Les bases de la programmation en langage C

L'écriture de programmes en C nécessite de comprendre un grand nombre de concepts tous liés les uns aux autres. Pour cette raison, il n'est pas possible de tous les étudier en détails d'entrée. Dans cette partie, on se contentera donc d'introduire les notions essentielles de façon superficielle, afin de pouvoir progresser dans le cours. La plupart de ces notions seront ensuite revues plus tard, de façon plus approfondie.

1. Historique

Le langage C a été conçu en 1972 par Dennis Richie et Ken Thompson. En 1978, Brian Kernigham et dennis Richie publient la définition classique du C dans le livre « The C Programming Language ». Le C devenant de plus en plus populaire dans les années 80. En 1983, l'ANSI (American National Standards Institute) décida de normaliser le langage, ce travail s'acheva en 1989 par la définition de la norme ANSI C.

Le langage C a été conçu pour l'écriture de systèmes d'exploitation et du logiciel de base (90% du noyau du système UNIX est écrit en C, le compilateur C lui-même est écrit en grande partie en langage C, de nombreux logiciels du domaine des ordinateurs personnels, tels que Microsoft Word ou Excel sont écrits à partir du langage C ou en son successeur orienté objet C++.

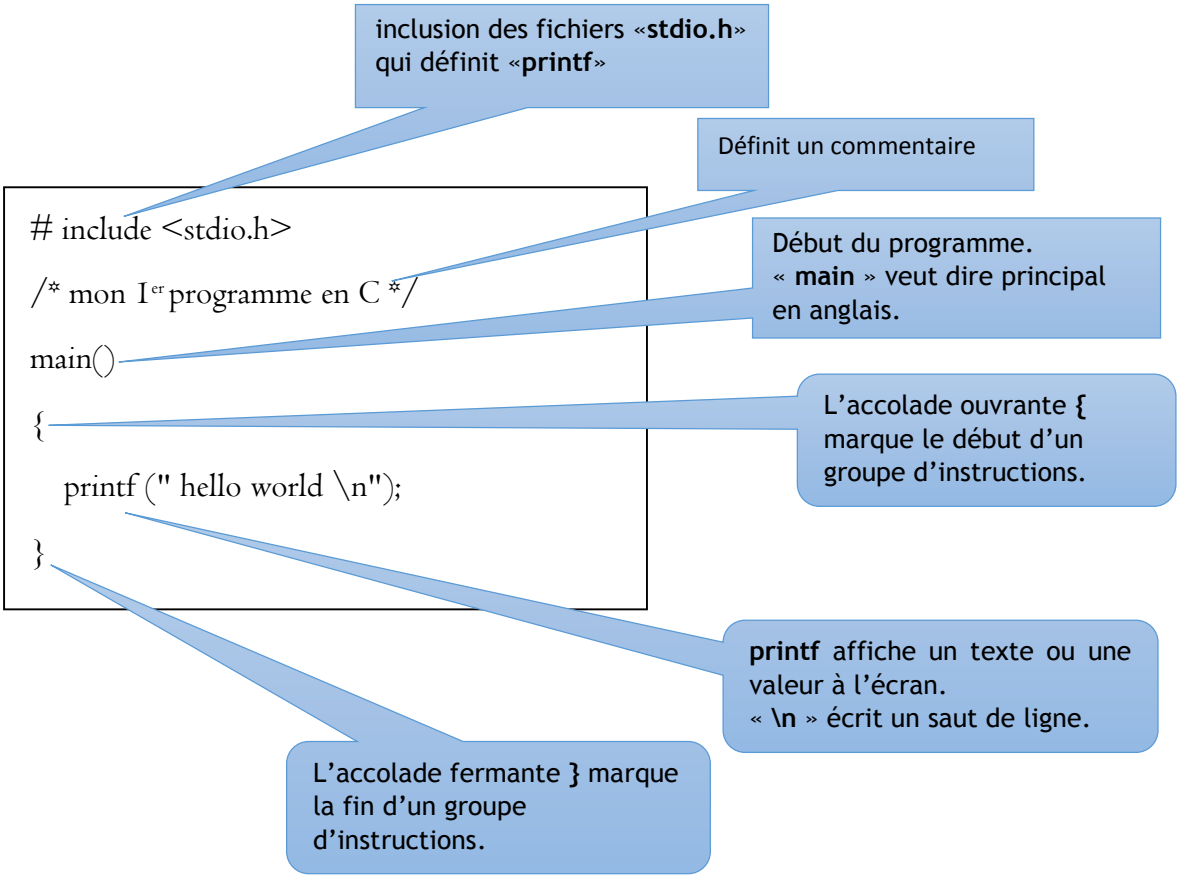
2. Structure d'un programme en C

Une instruction est suivie d'un point-virgule, plusieurs instructions peuvent être rassemblées par des accolades { } pour former un bloc.

Un programme C se présente de la façon suivante :

```
[ directives au préprocesseur]
[ déclarations de variables externes]
[ fonctions secondaires]
main()
{
déclarations de variables internes
instructions
}
```

Exemple de programme simple :



- **Directives au préprocesseur:**

Selon ce que l'on souhaite faire dans notre programme, on peut avoir besoin de différentes fonctions.

Celles-ci sont disponibles dans des bibliothèques. Dans d'autres documents, cette étape est appelée « importation des bibliothèques ». Par exemple :

 # include <stdio.h> : permet d'utiliser des fonctions d'affichage « printf() » et de saisie « scanf() ».

 # include <math.h> : permet d'utiliser les fonctions mathématiques

- **Commentaires** : Un commentaire est une séquence de caractères ignorée par le compilateur, on s'en sert pour expliquer des portions de code. On délimite un commentaire par /* et */.

/* votre commentaire */

Par exemple : /* mon 1er programme en C */

2.1. Types prédéfinis

Les types de base en langage C sont les caractères, les entiers et les flottants (nombres réels). Ils sont désignés par les mots clefs suivants :

- **Char** : désigne un objet de caractère
- **Int, short, long** : désigne un entier naturel
- **Float, double** : désigne un réel
- **Unsigned** : signifie que l'entier n'a pas de signe

2.2. Variables

Une variable est un emplacement de la mémoire dans lequel est stockée une valeur. Chaque variable porte un nom et c'est ce nom qui sert à identifier l'emplacement de la mémoire représenté par cette variable. Pour utiliser une variable, la première étape est la déclaration.

- **Déclaration :** Déclarer une variable, c'est prévenir le compilateur qu'un nom va être utilisé pour désigner un emplacement de la mémoire. En C, on déclare les variables juste après l'accolade suivant main(). On place les instructions à exécuter à la suite de la déclaration de variables.

  La forme générale d'une déclaration est la suivante :

NomDuType NomVariable_1, NomVariable_2,…, NomVariable_n ;

Exemple : int x, y ; float z ; char c ;

Remarque : Il est important de noter qu'en C la déclaration d'une variable n'entraîne pas son initialisation et le compilateur signalera une erreur si le programmeur la référence sans l'avoir initialisée.

L'initialisation d'une variable peut se faire au moment de sa déclaration de la manière suivante :

```
NomDuType NomVariable = Valeur ;
```

Exemple : int x = 5 ; char c= 'a' ; float =3.12 ;

2.3. Les constantes

Une constante est une valeur qui apparaît littéralement dans le code source d'un programme, le type de la constante étant déterminé par la façon dont les constantes sont écrites. Les constantes peuvent être de trois types : entier, flottant, caractère.

```
const NomDuType NomVariable = Valeur ;
```

Exemple : const float Pi = 3.14 ;

- **Syntaxe correcte pour le nom des variables**

Identificateurs corrects	Identificateurs incorrects	Pourquoi ?
nom1	1nom	Ne peut commencer par un chiffre
nom_2	nom.2	Les points ne sont pas autorisés
_nom_3	-nom-3	Les tirets ne sont pas autorisés
nom_de_variable	Nom de variable	Les espaces ne sont pas autorisés
deuxieme_choix	deuxième_choix	Les caractères accentués ne sont pas autorisés
mot_francais	mot_français	Les cédilles ne sont pas autorisées

3. Les opérateurs

3.1. L'affectation

Si on souhaite affecter à la variable v une valeur, on utilise l'opérateur (=). Par exemple,

```
main () {
int i ;
i=5 ;
}
```

Ce programme déclare une variable de type entier que l'on appelle v, puis lui affecte la valeur 5.

3.2. Les opérateurs arithmétiques

Les opérations arithmétiques disponibles sont l'addition (+), la soustraction (-), la multiplication (*), la division entière dans l'ensemble des entiers relatifs (quotient : /, reste : %).

Pour la puissance, il faut utiliser pow(s,y) de la librairie math.h pour calculer x^y

```
main ( )
{
int v , w , z ;
v = 5 ;
w = v + 1 ;
z = v + w / 2 ;
v = z % 3 ;
v = v * 2 ;
}
```

Si les deux opérandes sont de type entier, l'opérateur produira une division entière. Par contre, il délivrera une valeur flottante dès que l'un des opérandes est un flottant.

Exemple :
float x;
x = 3 / 2;
affecte à x la valeur 1. Par contre
x = 3 / 2.;
affecte à x la valeur 1.5.

3.3. Les opérateurs relationnels

Comme tout langage, C permet de comparer des expressions à l'aide d'opérateurs classiques de comparaison.

OPÉRATEUR	SIGNIFICATION
<	inférieur à
<=	inférieur ou égal à
>	supérieur à
>=	supérieur ou égal à
==	égal à
! =	Différent de

Exemple : 2 * a > b + 5

Le résultat de la comparaison est :

• 0 si le résultat de la comparaison est faux,

• 1 si le résultat de la comparaison est vrai.

3.4. Les opérateurs logiques

C dispose de trois opérateurs logiques classiques : **et** (noté &&), **ou** (noté ||) et **non** (noté !). Par exemple :

● **(a<b) && (c<d) :** prend la valeur 1 (vrai) si les deux expressions a<b et c<d sont toutes deux vraies (de valeur 1), et prend la valeur 0 (faux) dans le cas contraire.

● **(a<b) || (c<d) :** prend la valeur 1 (vrai) si l'une au moins des deux conditions a<b et c<d est vraie (de valeur 1), et prend la valeur 0 (faux) dans le cas contraire.

● **! (a<b) :** prend la valeur 1 (vrai) si la condition a<b est fausse (de valeur 0) et prend la valeur 0 (faux) dans le cas contraire. Cette expression est équivalente à : a>=b.

3.5. Les opérateurs d'incrémentation et de décrémentation

Dans des programmes écrits dans un langage autre que C, on rencontre souvent des expressions (ou des instructions) telles que :

```
i = i + 1
n = n - 1
```

Qui incrémentent ou qui décrémentent de 1 la valeur d'une variable. En C, ces actions peuvent être réalisées par des opérateurs « unaires » , ainsi nous obtenons :

```
++i
```

A pour effet d'incrémenter de 1 la valeur de i, et sa valeur est celle de i **après incrémentation**. La valeur de cette expression est celle de i après incrémentation. Ainsi, si la valeur de i est 5, l'expression :

```
n = ++i - 5
```

Affectera à i la valeur 6 et à n la valeur 1.

En revanche, lorsque cet opérateur est placé *après* la *lvalue* sur laquelle il porte, la valeur de l'expression correspondante est celle de la variable **avant incrémentation**.

Ainsi, si i vaut 5, l'expression :

n = i++ - 5

Affectera à i la valeur 6 et à n la valeur 0 (car ici la valeur de l'expression i++ est 5).

On dit que ++ est :

- un opérateur de **pré incrémentation** lorsqu'il est placé à gauche de la *lvalue* sur laquelle il porte,
- un opérateur de **post incrémentation** lorsqu'il est placé à droite de la *lvalue* sur laquelle il porte.

De la même manière, il existe un opérateur de décrémentation noté -- qui, suivant les cas, sera :

- un opérateur de **prédécrémentation** lorsqu'il est placé à gauche de la *lvalue* sur laquelle il porte,
- un opérateur de **postdécrémentation** lorsqu'il est placé à droite de la *lvalue* sur laquelle il porte.

3.6. Les opérateurs d'affectation composés

+= -= *= /= %=

L'expression : expr1 op = expr2

Est équivalente à :

expr1= expr1 op expr2

4. Les fonctions d'entrées/sorties

Deux fonctions d'entrées sorties les plus utilisées dans le langage C, qui sont : printf et scanf

4.1. Printf ()

Est une fonction d'impression formatée. Syntaxe :

Printf("chaine de contrôle", exp1,…., expn) ;

La chaine de contrôle contient le texte à afficher et les spécifications de format correspondant à chaque expression de la liste. Elles sont introduites par le caractère % suivi de caractère désignant le format d'impression.

Exemple :

%.12f : spécifie qu'un flottant sera affiché avec 12 chiffres après la virgule

Traduisons maintenant l'instruction *Afficher variable* en C. Cette instruction permet d'afficher la valeur d'une variable.

printf ("%d" , <variable >);

Cette instruction affiche la valeur contenue dans la variable. Nous avons étendu, en algorithmique, l'instruction Afficher en intercalant des valeurs de variables entre les messages affichés. Il est possible de faire de même en C :

printf (" la valeur de la variable v est %d" , v) ;

Cette instruction substitue à %d la valeur de la variable v. Il est possible de l'étendre à volonté :

printf (" l e s v a l e u r s des v a r i a b l e s x , y e t z sont %d , %d e t %d" , x , y , z) ;

%ld	long int	décimale signée
%f	double	décimale virgule fixe
%c	unsigned char	caractère
%s	char*	chaîne de caractères

4.2. Scanf ()

est une fonction qui permet de saisir des données au clavier et les stocker aux adresses spécifiées par les arguments de la fonction.

Syntaxe :

Scanf(" chaine de contrôle", &argument1,……..&argumentn)

La chaine de contrôle indique le format dans lequel les données lues sont converties. Comme pour printf, les conventions de format sont spécifiées par un caractère précédé du signe%.

On peut fixer le nombre de caractères de la donnée à lire :

%3s : pour une chaine de 3 caractères

Traduisons en C l'instruction Saisir <variable> que nous avons vu en algorithmique. Pour récupérer la saisie d'un utilisateur et la placer dans une variable <variable>, on utilise l'instruction suivante :

scanf ("%d" , &<variable >);

scanf suspend l'exécution du programme jusqu'à ce que l'utilisateur ait saisi une valeur et presse la touche return. La valeur saisie est alors affectée à la variable <variable>. N'oubliez surtout pas le &, sinon des choses absurdes pourraient se produire

%ld	long int	décimale signée
%f	double	décimale virgule fixe
%c	unsigned char	caractère
%s	char*	chaîne de caractères

Exercices :

Ecrire un programme qui fait l'addition de deux nombres a et b et affiche leur résultats c.

Ecrire un programme en langage C qui affiche la moyenne de deux valeurs