

LEÇON 1 : Les Concepts généraux du Langage C






Table des matières

Objectifs	5
I - Pré requis	7
II - INTRODUCTION	9
III - I- LA STRUCTURE D'UN PROGRAMME EN LANGAGE C	11
IV - Exercice	13
V - Exercice	15
VI - II- LES TYPES DE DONNÉES	17
VII - Exercice	19
VIII - Exercice	21
IX - III- LES VARIABLES ET CONSTANTES	23
X - Exercice	25
XI - Exercice	27
XII - IV- LES OPÉRATEURS	29
XIII - Exercice	31
XIV - Exercice	33
XV - V- LES ENTRÉES / SORTIES	35
XVI - Exercice	37

XVII - Exercice	39
XVIII - VI- LES SÉQUENCES D'ÉCHAPPEMENT	41
XIX - Exercice	43
XX - Exercice	45
XXI - VII- LES INSTRUCTIONS ET COMMENTAIRES	47
XXII - Exercice	49
XXIII - Exercice	51
XXIV - Exercice	53
XXV - Exercice	55
XXVI - VIII- EXEMPLE DE PROGRAMME EN LANGAGE C	57
Solution des exercices	59



Objectifs

Ce cours devra vous permettre de:

- **Connaître** la structure de base d'un programme en langage C
- **Connaître** les types de données en langage c
- **Compiler** et **exécuter** un programme en langage C



Pré requis

I

Avant de suivre cette leçon, vous devez avoir pour acquis la connaissance de l'algorithme





INTRODUCTION

II

Le langage est un langage structuré assez proche du langage machine destiné à des applications de contrôle de processus.

Un programme en langage C est compilé avant d'être exécuté. La compilation consiste à transcrire le programme dans un langage compréhensible par la machine (destiné au microprocesseur). Le compilateur indique les erreurs de syntaxe qui pourraient éventuellement survenir dans le programme tout en ignorant les fonctions bibliothèques encore appelées directives.

I- LA STRUCTURE D'UN PROGRAMME EN LANGAGE C



Définition

Un programme simple se compose de plusieurs parties :

- Des directives de pré-compilation
- Les différentes fonctions éventuelles dont l'une s'appelle obligatoirement `main()` et à l'intérieure de laquelle est défini le programme principal
- La déclaration de toutes les variables et constantes
- Les différentes instructions du programme principal

1- LES DIRECTIVES DE PRE-COMPILATION

Encore appelées fonctions bibliothèques, les directives commencent par le symbole `#`.

ci-dessous quelques exemples de bibliothèques :

- **`#include <stdio.h>`** : elle permet d'utiliser les fonctions `printf()` et `scanf()`.
- **`#include <math.h>`** : elle permet d'utiliser les fonctions mathématiques.
- **`#include <conio.h>`** : elle permet de manipuler la console windows (par exemple effacer l'écran, changer la couleur de police de console etc.).
- **`#include <string.h>`** : elle permet l'utilisation des fonctions liées aux chaînes de caractères.

2- LA FONCTION `main()`

Utilisée pour définir le programme principal, cette fonction commence par une accolade ouvrante «`{`» et se termine par une accolade fermante «`}`».



Exemple

```
main()
{
    Corps du programme
}
```



Exercice

IV

[Solution n°1 p 59]

Un programme en langage C est:

- | | |
|-----------------------|--|
| <input type="radio"/> | Exécuté avant d'être compilé |
| <input type="radio"/> | Ne peut être exécuté sans être compilé |
| | N'a pas forcément besoin d'être compilé avant d'être exécuté |



Exercice



[Solution n°2 p 59]

La directive qui assure l'utilisation des fonctions cosinus et sinus est:

	Stdio.h
	Conio.h
	Math.h

II- LES TYPES DE DONNÉES



Définition

Il faut déclarer le type de toutes les variables et de toutes les fonctions qui indique à la fois l'intervalle de définition et les opérations licites.

1- LES TYPES DE DONNÉES PREDEFINIES

Type	Signification	Taille (bits)	Valeurs limites
int	entier	16	-32768 à +32768
short	entier	16	-32768 à +32768
long	entier	32	-2 147 483 648 à +2 147 483 648
char	caractère	8	-128 à +127
float	réel		-10 E-37 à +10 E+38
double	réel		-10 E-307 à +10 E+308

2- LES FORMATS D'AFFICHAGE

Ils permettent de définir le format dans lequel les données seront soit affichées ou récupérées. Ainsi, nous distinguons les formats suivants :

- **%d** ou **%i** pour les entiers
- **%f** pour les réels
- **%c** pour les caractères
- **%s** pour les chaînes de caractères
- **%x** pour le code hexadécimal

Il précède toujours la variable à laquelle il est rattaché.



Exercice

VII

[Solution n°3 p 59]

Lors du formatage des données, %s est utilisé pour :

	Une chaîne de caractère
	Une valeur de type réel
	Un caractère



Exercice

VIII

[Solution n°4 p 59]

Quel est le formatage d'un entier ?

	%f
	%c
	%d

III- LES VARIABLES ET CONSTANTES

1- LES VARIABLES

Les variables sont des adresses de la mémoire vive représentées par des identificateurs. Une variable peut à chaque fois changer de valeur mais ne reçoit qu'une seule valeur à la fois.



Syntaxe : Syntaxe de définition d'une variable :

TYPE identificateur1[, identificateur2, ...];

Ou encore **TYPE** identificateur[Taille] pour les variables de type chaîne de caractères

2- LES CONSTANTES

Contrairement à la variable, une constante ne change pas de valeur durant le déroulement du programme. Il existe deux méthodes de définition d'une constante (soit avec **Const** ou la **#define**).



Syntaxe : Sa syntaxe est la suivante :

Méthode 1: **Const** Type identificateur = valeur;

Méthode 2: **#define** identificateur = valeur;



Exercice

X

[Solution n°5 p 60]

Quelle syntaxe de déclaration de variable est-elle correcte ?

	A, B : int
	int : A,B
	int A,B



Exercice

XI

[Solution n°6 p 60]

Quelle syntaxe de déclaration de constante est-elle correcte ?

	<code>A =10 ;</code>
	<code>#define A = 10 ;</code>
	<code>const A : 10 ;</code>



IV- LES OPÉRATEURS

XII

Définition

Comme dans tous les langages, le langage C se sert des opérateurs pour la manipulation des opérandes. Ceci dit, nous distinguons plusieurs types d'opérateurs à savoir :

1- LES OPÉRATEURS ARITHMÉTIQUES

+	L'addition
-	La soustraction
/	La division
*	Le produit
%	Le modulo
++	L'incrémentement
--	La décrémentation

2- LES OPÉRATEURS DE COMPARAISON (OU RELATIONNELS)

==	L'égalité de valeur
>	Supérieur
<	Inférieur
>=	Supérieur ou égal
<=	Inférieur ou égal
!=	La différence

3- LES OPÉRATEURS D'ASSIGNATION

=	L'affectation
+=	La somme et affectation
-=	La soustraction et l'affectation
/=	La division et l'affectation
*=	Le produit et l'affectation

IV- LES OPÉRATEURS

%=	Le modulo et l'affectation
----	----------------------------

4- LES OPÉRATEURS LOGIQUES

!	Le NON logique
&&	Le ET logique
	Le OU logique





Exercice

XIII

[Solution n°7 p 60]

A quoi sert le symbole || ?

	NON logique
	OU logique
	ET logique





Exercice

XIV

[Solution n°8 p 60]

A quoi sert le symbole % ?

	Le modulo et l'affectation
	L'incrémentation
	Le modulo

V- LES ENTRÉES / SORTIES

XV

1- LES ENTRÉES DE DONNEES

Les entées de données sont effectuées à l'aide des fonctions **gets** (pour les chaînes de caractères) ou **scanf**(généralement). La fonction **scanf** prend deux paramètres, le premier qui spécifie le formatage et le second, l'adresse mémoire de la variable (généralement précédée d'un & lorsqu'il s'agit de données autre que les caractères)



Exemple

```
scanf("%d",&v1); //cela signifie que la valeur saisie par l'utilisateur est de type ENTIER et que cette variable sera récupérée par la variable v1.
```

2- LES SORTIES DE DONNÉES

L'affichage de données s'effectue à l'aide des fonctions **puts** (pour les chaînes de caractères) ou **printf**(généralement). Lorsque la fonction **printf** doit afficher le contenu d'une variable, elle prend deux paramètres, le premier qui spécifie le format de la variable et le second, la variable pour laquelle, la valeur doit être retournée.

Exemple

1. `printf("Mon premier code en langage C");` //Cette instruction affichera: Mon premier code en langage C.
2. `Val1=5; Printf("La valeur récupérée est %d",val1);`//cette instruction affichera: La valeur récupérée est 5.



Exercice

XVI

[Solution n°9 p 60]

Quelle syntaxe est correcte pour récupérer une valeur de type réel à l'aide de la variable v1

<input type="checkbox"/>	<code>scanf("%d",&v1);</code>
<input type="checkbox"/>	<code>scanf("%f",v1);</code>
<input type="checkbox"/>	<code>scanf("%f",&v1);</code>



Exercice

XVII

[Solution n°10 p 60]

Pour retourner la valeur de la variable val1, laquelle de ces syntaxes est correcte ?

	<code>val1=5; printf("La valeur reçue est val1");</code>
	<code>val1=5; printf("La valeur reçue est val1",%d);</code>
	<code>val1=5; printf("La valeur reçue est %d",val1);</code>



VI- LES SÉQUENCES D'ÉCHAPPEMENT

XVII I

\b	Retour arrière (backspace)
\f	Saut de page
\n	Saut de ligne
\r	Retour chariot
\t	Tabulation horizontale
\v	Tabulation verticale
\\	\
\'	'
\"	"



Exercice

XIX

[Solution n°11 p 61]

```
7. main()
{
printf("UVC\nCocody");
}
```

Que retourne cette instruction ?

	UVC Cocody
	UVC_Cocody
	UVC_\bCocody



Exercice

XX

[Solution n°12 p 61]

7. *main()*

```
{  
printf("UVC I \bCocody");  
}
```

Que retourne cette instruction ?

	UVC I Cocody
	UVCICocody
	UVC I \bCocody



VII- LES INSTRUCTIONS ET COMMENTAIRES

1- LES INSTRUCTIONS

Les instructions permettent de définir les différentes tâches à exécuter. Plusieurs instructions définies constituent un bloc d'instructions. Il est défini entre les accolades, il peut s'agir d'un bloc d'instructions simples (instructions se terminant par un point-virgule) ou d'un bloc d'instructions structurées (choix, boucle etc.).

2- LES COMMENTAIRES

Les commentaires dans un programme assurent une compréhension du code qui s'y trouve. Il permet de décrire les différentes instructions écrites ou à écrire. Les commentaires ne sont pas interprétés donc n'ont aucun effet sur l'exécution d'un programme. En langage C, un commentaire peut se définir de deux manières :

- Sur une seule ligne : `// commentaire`
- Sur plusieurs lignes : `/* commentaire */`



Exercice

XXII

[Solution n°13 p 61]

1. *main()*

```
{  
int val1, val2;  
val1=5;  
val2=val1++;  
}
```

Que valent *val2* et *val1* à la fin du programme ?

	val2 vaut 5 et val1 vaut 6
	val2 vaut 6 et val1 vaut 5
	val2 vaut 6 et val1 vaut 6

Exercice

XXII

I

[Solution n°14 p 61]

12. *main()*

```
{  
int val1, val2;  
val1=5;  
val2=++val1;  
}
```

Que valent val2 et val1 à la fin du programme ?

	val2 vaut 5 et val1 vaut 6
	val2 vaut 6 et val1 vaut 5
	val2 vaut 6 et val1 vaut 6

Exercice

XXI

V

[Solution n°15 p 61]

```
main()
{
  int val1, val2, val3, val4;
  val1=0;
  val2=1;
  val3=(val1 && val2);
  val4=!val3 || val1;
  printf("%d", val4);
}
```

Que vaut val4 à la fin du programme ?

1
0
3



Exercice

XXV

[Solution n°16 p 61]

```
1. main()
{
int val1, val2, val3, val4;
val1=0;
val2=1;
val3=(val1 && val2);
val4=(!val3 || val1)&& (val2 && !val3);
printf("%d", val4);
}
Que vaut val4 à la fin du programme ?
```

	Aucun résultat généré
	0
	1

VIII- EXEMPLE DE PROGRAMME EN LANGAGE C

XXV
I

1- ÉNONCE

Écrire un programme en langage C qui calcul la moyenne de deux nombres entiers saisis par l'utilisateur.

2- RÉOLUTION

```
#include<stdio.h> // Appel de la bibliothèques pour utiliser les fonctions printf() et scanf().
#include<conio.h> // Appel de la bibliothèques pour manipuler la console windows
main()
{
float v1,v2,moy; // Déclaration des variables v1,v2,moy de type float (réel)
printf("Entrez le premier nombre svp!\n"); // affiche à l'écran le message : Entrez le premier nombre svp! et retourne à la ligne
scanf("%f",&v1); // Permet de récupérer la valeur saisie exemple on entre 12 .
printf("Entrez le deuxième nombre svp!\n"); // affiche à l'écran le message : Entrez le deuxième nombre svp! et retourne à la ligne
scanf("%f",&v2); //Permet de récupérer la valeur saisie exemple on entre 18 .
moy=(v1+v2)/2; // moy = (18 +12)/2
printf("\n La moyenne de %f et %f donne %f",v1,v2,moy); // affichera : La moyenne de 12 et 18 donne 15
}
```



Solution des exercices

> Solution n°1 (exercice p. 13)

- | | |
|----------------------------------|--|
| <input type="radio"/> | Exécuté avant d'être compilé |
| <input checked="" type="radio"/> | Ne peut être exécuté sans être compilé |
| <input type="radio"/> | N'a pas forcément besoin d'être compilé avant d'être exécuté |

> Solution n°2 (exercice p. 15)

- | | |
|----------------------------------|---------|
| | Stdio.h |
| | Conio.h |
| <input checked="" type="radio"/> | Math.h |

> Solution n°3 (exercice p. 19)

- | | |
|--|-------------------------|
| | Une chaîne de caractère |
| | Une valeur de type réel |
| | Un caractère |

> Solution n°4 (exercice p. 21)

- | | |
|--|----|
| | %f |
| | %c |
| | %d |

> Solution n°5 (exercice p. 25)

A, B : int
int : A,B
int A,B

> Solution n°6 (exercice p. 27)

A =10 ;
#define A = 10 ;
const A : 10 ;

> Solution n°7 (exercice p. 31)

NON logique
OU logique
ET logique

> Solution n°8 (exercice p. 33)

Le modulo et l'affectation
L'incrémentation
Le modulo

> Solution n°9 (exercice p. 37)

scanf("%d",&v1);
scanf("%f",v1);
scanf("%f",&v1);

> Solution n°10 (exercice p. 39)

val1=5; printf("La valeur reçue est val1");
val1=5; printf("La valeur reçue est val1",%d);
val1=5; printf("La valeur reçue est %d",val1);

> Solution n°11 (exercice p. 43)

Solution des exercices

	UVCi Cocody
	UVCi_Cocody
	UVCi_ \bCocody

> Solution n°12 (*exercice p. 45*)

	UVCi Cocody
	UVCiCocody
	UVCi \bCocody

> Solution n°13 (*exercice p. 49*)

	val2 vaut 5 et val1 vaut 6
	val2 vaut 6 et val1 vaut 5
	val2 vaut 6 et val1 vaut 6

> Solution n°14 (*exercice p. 51*)

	val2 vaut 5 et val1 vaut 6
	val2 vaut 6 et val1 vaut 5
	val2 vaut 6 et val1 vaut 6

> Solution n°15 (*exercice p. 53*)

	1
	0
	3

> Solution n°16 (*exercice p. 55*)

	Aucun résultat généré
	0
	1

