Langage C - Memo

Chapitre 1: Introduction à la programmation

1.1) Qu'est ce qu'un programme?

Un programme informatique réalise trois type de tâche

1. **Lecture des données** → la source peut être un utilisateur, une base de données, stockée sur un support type disque dur, ou même la sortie d'un autre programme.
2. **Calculs** → C'est la séquence d'instruction écrite par le développeur. C'est ce qu'on appelle l'algorithme.
3. **Renvoyer des données** → Le retour peut être affiché, stocké en base ou sous forme de fichiers ou renvoyer vers un autre programme.

1.2) Un programme en C

1. Rédiger les sources dans un document texte en respectant la syntaxe du langage C. Si un simple outil tel que notepad ou gedit peut suffire, on préférera un IDE (*integrated development environment*), c'est à dire un logiciel permettant de facilité la rédaction de code. Pour le C, le logiciel s'appelle Codeblocks (il est disponible sur Linux, Mac et Windows)
2. Une fois le code terminé, on enregistre le fichier avec une extension c. Exemple : monprogramme.c. On verra plus tard que certains fichiers prendront également l'extension .h
3. On compile alors le programme pour le rendre compréhensible par l'ordinateur. Cette étape permettra également de vérifier que la syntaxe C est respectée (oubli d'un <;> en fin de ligne, etc.). Cela va générer un fichier de liens .o. Exemple monprogramme.o
   En ligne de commande, cette étape se fait de la manière suivante:
   - `gcc -c monprogramme.c -o monprogramme.o`
   - Le <gcc> est ce qu'on appelle le compilateur; le -c indique que l'on va compiler le fichier monprogramme.c et le -o indique que la sortie de la commande sur le fichier monprogramme.o
4. L'étape suivante est l'édition de le liens. Cela permettra de générer un fichier binaire exécutable. Sous Windows, ce sont les fichiers .exe, sous Linux et Mac, il n'y a pas forcément d'extension.
   En ligne de commande, cette étape se fait de la manière suivante:
   - `gcc monprogramme.o -o monprogramme.exe`

- Toujours avec le compilateur gcc, on reprend le fichier .o et on génère le fichier .exe à l'aide le l'option -o (dans les deux cas, -o est le raccourcis pour –output)

5. On peut maintenant exécuter le programme dans un environnement approprié (le même que celui de la compilation)

Différence entre compilation et interprétation

Le langage C est un langage compilé à la différence de Python qui est un langage interprété. La différence réside dans les conditions d'exécution du programme.
Un programme compilé pourra s'exécuter sur une machine sans avoir besoin d'outils supplémentaire.
Un langage interprété aura besoin d'un interpréteur pour être traduit et être utilisé sur une machine. Exemple, pour lancer un programme Python, on doit utiliser la commande <python> suivit du nom du script.
L'avantage d'un langage interprété est qu'il peut s'exécuter sur n'importe quel machine possédant l'interpréteur.
A l'inverse, un programme compilé ne pourra s'exécuter que sur un environnement similaire à celui où à eu lieu la compilation.
On peut résumer les avantages et inconvénient dans le tableau suivant

	Avantage	Inconvénient
Langages compilés	Indépendant (+léger)	Ne fonctionne que sur des environnements identique à celui de la compilation
Langages interprétés	Code portable sur n'importe quel environnement possédant un interpréteur	Nécessite un interpréteur (+gourmand en ressource)

1.2.1 Analyse d'un programme simple en C

```
1  #include <stdio.h>
2  /*Un commentaire sur
3  plusieurs lignes */
4  int main() {
5     printf("Hello world!");
6     return 0;
7  }
```

Les lignes 2 et 3 sont des commentaires. Ils sont encadré par les symboles </*> et <*/>.
Pour faire des commentaires sur une seule ligne, on peut également utiliser <//> en début de ligne.
Les commentaires ne sont pas interpréter dans la compilation et ne modifie en rien le comportement

du programme. En revanche, ils sont très utiles aux développeurs car ils permettent d'indiquer comment un bout de code fonctionne

A la ligne 1, on retrouve une directive qui indique au compilateur qu'on aura besoin du code contenu dans stdio.h pour que la compilation soit un succès.

A la ligne 4, on retrouve une fonction main. Toutes les fonctions en C se déclare de la manière suivante:

typeRetour nomFonction (typeArgument nomArgument) {}

- Le typeRetour correspond à ce que l'on attend en sortie (est-ce un entier ? Une chaine de caractère?).
- Le nom de la fonction c'est comment celle-ci est identifiée.
- Entre les parenthèses, on indique les arguments que l'on va utilisé dans la fonction en précisant son type, puis son nom
- On indique entre accolades le contenu de la fonction

La fonction main est un peu particulière puisqu'il s'agit d'une fonction essentielle dans un programme C : c'est elle qui va être exécutée en premier et qui appellera les autres fonctions. **TOUS PROGRAMMES C DOIT AVOIR UNE FONCTION MAIN DANS L'UN DES FICHIERS**.

A la ligne 7, le return permet de renvoyer un résultat mais également de clore la fonction. Dans le cas de la fonction main, on renvoie 0 pour indiquer au système que tout s'est déroulé correctement. C'est pour ça que le type de retour défini dans la déclaration est int.

Enfin, a la ligne 5, on retrouve une instruction simple qui permet d'afficher dans la console. Toutes instructions en c se termine par <;>

Chapitre 2: Les types de données

2.1) Les types

Tout contenu sur un système informatique est stocké sous forme de bits et localisé grâce à des adresses. Cependant connaître l'adresse du contenu que l'on veut consulter ne suffit pas, il faut également connaître son type. Par exemple, le caractère 'A' et l'entier 65 sont stocké de la même manière, seul le type permet de les différencier

2.2) Les types de bases

2.2.1) Les entiers

Noté int, et dans l'espace mémoire qui est alloué au stockage d'un int, le premier bits sers à noté le signe. La taille de l'espace mémoire peut varier à l'aide des variations suivantes :

- short int → plus petit
- long int → plus long que int
- long long int → encore plus long !

Pour connaître la taille en mémoire, il faut regarder le fichier limits.h ou utiliser la fonction sizeof

On peut utiliser la variation unsigned pour n'avoir qu'un nombre en valeur absolue.

2.2.2) Les types flottants

Permet de représenter les réels. Les types float sont

- float
- double
- long double

Les tailles de stockage sont trouvable dans le fichier float.h

2.2.3) Le type caractère

Les caractères sont stockés comme des entiers mais sont interprétés comme des caractères grâce à la table ASCII

La déclaration se fait avec le mot clef char

2.2.4) Affectation et conversion

Toute variables utilisées doit être préalablement déclaré. Pour déclarer une variable on procède comme suit :

```
int var1= 0, var2 = 3;
```

```

```

Les variables var1 et var2 sont de type int et prennent pour valeur 0 et 3 ;

Une fois qu'elle est déclarée, qu'elle est une valeur ou non, on peut lui affecter une autre valeur :

```
var1 = 2*var2+7;
```

La variable var1 prend alors comme valeur 2x3+7 soit 13

Pour convertir une valeur d'un certains type vers un autre, on utilise un cast.

```
int n;
float x = 3.5;
n = (int)x;
```

Dans ce cas, x prend pour valeur la valeur entière de x, c'est à dire 3.

2.3) Les constantes

Les constantes sont des valeurs que l'on va définir en début de programmes et qui ne bougerons pas quelque soit le traitement. En C, il existe deux manière de déclarer une variable :

A) Avec la directive #define

```
#define PI 3.14
```

On constate que la constante est déclarée en majuscule et qu'il n'y a pas de <;> en fin de ligne. De plus, le type de la constante est déduit de la valeur indiquée.

B) Avec le mot clef const

```
const float PI = 3.14;
```

Comme il s'agit d'une instruction et non d'une directive, on termine cette fois ci la ligne avec un <;>. De plus, on doit typé la constante pour que cela fonctionne

La différence entre les deux sera la visibilité de celles-ci. Nous verrons ça dans un chapitre ulterieur.

2.4) Définir ses propres types

Il est possible de créer ses propres type et les renommer à l'aide du mot clef typedef

```
typedef int Entier;
entier j = 0;
```