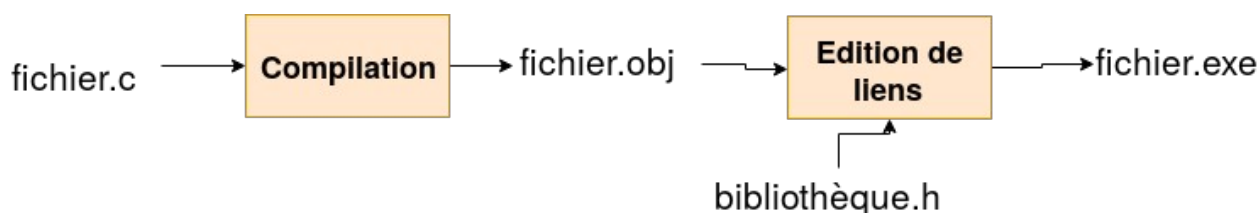


TP2 :Introduction au langage C

Le langage C est un langage de programmation procédural compilé. Il a été développé initialement par Dennis Ritchie en 1972. Le langage C est rapide, portable et disponible sur toutes les plateformes. De plus, il est à usage général : création des programmes console, création des programmes avec interface graphique, création des systèmes d'exploitation, programmation système, programmation des compilateurs, etc.

1 Programme en C

Un programme écrit en C, est un fichier (ou ensemble de fichiers) à extension « .c » ou « cpp ». Afin d'exécuter un programme écrit en C, le fichier contenant le code (la fonction main()) doit être compilé. La compilation génère un fichier qui a souvent l'extension « .obj » (parfois « .o ») ; ce fichier contient la traduction en langage assembleur. Ensuite, la phase « Édition des liens » est exécutée et un fichier exécutable (à l'extension .exe sous Windows) est généré à partir du fichier obj.



Remarque : le fichier exécutable n'est généré qu'après le succès de la phase de compilation. En d'autres termes, aucune erreur ne s'est produite durant la compilation.

2 Structure d'un programme en C

Structure d'un algorithme	Structure d'un programme écrit en C
Algorithme <nom> <partie déclarations des variables> ; Début <partie d'instructions> ; Fin.	#include<stdio.h> int main(){ <partie déclarations des variables> ; <partie d'instructions> ; return 0;}

En langage C, un programme est constitué de 5 parties de base :

1. **Entête (header)** : comprend l'appel des bibliothèques (stdio.h). Une bibliothèque est un fichier à extension .h, contenant des fonctions prédéfinies comme les fonctions d'entrée/sortie définies dans le fichier stdio.h.
2. **main()** : représente le début du programme principal.

3. **Déclaration des variables** : comprend la déclaration des variables à utiliser dans la partie « instructions ».
4. **Corps ou partie d'instructions (body)** : Comprend les différentes instructions qui constituent le programme.
5. **Fin (return)** : représente la fin du programme principal.

3 Mots clés et identifiants

Le langage C est constitué de :

- **Alphabets** : A, ... , Z, a, ... ,z.
- **Chiffres**:0, ... ,9.
- **Caractères spéciaux** : , < > . _ () ; \$: % [] # ? & { } ' " ^ ! * / | - \ ~ +
- **Caractères blancs** : espace, ligne vide,
- **Mots clés** :

C Keywords

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

3.1 Identifiants

Un identifiant est le nom attribué aux entités du langage comme les variables. Il doit être :

- unique (deux entités n'ont jamais le même nom),
- différent des mots clés (une entité n'aura jamais 'main' comme nom par exemple),

- ne doit jamais commencer par un chiffre.

Exemples des identifiants valides : n, max, min, x1, x2, cest_un_identifiant.

Exemple des identifiants non valides : int, char, goto.

4 Variables et constantes

Une variable est un espace où les données sont stockées. Lorsque une variable est déclarée, un espace du mémoire centrale est réservé pour ensuite stocker nos données.

En langage C, une variable doit être déclarée avant son utilisation. La déclaration d'une variable en C se fait comme suit :

```
<type> <identifiant> ;
<type> <identifiant_1>, <identifiant_2>, ... , <identifiant_n> ;
```

Exemples :

```
int x ;
int x, y, z ;
```

Une constante est un espace de stockage où la variable doit être initialisée à sa déclaration et elle ne peut jamais changer de valeur par la suite.

De même, elle doit être déclarée et initialisée, avant l'utiliser. La déclaration d'une constante en C se fait comme suit :

```
const <type> <identifiant> = <valeur> ;
const<type> <identifiant_1> = <valeur_1>, <identifiant_2>, ... , <identifiant_n> = <valeur_n> ;
```

Il est préférable que le nom d'une constante soit écrit en majuscule.

Remarque : N et n sont deux identifiants différents !

5 Types de données

En langage C, il existe 11 types de base. Chaque type est codé sur un nombre précis d'octet. Toutefois, ce dernier peut varier d'un système à un autre (c'est le cas du type 'int' par exemple). Une variable déclarée sous un type précis, peut prendre des valeurs appartenant à un intervalle précis. Le tableau suivant résume les différents types de base du langage C ainsi que leurs tailles et leurs intervalles :

Types	Taille (octet)	L'intervalle
char	1	[-128, 127]
unsigned char	1	[0,255]

int	2 ou 4	[- 32 768, 32 767] ou [- 2 147 483 648, 2 147 483 647]
unsigned int	2 ou 4	[0, 65 535] ou [0, 4 294 967 295]
short	2	[- 32 768, 32 767]
unsigned short	2	[0, 65 535]
long	8	... ?
unsigned long	8	... ?
float	4	
double	8	
long double	10	

6 Opérateurs

6.1 Opérateurs arithmétiques

+	L'addition	/	La division
-	La soustraction	%	Le modulo
*	La multiplication	++/--	incrément/décément

6.2 Opérateurs d'assignation

=	Assignation simple (a=b)	*=	a*=b ; équivalent à a=a*b ;
+=	a+=b ; équivalent à a=a+b ;	/=	a/=b ; équivalent à a=a/b ;
--	a-=b ; équivalent à a=a-b ;	%=	a%=b ; équivalent à a=a %b ;

6.3 Opérateurs de comparaison

==	Égal à	!=	Différent de
>	Supérieur à	>=	Supérieur ou égal à
<	Inférieur à	<=	Inférieur ou égal à

6.4 Opérateurs logiques

Les opérateurs logiques en langage C retournent deux valeurs significatives :

- 1 si l'expression est vraie,
- 0 sinon (l'expression est fausse)

Les opérateurs sont :

&&	'et'
	'ou'

!	'non' (not)
---	-------------

7 Fonctions d'entrée/sortie

printf()	C'est une des fonctions d'affichage de base. Elle permet d'afficher des messages formatés sur l'écran
scanf()	C'est une des fonctions de lecture la plus utilisée. Elle permet de lire des données formatées à partir du clavier.

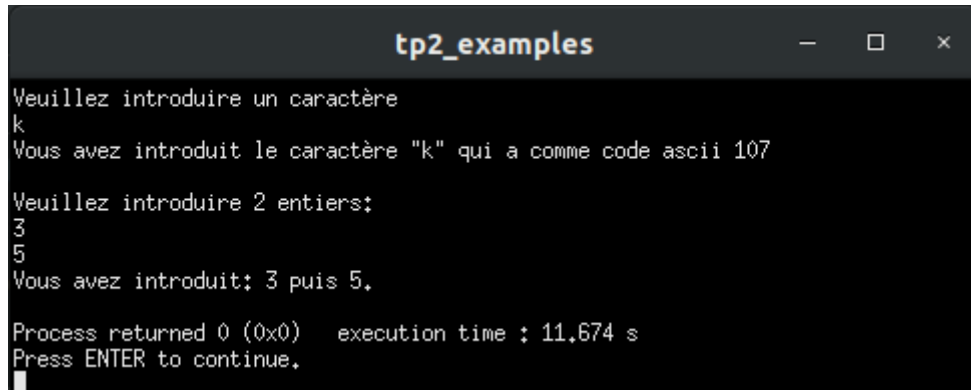
Les fonctions présentées ci-dessus permettent d'afficher ou de lire des valeurs de différents types à savoir les entiers, les réels, les caractères et chaînes de caractères. Pour spécifier le type des valeurs à afficher, des spécificateurs de format sont utilisés et durant l'exécution ces derniers vont être remplacés par les valeurs souhaitées. Le tableau ci-dessous résume quelques spécificateurs de format avec leurs types adéquats :

Spécificateur	Type	Spécificateur	Type	Spécificateur	Type
%d	int	%c	char	%f	float
%u	unsigned int	%lf	double		

Exemple :

```
main.c ✕
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int a, b;
7      char c;
8
9      printf("Veuillez introduire un caractère\n");
10     scanf("%c", &c); // le & est obligatoire
11     printf("Vous avez introduit le caractère \"%c\" qui a comme code ascii %d\n\n", c, c); // affichage d'un caractère
12                                     // et son code ascii
13
14     printf("Veuillez introduire 2 entiers:\n");
15     scanf("%d%d", &a, &b); // lecture de deux valeurs de type entier.
16     printf("Vous avez introduit: %d puis %d.\n", a, b); // affichage de deux valeurs de type entier.
17     return 0;
18 }
```

Exécution :



```
tp2_examples
Veillez introduire un caractère
k
Vous avez introduit le caractère "k" qui a comme code ascii 107

Veillez introduire 2 entiers:
3
5
Vous avez introduit: 3 puis 5.

Process returned 0 (0x0)   execution time : 11.674 s
Press ENTER to continue.
```

Exercices

1. Écrire un programme en C qui permet de faire la permutation entre deux valeurs saisies par l'utilisateur puis les afficher (proposer deux solutions).
2. En utilisant la fonction *sizeof()*, écrire un programme qui affiche la taille des variables de type : int, float, char et double.
3. Que fait ce programme ?

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

int main(int argc, char** argv)
{

    printf("CHAR_BIT   : %d\n", CHAR_BIT);
    printf("CHAR_MAX   : %d\n", CHAR_MAX);
    printf("CHAR_MIN   : %d\n", CHAR_MIN);

    return 0;
}
```

Modifier le programme pour afficher le min et le max des types suivants : int, long, short, unsigned int, unsigned short, unsigned long (voir l'annexe).

4. À l'aide de la bibliothèque *math.h*, calculer : a^2 , racine de a et a puissance b . Utiliser les fonctions *pow(x,y)* et *sqrt(x)*.

Que donnent les fonctions suivantes : *ceil(x)*, *floor(x)*, *round(x)*, *trunc(x)*, *abs(x)*, *fabs(x)*, sachant que x est un réel.

5. Écrire un programme qui fonctionne de la même façon que *abs(x)* vue précédemment, sachant que x est un réel positif.

Modifier le programme si x est un réel négatif.

6. Écrire un programme qui fonctionne de la même façon que `ceil(x)`.
7. Écrire un programme qui fonctionne de la même façon que `floor(x)`.
8. Écrire un programme qui permet d'afficher si : a est égale à b, a est inférieur à b, a est différent de b.
9. Écrire un programme qui permet de voir si un nombre est impair.

Annexe

name	expresses	value*
CHAR_BIT	Number of bits in a char object (byte)	8 or greater*
SCHAR_MIN	Minimum value for an object of type signed char	-127 (-2^7+1) or less*
SCHAR_MAX	Maximum value for an object of type signed char	127 (2^7-1) or greater*
UCHAR_MAX	Maximum value for an object of type unsigned char	255 (2^8-1) or greater*
CHAR_MIN	Minimum value for an object of type char	either SCHAR_MIN or 0
CHAR_MAX	Maximum value for an object of type char	either SCHAR_MAX or UCHAR_MAX
MB_LEN_MAX	Maximum number of bytes in a multibyte character, for any locale	1 or greater*
SHRT_MIN	Minimum value for an object of type short int	-32767 ($-2^{15}+1$) or less*
SHRT_MAX	Maximum value for an object of type short int	32767 ($2^{15}-1$) or greater*
USHRT_MAX	Maximum value for an object of type unsigned short int	65535 ($2^{16}-1$) or greater*
INT_MIN	Minimum value for an object of type int	-32767 ($-2^{15}+1$) or less*
INT_MAX	Maximum value for an object of type int	32767 ($2^{15}-1$) or greater*
UINT_MAX	Maximum value for an object of type unsigned int	65535 ($2^{16}-1$) or greater*
LONG_MIN	Minimum value for an object of type long int	-2147483647 ($-2^{31}+1$) or less*
LONG_MAX	Maximum value for an object of type long int	2147483647 ($2^{31}-1$) or greater*
ULONG_MAX	Maximum value for an object of type unsigned long int	4294967295 ($2^{32}-1$) or greater*
LLONG_MIN	Minimum value for an object of type long long int	-9223372036854775807 ($-2^{63}+1$) or less*
LLONG_MAX	Maximum value for an object of type long long int	9223372036854775807 ($2^{63}-1$) or greater*
ULLONG_MAX	Maximum value for an object of type unsigned long long int	18446744073709551615 ($2^{64}-1$) or greater*

Références

- <https://www.geeksforgeeks.org/c-language-set-1-introduction/>
- <https://www.tutorialspoint.com/cprogramming/index.htm>
- <http://www.cplusplus.com/reference/climits/>