve un ;.

```
/*
En-tête du programme en commentaires :
Nom du programme
Auteur du programme
Date de création du programme
Version du programme
Intention du programme
*/

/*-------------------------------------------------------*/
/* IMPORTATION DES BIBLIOTHEQUES */
/*-------------------------------------------------------*/
# include <stdio.h>
# include <stdint.h>


/*-------------------------------------------------------*/
/* PROGRAMME PRINCIPAL                                   */
/*-------------------------------------------------------*/

/* Définition du programme principal :
type de la variable retournée par le programme principal puis main()*/

int main()
{
        /*-----------------------------------------*/
        /* Déclaration de variables                */
        int a = 18; // ou bien int a ;

        /*-----------------------------------------*/
        /* Instructions*/
        printf ("Hello world ! \n");

        /* Changement de la valeur de a */
        a = 20;
        printf ("En decimal : a=%d\n", a);

        return 0;
}
```

La dernière ligne du bloc d'instructions du programme principal indique return 0 ; L'usage est de retourner 0 lorsque le programme se déroule correctement.	

Règle de nommage des variables

Les 26 lettres de l'alphabet (minuscule et majuscule) peuvent être utilisées, ainsi que l'underscore (_), les chiffres (à condition que le nom ne commence pas par un chiffre)

Aucun caractère accentué, aucun caractère de ponctuation (y compris les espaces) ne doit être utilisé dans les noms de variables.

Les noms ne peuvent excéder 63 caractères. Mais il est très fortement conseillé d'opter pour des noms plus courts.

En C, il est d'usage de nommer les variables en minuscules, en distinguant les noms en utilisant les majuscules : nomDeLaVariable.

Les mots réservés sont :

auto	else	long	switch	_Atomic
break	enum	register	typedef	_Bool
case	extern	restrict	union	_Complex
char	float	return	unsigned	_Generic
const	for	short	void	_Imaginary
continue	goto	signed	volatile	_Noreturn
default	if	sizeof	while	_Static_assert
do	inline	static	_Alignas	_Thead_local
double	int	struct	_Alignof	

Bibliothèques

Syntaxe : #include <nomBibliotheque.h>

Deux bibliothèques sont utilisées pour ce premier TP.

Nom de la bibliothèque	Utilisation	Remarque
<stdint.h>	Définition de divers types d'entiers	
<stdio.h>	Contient les opérations d'entrée/sortie standard (printf, fprintf, scanf, fscanf, fgetc, fgets, fopen, fread, …)	Pour les entrées clavier, sorties écran, manipulation des fichiers.

Définition des types

Au cours du premier TP, il est demandé de manipuler des nombres. Ces nombres peuvent être stockés sous différents types en mémoire, suivant le nombre d'octets (ensemble de 8 bits) nécessaires pour les définir (pour les détails, voir cours sur la représentation des nombres en mémoire à venir).

Le codage des nombres se fait en binaire.

Par exemple, pour un entier stocké sur 3 bits, on a 8 = 2^3 valeurs possibles, entre 0 et 7 :

Binaire	Décimal		Binaire	Décimal
000	$0x2^2 + 0x2^1 + 0x2^0 = 0$		001	$0x2^2 + 0x2^1 + 1x2^0 = 1$
010	$0x2^2 + 1x2^1 + 0x2^0 = 2$		011	$0x2^2 + 1x2^1 + 1x2^0 = 3$
100	$1x2^2 + 0x2^1 + 0x2^0 = 4$		101	$1x2^2 + 0x2^1 + 1x2^0 = 5$
110	$1x2^2 + 1x2^1 + 0x2^0 = 6$		111	$1x2^2 + 1x2^1 + 1x2^0 = 7$

D'une manière générale, pour un nombre codé sur n bits, on peut avoir 2^n valeurs possibles, comprises entre 0 et 2^n-1.

Pour un entier négatif, le premier bit (bit de poids fort, le plus à gauche) sert à définir le signe : 0 pour un signe positif, 1 pour un signe négatif. La valeur absolue est donc codée sur n-1 bits. Le zéro est codé à deux reprises (+0 et -0).

Les types disponibles en C sont les suivants :

Type	**Nombre d'octets pour le stockage**	**Minimum**	**Maximum**
signed char	1	-127	127
unsigned char	1	0	255
short	2	-2^{15}-1 = -32 767	2^{15}-1 =32 767
unsigned short	2	0	2^{16}-1 = 65 535
int	4	-2^{15}-1	2^{15}-1
unsigned int	4	0	2^{32}-1
long	4	-2^{15}-1	2^{15}-1
unsigned long	4	0	2^{32}-1
long long	8	-2^{31}-1	2^{31}-1
unsigned long long	8	0	2^{64}-1
float	8	-1.10^{37}	1.10^{37}
double	8	-1.10^{37}	1.10^{37}

L'instruction sizeof (nomVariable) permet de récupérer le nombre d'octets permettant de stocker la variable nomVariable.

L'instruction double (nomVariable) effectue la conversion de la variable spécifiée entre parenthèse vers le type indiqué entre parenthèses. La conversion est alors explicite (demandée par le programmeur). Certaines conversions peuvent être implicites. Il convient d'être prudent lors des conversion car elles peuvent occasionner des pertes de précision (troncature).

La bibliothèque associée à l'en-tête stdint.h permet d'avoir accès à d'autres types d'entiers, parmi lesquels : int8_t, uint8_t, int32_t, uint32_t, int64_t, uint64_t, qui seront détaillés dans le TP.

Entrées et sorties standard

Les échanges d'informations (entrée ou input et sortie ou output) avec l'utilisateur se font par l'intermédiaire de commandes spécifiques. Ces commandes sont accessibles dans la bibliothèque stdio.h.

Sorties

Les sorties permettent d'émettre des données vers les périphériques, dont l'écran. Afin d'écrire une information sur la fenêtre de sortie, la commande printf() sera utilisée.

Affichage d'un message	printf (‘’ Message a afficher’’) ;
Affichage de la valeur de la variable a. Le tableau ci-dessous indique les codes pour les différents formats.	printf (‘’ Variable a : %format’’, a) ;

Formats d'affichage (non exhaustif)

Caractère (char) ou booléen (Bool)	%c		Unsigned long long	%llu
Short, int, signed char	%d		Float ou double (par défaut, 10 décimales)	%f
Unsigned char, unsigned short, unsigned int	%u		Valeur en base 8 (octale)	%o
Long	%ld		Valeur en base 16 (hexadécimale)	%x
Unsigned long	%lu		Chaîne de caractères	%s
Long long	%lld			

Caractères spéciaux

Séquence d'échappement	**Signification**		**Séquence d'échappement**	**Signification**
\n	Saut de ligne		\f	Saut de page
\t	Tabulation horizontale		\v	Tabulation verticale
\r	Retour chariot		\’’	Guillemets
			\\	\

Entrées

La fonction scanf (‘’%format’’, &NomVariable) permet l'acquisition d'une information via le clavier.
La fonction scanf attend que l'utilisateur saisisse un nombre au clavier (fonction bloquante). Une fois l'information tapée et la touche entrée validée, l'information est stockée à l'adresse de la variable (symbole &) dont le nom est mentionné. La variable doit avoir été déclarée au préalable.

Exemple :

```
int main()
{
    int entier ;
    printf (‘’Entrer un entier : \n‘) ;
    scanf(‘’%d’’, &entier) ;
    printf(‘’Valeur de l entier : %d’’, entier) ;
    return 0 ;
}
```

Formats pour le scanf (attention, certains sont différents de celui utilisé pour le printf)

Caractère (char)	%c		Long	%ld
Signed char	%hhd		Unsigned long	%lu, %lx, %lo
Unsigned char	%hhu, %hho, %hhx		Long long	%lld
Short	%hd		Unsigned long long	%llu, %llo, %llx
Unsigned short	%hu, %ho, %hx		Float	%f
Int	%d		Double	%lf
Unsigned int	%u, %x, %o			

Attention : %c ne peut récupérer qu’un caractère, pas un nombre.

Opérateurs de base

Les opérateurs de base sont les suivants :

+	Addition
-	Soustraction
*	Multiplication
/	Division Si les deux opérandes sont des entiers, le résultat sera aussi un entier : 15 / 6 donnera 2 Par contre : 15. / 6 ou 15 / 6. ou 15. / 6. donnera 2.5
%	Reste de la division euclidienne. Seulement entre deux entiers positifs : 15 % 6 = 3