TP 1 : Introduction à C

12 septembre 2023

1 Environnement de Travail

Pour programmer en C, il est nécessaire d'avoir un espace pour :

— écrire le programme
— le compiler
— l'exécuter et le debugger

Un logiciel ayant des composantes permettant d'effectuer toutes ou une partie de ces tâches s'appelle un **IDE (Integrated Development Environment)**. Le programme permettant d'effectuer la compilation s'appelle un **compilateur**. Des compilateurs sont généralement inclus dans le téléchargement des IDE.

Il existe également des IDE en ligne permettant d'exécuter des programme dans un terminal en ligne, sans avoir à télécharger de logiciels pour commencer à coder. Dans ce TP, on utilisera OnlineGDB :

https ://www.onlinegdb.com/online_c_compiler

Les éditeurs de texte des IDE ont notamment comme intérêt de mettre en forme le texte selon la syntaxe, en modifiant par exemple la couleur d'un mot du programme selon ce à quoi il correspond (type, variable, fonction...). Cela permet notamment de facilement repérer certaines erreurs avant compilation, lorsque la mise en forme observée ne correspond pas à celle attendue.

2 Caractéristiques de C

2.1 Langage Impératif

Le langage C est un langage impératif. Cela signifie que le contenu du programme consiste en une suite d'instructions à exécuter pour la machine ; les opérations correspondent à des modifications de l'état du programme.

L'une des caractéristiques des langages impératifs est l'utilisation de **variables**, que l'on peut définir et auxquelles on peut affecter une valeur (assignation). Une variable est caractérisée par :

— son **identifiant** (nom donné dans le programme)

— son **type**

— sa **valeur** dans l'ensemble des valeurs possibles pour son type, une fois qu'on lui en a affectée une.

L'affectation d'une valeur (12, par exemple) à une variable `v` donnée se fait simplement via la ligne de commande :

```
v = 12;
```

Les variables représentent des espaces en mémoire : lorsque l'on affecte une valeur à une variable, on stocke cette valeur dans un espace mémoire, et faire appel à cette variable permet d'utiliser sa valeur et d'effectuer des opérations dessus.

2.2 Langage Typé

ATTENTION : le C est un langage typé. **Il est nécessaire de déclarer la variable avec son type pour pouvoir l'utiliser**, sans quoi une erreur sera détecter. Cela diffère en particulier d'un langage comme Python, où la déclaration d'une variable et de son type se fait implicitement au moment où on lui affecte une valeur. En Python, par exemple, il n'est pas incorrect d'avoir une seule ligne `v = 12` pour affecter la valeur 12 à la variable `v`.

En C, en revanche, écrire la seule ligne `v = 12;` à l'intérieur d'un programme pour donner à `v` la valeur 12 est insuffisant, et renverra une erreur. Il est nécessaire de déclarer la variable et son type (ici `int`, pour Entier) avant de lui affecter une valeur :

```
int v;
v = 12;
```

Remarque. La déclaration de variable ne doit pas obligatoirement être suivie d'une affectation. Par ailleurs, il est possible d'affecter des nouvelles valeurs à une variable au cours de l'exécution du programme (et notamment affecter une nouvelle valeur en fonction de sa valeur précédente).

Remarque. La syntaxe du C permet également de déclarer une variable et de lui affecter une valeur simultanément. La ligne de commande ci-dessous équivaut aux deux lignes écrites précédemment :

```
int v=12;
```

En C, les types élémentaires sont :

— Les **entiers** (`int`)

— Les nombres à virgules flottante, où **flottants** (`float`)

— Les **caractères** (`char`), notés entre guillemets simples (`'a'`)

— Les **booléens** (`bool`), qui peuvent valoir `true` ou `false`

On peut également utiliser des opérations de base sur des variables en fonction de leur type : opérations mathématiques élémentaires sur des entiers et des flottants ; opérateurs logiques (négation, conjonction, disjonction) sur les booléens ; opérateurs de comparaison (égalité, différence, inégalités strictes et larges) qui prennent en entrée deux variables numériques (entiers ou flottants) et renvoient un booléen...

2.3 Langage Compilé

Le langage C est un langage **compilé** : le programme en C est stocké dans un fichier .c, puis il est traduit en langage machine pour créer un fichier exécutable. La compilation d'un programme se fait à l'aide d'un compilateur ; elle peut se faire au sein d'un IDE, ou en ligne de commande dans un terminal. Le compilateur le plus connu pour C est GCC.

3 Premiers Pas, Chaînes de Caractères

Exercice 1 :

1. Ouvrez OnlineGDB à cette adresse : `https://www.onlinegdb.com/online_c_compiler`
2. Appuyez sur "Run" pour lancer la compilation et l'exécution du programme.
3. Décrivez le résultat.

Le programme compilé et exécuté par défaut sur le site est un programme Hello World : il a pour seul but d'afficher la chaîne de caractères "Hello World", servant dans l'apprentissage à montrer un exemple simple de bonne exécution d'un programme. On peut déjà observer certaines composantes fondamentales d'un programme élémentaire en C :

— La première instruction exécutée (`#include <stdio.h>` à la ligne 9) correspond au **chargement d'une librairie** (ou bibliothèque, *library* en anglais). De même que sous Python, il est possible de faire appel à des librairies qui contiennent un certain nombre de fonctions utiles. En particulier, en C, la plupart des outils élémentaires nécessitent de charger des librairies : la fonction `printf` se trouve ici dans l'en-tête (header) `stdio.h` de la bibliothèque standard du C.

— Dans un fichier .c, on peut écrire plusieurs fonctions, mais le coeur du fichier est la **fonction** `main` : c'est la fonction qui sera lancée au moment de l'exécution (les autres fonctions définies pouvant être appelées à l'intérieur de cette fonction `main`).

— Les fonctions ont des types pour les paramètres d'entrée et pour la sortie. Dans le programme, la fonction main à un **type** `int`. On voit par ailleurs qu'elle **renvoie** (`return`) la valeur 0 : cela sert à signaler la bonne exécution du code de la fonction main. Cette instruction est rajoutée implicitement quand elle n'est pas écrite dans le programme.

— Enfin, on observe que du texte écrit en anglais est présent sur les premières lignes, sans qu'il semble avoir été pris en compte lors de la compilation. Il s'agit de **commentaires** : cela permet notamment de décrire explicitement en langage humain ce qui est effectué dans certaines portions de programme. **Il est très important de commenter son code** pour le rendre compréhensible à la lecture par autrui (ou à la relecture pour soi...), surtout lorsque l'on commence à concevoir des programmes plus sophistiqués.

Vous pouvez enregistrer le code dans votre dossier personnel sur l'ordinateur (bouton bleu clair avec une flèche vers le bas), dans un fichier .c (`hello_world.c` par exemple), ce qui vous permettra de le compiler et l'exécuter localement si vous avez un IDE sur votre ordinateur.

4 Nombres et Affichage, Programmes Interactifs

On a vu que l'on disposait de deux types élémentaires pour représenter des nombres : `int` pour les entiers et `float` pour les flottants (qui servent d'approximation des réels). On peut utiliser les opérations de base, addition et soustraction par exemple, entre deux entiers ou entre deux flottants.

Exercice 2 :

Ecrire un programme qui dans `main` affecte à un entier `x` la valeur 5, à un entier `y` la valeur 10 et à un entier `z` la valeur `x+y`.

On a vu que notre fonction `main` renvoie 0 ; on peut avoir envie d'afficher la valeur de `z` pour vérifier que le calcul s'est bien effectué.

La syntaxe pour afficher la valeur d'une variable dans une chaîne de caractères en C est la suivante : dans le `printf`, à la position où l'on souhaite afficher la valeur de la variable dans une chaîne de caractère, on écrit `%d` (pour un entier) ou `%f` (pour un flottant) ; puis à la suite de la chaîne de caractère, toujours à l'intérieur du printf, on écrit la liste des variables que l'on souhaite afficher, séparées par des virgules.

Exercice 3 :

1. Dans le programme précédent, rajouter à l'intérieur de `main` la ligne suivante :
   `printf("%d + %d = %d",x,y,z);`
2. Décrire le résultat affiché.

On peut également faire en sorte que l'utilisateur rentre des valeurs dans le terminal pendant l'exécution d'un programme, pour que ces valeurs soient affectées à des variables. La syntaxe (que l'on détaillera plus tard) pour demander à un utilisateur de rentrer une valeur pour un entier `x` est la suivante :

`scanf("%d",&x)`

Exercice 4 :

1. Modifier le programme précédent pour demander à l'utilisateur de rentrer la valeur de `x` (précédé du texte "Valeur de x : ") et la valeur de `y` (précédé du texte "Valeur de y : "). ATTENTION, il faut déclarer les types de x et y avant de demander leur valeur.
2. Tester le programme sur les valeurs 213 et 357 pour x et y.
3. Enregistrer le programme dans votre dossier personnel dans un fichier `somme.c`.

5 Booléens

On rappelle que les variables booléennes, dans les langages de programmation, sont des variables dont les valeurs possible sont Vrai et Faux. En C, les booléens sont représentés par les nombres 1 (pour Vrai) et 0 (pour Faux).

Exercice 5 :

1. Exécuter un programme avec l'instruction `printf("%d",2==2);` dans la fonction `main`. Décrire le résultat.

2. Même question avec l'instruction `printf("%d",3==2);` à la place.

Dans le cas ci-dessus, l'égalité entre deux entiers est une opérations booléenne, qui prend deux entiers et renvoie 1 si l'égalité est vérifiée et 0 sinon. Les opérateurs logiques (disjonction `||`, conjonction `&&` et négation `!`) permettent de faire du **calcul booléen** :

Exercice 6 :

1. Exécuter un programme avec l'instruction `printf("%d",!(2==2));` dans la fonction `main`. Décrire le résultat.
2. Même question avec l'instruction `printf("%d",2==2 && 3==2);` à la place.
3. Même question avec l'instruction `printf("%d",2==2 || 3==2);` à la place.

Il n'existe pas de type booléen à proprement parler en C, avec des variables pouvant valoir Vrai ou Faux (comme en Python où l'on a True et False). On peut cependant définir ce type et ces variables (de telle sorte que true vale 1 et false vale à) pour les manipuler.

Exercice 7 :

1. Exécuter un programme avec l'instruction `printf("%d",true);` dans la fonction `main`. Décrire le résultat.
2. Même question en rajoutant `#include <stdbool.h>` en en-tête dans le programme.

6 Conditions

En C, il est possible d'utiliser des **instructions conditionnelles** qui permettent d'enrichir les possibilités de structures du programme. Des instructions sont exécutées seulement si une condition est vérifiée ; cette condition est exprimée comme un booléen. On peut par exemple souvent être amené à utilisé les différents opérateurs de comparaison : `==`, `!=`, `>`, `<`, `>=` ou `<=`.

La syntaxe d'une condition est la suivante :

```
if (condition) {
[instructions]
}
```

On peut également utiliser `else` (bien que ce ne soit pas obligatoire) pour exécuter des instructions si la condition n'a pas été vérifiée :

```
if (condition) {
[instructions]
} else {
[autres instructions]
}
```

Si l'on souhaite lister plusieurs alternatives jusqu'à ce qu'une condition soit vérifiée (ou non), on peut utiliser `else if` :

```
if (condition_1) {
[instructions]
} else if (condition_2) {
```

```
[autres instructions]
} else {
[autres instructions]
}
```

On peut rajouter autant de `else if` que l'on souhaite.

Exercice 8 :

Ecrire un programme qui demande à l'utilisateur une valeur d'un entier x (précédé du texte "valeur de x : ") et qui affiche "x est positif", "x est nul" ou "x est négatif" en fonction de la valeur de x.

7 Exercices

Exercice 9 :

On rappelle que pour un entier n donné, son successeur dans la suite de Syracuse vaut $3n+1$ si n est impair et $n/2$ si n est pair. On rappelle également qu'on calcule le reste dans la division euclidienne de a par b avec $a\%b$.

Ecrire un programme qui demande un entier n et affiche son successeur dans la suite de Syracuse.

Exercice 10 :

Ecrire un programme qui demande trois réels a, b et c (avec a non-nul) et qui affiche le nombre de racine de la fonction polynomiale définie par $f(x) = ax^2 + bx + c$.

Exercice 11 :

Ecrire un programme qui demande trois longueurs positives de côtés l_1, l_2 et l_3 (où aucune n'est supérieure à la somme des deux autres), et indique si le triangle dont les longueurs des côtés valent l_1, l_2 et l_3 est un triangle rectangle.

Exercice 12 :

Ecrire un programme qui demande trois longueurs positives de côtés l_1, l_2 et l_3 (où aucune n'est supérieure à la somme des deux autres), et indique si le triangle dont les longueurs des côtés valent l_1, l_2 et l_3 est un triangle rectangle, isocèle, isocèle rectangle ou équilatéral (ou aucun des 4).

Exercice 13 :

Ecrire un programme qui demande les coordonnées a_1, a_2, b_1, b_2, c_1, c_2 de trois points A, B et C dans le planet indique si le triangle ABC est un triangle rectangle.