MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET

DE L'ALPHABÉTISATION

UNIVERSITÉ GAMAL ABDEL NASSER DE CONAKRY

INSTITUT POLYTECHNIQUE

DÉPARTEMENT GENIE INFORMATIQUE

Mise à niveau par séances de Télé-travail

Programme: Licence 1

Cours: Programmation en

Table de matières

Première partie

Deuxième partie

Objectfis

- L’objectif de ce cours est de **fournir une connaissance complète du langage** C.
- Les étudiants seront capables de développer des logiques qui les aideront à créer des programmes et des applications en C.
- De plus, en apprenant les méthodes d’analyse et de constructions de programmes de base, ils pourront facilement passer à n'importe quel autre langage à l'avenir, tels que le c++, le java, python,…

Compétences à aquérir

A l'issu de cette formation vous aurez les compétences necessaires pour:

- Analyser un programme algorithmique
- Comprendre les syntaxes du langage C
- Saisir les différentes structures de contrôle à fin d'automatiser des tâches spécifiques
- Maîtriser la notion de fichiers d'entêtes et sources pour être à la hauteur de définir une architecture concrète d'un mini-projet ou projet réalisable avec le language C.
- Aquérir les notions fondamentales à fin d'aborder efficacement les structures de données en exploitant différents algorithmes sur les tableaux et chaînes de caractères.

Introduction au C

- ✓ C est probablement le langage de programmation le plus connu. Il est utilisé comme langage de référence pour les cours d'informatique partout dans le monde, et c'est probablement le langage que les gens apprennent le plus à l'école avec Python et Java.
- ✓ C n'est pas seulement ce que les étudiants utilisent pour apprendre la programmation. Ce n'est pas un langage académique. Et je dirais que ce n'est pas le langage le plus simple, car le C est un langage de programmation plutôt bas niveau.

- ✓ Aujourd'hui, C est largement utilisé dans les appareils embarqués et il alimente la plupart des serveurs Internet, qui sont construits avec Linux. Le noyau Linux est construit à l'aide de C, ce qui signifie également que C alimente le cœur de tous les appareils Android. On peut dire que le code C exécute une bonne partie du monde entier

- ✓ C est un langage de programmation compilé, comme Go, Java, Swift ou Rust. D'autres langages de programmation populaires comme Python, Ruby ou JavaScript sont interprétés. La différence est constante : un langage compilé génère un fichier binaire qui peut être directement exécuté et distribué.

Introduction au C

Prémier programme en C

```c
#include <stdio.h>

int main(void) {
    printf("Hello, World!");
}
```

Décrivons le code source du programme : nous importons d'abord la stdiobibliothèque stdio.h (signifie bibliothèque d'entrée-sortie standard).
Cette bibliothèque nous donne accès aux fonctions d'entrée/sortie.
C est un très petit langage à la base, et tout ce qui ne fait pas partie du noyau est fourni par les bibliothèques. Certaines de ces bibliothèques sont construites par des programmeurs normaux et mises à la disposition d'autres utilisateurs. Certaines autres bibliothèques sont intégrées au compilateur. Comme stdioet d'autres.
stdioest la bibliothèque qui fournit la printf()fonction.
Cette fonction est enveloppée dans une main()fonction. La main()fonction est le point d'entrée de tout programme C.

Introduction au C

Compiler un programme C

- Pour compiler un programme C, à l'aide d'un IDE(Integrated Development, Environnement), il faut cliquer simplement sur le bouton « build » et pour démarrer, cliquer sur le bouton « run »

- Nous pouvons également utiliser un terminal windows ou linux pour compiler: en utilisant la commande gcc.

```
gcc hello.c -o hello
```

Varibales et types

C est un langage typé statiquement.
Cela signifie que toute variable a un type associé, et ce type est connu au moment de la compilation.
C'est très différent de la façon dont vous travaillez avec des variables en Python, JavaScript, PHP et d'autres langages interprétés.
Lorsque vous créez une variable en C, vous devez spécifier le type de la variable à la déclaration.
Dans cet exemple nous initialisons une variable **age** avec le type **int**:

```
int age = 37;
```

Vous pouvez également initialiser une variable lors de la déclaration, en spécifiant la valeur initiale :

```
int age = 37;
```

Varibales et types

Une fois que vous déclarez une variable, vous pouvez alors l'utiliser dans votre code de programme. Vous pouvez modifier sa valeur à tout moment, à l'aide de l'opérateur **=** par exemple, comme dans age = 100; (à condition que la nouvelle valeur soit du même type).
Danc ce cas:

```
#include <stdio.h>

int main(void) {
    int age = 0;
    age = 37.2;
    printf("%u", age);
}
```

le compilateur déclenchera un avertissement au moment de la compilation et convertira le nombre décimal en une valeur entière.

Varibales et types

Les types de données intégrés C sont:
int, char, short, long, float, double, long double

Nombres entiers
C nous fournit les types suivants pour définir des valeurs entières :**int, char, short, long**

Entiers non signés
Pour tous les types de données ci-dessus, nous pouvons préfixer **unsigned** pour commencer la plage à 0, au lieu d'un nombre négatif. Cela pourrait avoir un sens dans de nombreux cas.

- ✓ **unsigned char** ira de 0 à au moins 255
- ✓ **unsigned int** ira de 0 à au moins 65,535
- ✓ **unsigned short** ira de 0à au moins 65,535
- ✓ **unsigned long** ira de 0 à au moins 4,294,967,295

Varibales et types

Nombres à virgule flottante

Les types à virgule flottante peuvent représenter un ensemble de valeurs beaucoup plus grand que les entiers, et peuvent également représenter des fractions, ce que les entiers ne peuvent pas faire.
Ce sont: **float, double, long double**

La configuration minimale requise pour toute implémentation C est que le type float peut représenter une plage comprise entre 10^-37 et 10^+37, et est généralement implémentée en utilisant 32 bits. Double peut représenter un plus grand ensemble de nombres. long double peut contenir encore plus de numéros.

Sur votre ordinateur spécifique, comment pouvez-vous déterminer la taille spécifique des types ? Vous pouvez écrire un programme pour faire cela :

```
#include <stdio.h>

int main(void) {
  printf("char size: %lu bytes\n", sizeof(char));
  printf("int size: %lu bytes\n", sizeof(int));
  printf("short size: %lu bytes\n", sizeof(short));
  printf("long size: %lu bytes\n", sizeof(long));
  printf("float size: %lu bytes\n", sizeof(float));
  printf("double size: %lu bytes\n",
    sizeof(double));
  printf("long double size: %lu bytes\n",
    sizeof(long double));
}
```

Constantes

Une constante est déclarée de la même manière que les variables, sauf qu'elle est précédée du mot-clé **const** et que vous devez toujours spécifier une valeur.
Comme ça: **const int taille = 80;**
Par convention, le nom des variables constantes sont généralement en majuscule.
Exemple: const TAILLE = 80;

Une autre façon de définir des constantes consiste à utiliser cette syntaxe :
#define TAILLE 80;

Dans ce cas, vous n'avez pas besoin d'ajouter de type, et vous n'avez pas non plus besoin du signe égal, et vous omettez le point-virgule à la fin.
Le compilateur C déduira le type à partir de la valeur spécifiée, au moment de la compilation.

Opérateurs

C nous offre une grande variété d'opérateurs que nous pouvons utiliser pour opérer sur les données.
En particulier, on peut identifier différents groupes d'opérateurs :

- ✓ opérateurs arithmétiques
- ✓ opérateurs de comparaison
- ✓ Opérateurs logiques
- ✓ opérateurs d'affectation composés
- ✓ opérateurs au niveau du bit
- ✓ opérateurs de pointeur
- ✓ opérateurs de structure
- ✓ opérateurs divers

Dans cette section, Nous allons tous les détailler, en utilisant 2 variables imaginaires a et b comme exemples. Je garde les opérateurs au niveau du bit, les opérateurs de structure et les opérateurs de pointeur hors de cette liste, pour simplifier les choses

Opérateurs

Opérateurs arithmétiques

Dans ce groupe, nous allons séparer les opérateurs binaires et les opérateurs unaires.

✓ Les opérateurs binaires fonctionnent avec deux opérandes :

OPÉRATEUR	NOM	EXEMPLE
`=`	Mission	`a = b`
`+`	Une addition	`a + b`
`-`	Soustraction	`a - b`
`*`	Multiplication	`a * b`
`/`	Division	`a / b`
`%`	Modulo	`a % b`

Opérateurs

Opérateurs arithmétiques

✓ Les opérateurs unaires ne prennent qu'un seul opérande:

OPÉRATEUR	NOM	EXEMPLE
`+`	Unaire plus	`+a`
`-`	Moins unaire	`-a`
`++`	Incrément	`a++` ou alors `++a`
`--`	Décrémenter	`a--` ou alors `--a`

La différence entre a++ et ++a est que a++ incrémente la avariable a après l'avoir utilisée. ++a incrémente la variable a avant de l'utiliser.
Par example:

```
int a = 2;
int b;
b = a++ /* b is 2, a is 3 */
b = ++a /* b is 4, a is 4 */
```

Opérateurs

Opérateurs de comparaison

✓ Les opérateurs unaires ne prennent qu'un seul opérande:

OPÉRATEUR	NOM	EXEMPLE
`==`	Opérateur égal	`a == b`
`!=`	Opérateur non égal	`a != b`
`>`	Plus grand que	`a > b`
`<`	Moins que	`a < b`
`>=`	Supérieur ou égal à	`a >= b`
`<=`	Inférieur ou égal à	`a <= b`

Opérateurs logiques

✓ !NON (exemple : !a)
✓ &&ET (exemple : a && b)
✓ ||OU (exemple : a || b)

Opérateurs

Opérateurs d'affectation composés

Ces opérateurs sont utiles pour effectuer une affectation et en même temps effectuer une opération arithmétique :

OPÉRATEUR	NOM	EXEMPLE
`+=`	Affectation supplémentaire	`a += b`
`-=`	Affectation de soustraction	`a -= b`
`*=`	Affectation de multiplication	`a *= b`
`/=`	Affectation de division	`a /= b`
`%=`	Affectation modulo	`a %= b`

Opérateurs

L'opérateur ternaire
L'opérateur ternaire est le seul opérateur en C qui fonctionne avec 3 opérandes, et c'est un moyen court d'exprimer des conditions.
Voici à quoi ça ressemble:

```
<condition> ? <expression> : <expression>
```

Exemple:

```
a ? b : c
```

Si **a** est évalué à true, alors l' instruction **b** est exécutée, sinon l'insruction **c** est exécutée.
L'opérateur ternaire est fonctionnellement identique à un conditionnel **if/else**, sauf qu'il est plus court à exprimer et qu'il peut être intégré dans une expression.

Conditionnels

Tout langage de programmation offre aux programmeurs la possibilité d'effectuer des choix.
Nous voulons faire X dans certains cas, et Y dans d'autres cas.
Nous voulons vérifier les données et faire des choix en fonction de l'état de ces données.
C nous propose 2 façons de le faire.

Le premier est l' **if** avec son assistant **else**, et le second est le **switch**.

if

Dans une instruction if, vous pouvez vérifier qu'une condition est vraie, puis exécuter le bloc fou
entre accolades :

```
int a = 1;

if (a == 1) {
  /* do something */
}
```

Vous pouvez ajouter un bloc **else** pour exécuter un bloc différent si la condition d'origine
s'avère être fausse :

```
int a = 1;

if (a == 2) {
  /* do something */
} else {
  /* do something else */
}
```

Conditionnels

Vous pouvez avoir plusieurs blocs **else** en empilant plusieurs instruction **if**:

```
int a = 1;

if (a == 2) {
  /* do something */
} else if (a == 1) {
  /* do something else */
} else {
  /* do something else again */
}
```