Chapitre 2

Introduction au langage C - types et variables

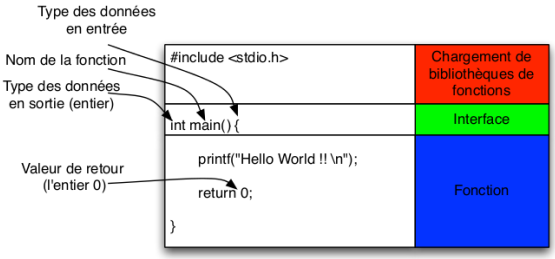
1 Le Langage C

Le langage C est un langage de bas niveau dans le sens où il permet l'accès à des données que manipulent les ordinateurs (bits, octets, adresses) et qui ne sont pas souvent disponibles à partir de langages évolués tels que Fortran, Pascal ou ADA.

Le langage C a été conçu pour l'écriture de systèmes d'exploitation (plus de 90% du noyau du système UNIX est écrit en langage C).

1.1 Structure d'un programme en langage C

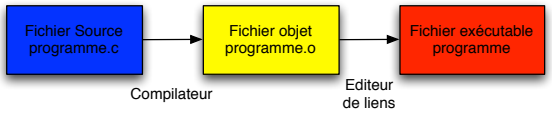
Les programmes écrits en langage C sont sauvegardés avec l'extension ".c", exemple : "`tp1.c`". Le fichier `tp1.c` peut être créé à l'aide de n'importe quel éditeur de texte, certains éditeurs sont néanmoins plus appropriés pour l'écriture de programmes en langage (`vi`, `xemacs`, `gedit`, `kate`, `nano`, etc.). Ci-dessous la structure générale d'un programme en langage C :



1.2 Chaine de production et compilation

Comme nous l'avons vu en cours, afin que la machine puisse exécuter des instructions, il faut que ces dernières soient traduites en **langage machine**. Une fois notre programme (instructions) écrit en langage C, il faut donc un intermédiaire pour traduire ce dernier en langage machine. C'est le rôle du **compilateur** qui à partir du fichier ".c" crée un fichier binaire dit **exécutable** qui servira à lancer l'**exécution** (par la machine) des instructions programmées.

Le schéma ci-dessous résume la chaine de production (simplifiée !!) du fichier exécutable :



Nous utiliserons ici le compilateur `gcc` de la façon suivante :

gcc –o nom_executable nom_fichier_source.c

Ainsi, la commande suivante :

```
gcc -o tp1 tp1.c
```

Elle permet de générer un exécutable à partir du fichier `tp1.c`. L'option *-o* de la commande `gcc` permet de spécifier le nom du fichier exécutable généré, ici `tp1` **Attention : ne pas mettre d'extension**.

2 Types de base et variables en langage C

Les types de base sont ceux prédéfinis du compilateur. Ils sont au nombre de six :

1. ***void*** c'est le type vide. Il a été introduit par la norme ANSI. Il est surtout utilisé pour préciser les fonctions sans argument ou sans retour (à voir en TP à partir de la séance 6).
2. ***int*** c'est le type entier. Ce type se décline avec des qualificatifs pour préciser sa taille (*long* ou *short*), et le fait qu'il soit uniquement positif (*unsigned*) ou positif et négatif (*signed*). Le qualificatif *signed* est appliqué par défaut, ainsi il n'y a pas de différence entre une variable de type *int* et une variable de type *signed int*.
3. ***char*** ce type est le support du code ASCII qui permet la représentation des caractères. Il peut aussi représenter un entier sur huit bits. Le *char* peut être déclaré en signé (*char*) ou alors en non signé (*unsigned char*)
4. ***float*** ce type sert pour la représentation des réels (avec parties décimales).
5. ***double*** idem que le *float* mais avec un précision plus importante.
6. *long double* idem que *double* mais avec une précision encore plus grande.

3 Définition de la fonction `printf`

Avant de commencer à faire vos premiers programmes en C, nous allons présenter la fonction `printf` qui vous sera très utile pour l'affichage des messages et des valeurs des variables. Cette fonction permet d'écrire des données dites formatées dans la sortie standard (écran). Voici deux exemples d'utilisation de `printf` :

1. `printf("Hello World!!");`
2. `printf("%d kilogramme équivaut à %d grammes", 1, 1000);`

L'argument de la fonction `printf` dit **format** est une chaîne de caractères qui détermine ce qui sera affiché par `printf` et sous quelle forme. Dans l'exemple 1 c'est `"Hello World!!"` et dans l'exemple 2 c'est `"%d kilogramme équivaut à %d grammes"`. Cette chaîne est composée de texte "normal" et de séquences de contrôle permettant d'inclure des variables dans la sortie. Ainsi, dans l'exemple 2, lors de l'affichage le premier `%d` sera remplacé par la valeur `1` et le second `%d` par la valeur `1000`. Il en résulte l'affichage suivant : `1 kilogramme équivaut à 1000 grammes`.

Les séquences de contrôle commencent par le caractère « `%` » suivi d'un caractère parmi :

- *d* ou *i* pour afficher un entier signé au format décimal (*int*) ;
- *u* pour un entier non signé au format décimal ;
- *o* pour afficher un entier au format octal ;
- *x* ou *X* pour afficher un entier au format hexadécimal (avec les lettres `"abcdef"` pour le format *x* et `"ABCDEF"` avec le format *X*) ;
- *f* pour afficher un réel (*float* et *double*) ;
- *c* pour afficher en tant que caractère ;
- *s* pour afficher une chaine de caractère ;

Il existe d'autres séquences de contrôle non abordées dans ce cours.

4 Définition de la fonction `scanf`

Cette fonction permet de lire des données dites formatées à partir de l'entrée standard (clavier).
L'interaction en mode console avec l'utilisateur devient rare : nous nous limiterons à la saisie d'une valeur numérique [1].

1. La fonction scanf permet également de réaliser la saisie de caractères ou de chaînes de caractères, mais autorise aussi la saisies de valeurs multiples (usage déconseillé dans le cadre d'un module d'introduction).

Important `scanf` utilise les mêmes formats que printf mais on fait précéder le nom de la variable du caractère esperluette ("&").

Syntaxe (pour une variable *i* de type *int*) :

```
scanf ("%d", &i);
```

N.B. : seul le format est précisé et passé en paramètre : il ne faut bien sûr n'ajouter aucun message ni autres caractères (même des espaces) sous risque d'un comportement imprévisible...

5 Écrire un programme en langage C

5.1 Syntaxe élémentaire et règles d'écriture

Voici quelques règles très basiques sur la syntaxe du langage C (le reste suivra au fur et à mesure de votre avancement) :

- Chaques instruction se termine toujours par un ";"
- les instructions sont toujours comprises dans des blocs délimités par une ouverture d'accolade "{" et une fermeture d'accolade "}"
- certains mots sont reservés (voir ci-dessous) et ne doivent pas être utilisés comme noms de variables ou de fonctions

Il est important de respecter la syntaxe du langage mais il est aussi important de respecter des règles d'écriture qui ne seront pas vérifiées par le compilateur mais qui permettent une meilleure lisibilité de votre programme, ceci facilite notamment les phases de debugage de votre code C!!! :

- Pensez à bien indenter votre programme
- ne donnez pas de nom de variables ou de fonctions qui commencent par un chiffre
- ne pas insérer d'espace dans les noms des variables et fonctions (l'espace sera interprété par le compilateur)
- donnez toujours des noms explicites à vos variables et fonctions
- dans le cas d'un nom complexe, si la variable indique par exemple votre note en informatique utilisez plutôt : `note_informatique` ou `noteInformatique`, ne pas utiliser le tiret ("-") qui correspond à l'opération soustraction (et ne surtout pas mettre d'espace!!)
- aérez votre code, laissez des espaces entre, par exemple, la partie déclaration des variables, instructions, affichages etc.

Mots réservés

Ce sont les mots prédéfinis du langage C. Ils ne peuvent pas être réutilisés pour des identifiants. Ils sont relatifs aux différents concepts du langage :

- type des données : `char const double float int long short signed unsigned void volatile`
- instructions de boucle : `do for while`
- sélections : `case default else if switch`
- ruptures de séquence : `break continue goto return`
- classes d'allocation (à voir en S2 et plus!!) : `auto extern register static`
- constructeurs (à voir en S2 et plus!!) : `enum struct typedef union`
- divers : `asm entry fortran sizeof`

5.2 Premier programme en langage C

Ouvrez un terminal et tapez-y la commande suivante :

```
gedit HelloWorld.c &
```

Cette commande permet de lancer la création du fichier `HelloWorld.c` à l'aide de l'éditeur de texte `gedit`. Le caractère `&` permet de lancer cette commande en tâche de fond. Sans le `&`, vous seriez dans l'obligation de fermer votre éditeur si vous voulez de nouveau lancer une commande via le terminal.

Tapez **dans l'éditeur de texte**, le programme suivant :

```
#include <stdio.h>

int main () {
    printf("Hello World !!\n") ;
    return 0 ;
}
```

Une fois le fichier `HelloWorld.c` créé, sauvegardez-le (Enregistrer), puis, dans un terminal, lancez la compilation comme suit :

```
gcc -o HelloWorld HelloWorld.c
```

Le fichier exécutable `HelloWorld` est alors créé. pour l'exécuter, il suffit de taper toujours dans le terminal : `./HelloWorld`

Question que veut dire le "./" avant le nom du fichier exécutable ?).

5.3 Erreurs de compilation

Si la compilation échoue (vous avez fait une erreur de syntaxe ou oublié un point-virgule, par exemple), le compilateur vous signale l'erreur (et souvent le numéro de ligne). Par exemple :

```
HelloWorld.c: In function 'main':
HelloWorld.c:7: error: expected ';' before '}' token
```

Il ne vous reste alors qu'à re-éditer votre fichier `HelloWorld.c` en corrigeant l'erreur, **le sauvegarder** et recompiler.

Lorsque le compilateur vous signale beaucoup d'erreur, commencez à corriger la première. Cela suffit parfois pour éliminer les autres messages d'erreurs qui dépendaient de cette dernière.

Les erreurs de compilation courantes. Lorsq le compilateur vous indique :

- `error: expected ';' before...` : un point-virgule a été oublié.
- `error: expected declaration or statement at end of input` : une accolade a été oubliée.
- `error: 'nom_variable' undeclared (first use in this function)` : vous utilisez une variable qui n'a pas été déclarée préalablement (ou orthographiée différemment).
- `a.c:(.text+0x19): undefined reference to 'nom_fonction'`
  `collect2: error: ld returned 1 exit status` vous utilisez une fonction non connue par le compilateur (mal orthographiée ?)
- `fatal error: nom_fichier.h: Aucun fichier ou dossier de ce type` les fichiers d'en-têtes que vous incluez ont été mal orthographiés.

5.4 Deuxième programme en langage C

```
/* Exemple pour tester "scanf" */
#include <stdio.h>

int main () {
    int nb1 ;
    float nb2 ;

    printf("Saisissez une valeur entiere (positive ou  negative) pour nb1 : ") ;
    scanf("%d",&nb1) ;
    printf("Saisissez une valeur reelle pour nb2 : ") ;
    scanf("%f", &nb2) ;

    printf("nb1 vaut %d ; nb2 vaut %f\n", nb1, nb2) ;
    return 0 ;
}
```

6 Exercices

Question 2-1 Bonjour monde → exercice de cours

Suivez les consignes de la section 5.1 afin de :

1. créer votre programme `HelloWorld.c`
2. le compiler et obtenir l'exécutable `HelloWorld`
3. lancer l'exécution du fichier `HelloWorld`.

Une fois que le programme `HelloWorld.c` fonctionne, le modifier afin qu'il affiche également la ligne "Bonjour monde" après "Hello World".

Question 2-2 Demander un nombre → exercice de cours

1. Recopiez le programme donné en 4 dans un fichier appelé `saisie.c`
2. Compilez le programme et testez-le.
3. Faites en sorte que votre programme `saisie.c` affiche "Bonjour monde" après avoir affiché les valeurs saisies par l'utilisateur.

Question 2-3 La moyenne → exercice d'assimilation

Ecrivez un programme C qui calculer la moyenne de 2 notes (des entiers). Pour cela :

1. déclarez 2 entiers appelés *note1* et *note2*, et vous les initialiserez respectivement avec les valeurs 8 et 14.
2. déclarez une troisième variable entière appelée *moyenne* qui contiendra le résultat.
3. utilisez enfin une instruction `printf` pour afficher le résultat.
4. Modifiez votre programme afin de calculer la moyenne de 4 et 2. Testez.
5. Modifiez votre programme afin de calculer la moyenne de 13 et 17. Testez.

Il est intéressant dans un programme comme celui-ci de demander à l'utilisateur de saisir les valeurs plutot que ce soit au programmeur de les coder "en dur" dans le programme.

1. A l'aide de la fonction `scanf`, modifiez votre programme afin que les moyennes soient saisies par l'utilisateur **après** le lancement du programme.
2. Testez.

Vous testerez ce programme avec une valeur paire pour *note1* et impaire pour *note2*. Vous remarquerez que le résultat affiché n'est tout à fait juste.

1. Pourquoi ?
2. comment résoudre le problème ?
3. écrivez le programme pour tenir compte de votre solution :

Question 2-4 Equation → exercice d'entrainement

Résoudre l'équation de premier degré $ax + b = 0$. a et b seront initialisés avec les valeurs de votre choix.

Question 2-5 Calcul d'image → exercice d'entrainement

On veut calculer l'image de n'importe quel réel x par la fonction f définie par $f(x) = 3x + 4$. Affichez cette image en initialisant la variable x avec diverses valeurs réelles.

Question 2-6 Surface → exercice d'entrainement

Calculer la surface d'un rectangle en fonction de sa longueur et de sa largeur.

Question 2-7 Affichage de la date → exercice d'entrainement

Initialisez 3 variables *jour*, *mois*, *annee* de sorte à ce qu'elles correspondent au jour, mois et année d'aujourd'hui. Affichez ensuite ces données comme dans l'exemple qui suit :

```
Jour : 10
Mois : 11
Année : 2011
on est le 10/11/2011
```

Question 2-8 Permutation → exercice d'assimilation

Ecrivez un programme qui :

1. initialise deux variables a et b.
2. afficher les valeurs respectives des 2 variables.
3. permute les valeurs de a et b (c'est-à-dire que a prend la valeur de b et b prend celle de a).
4. affiche les valeurs respectives des 2 variables (pour vérifier qu'elles ont bien été permutées).

Question 2-9 Permutation → exercice d'entrainement

Donner le programme en C pour résoudre l' exercice suivant : Permuter les valeurs de 3 variables entières a, b, et c. C'est-à-dire : $a \to b, b \to c, c \to a$.

Question 2-10 Suite → exercice d'entrainement

Connaissant le premier terme a et la raison r d'une progression arithmétique de n termes, calculer la somme s des termes et le dernier terme l avec les formules suivantes :

$$l = a + (n-1)r$$
$$s = \frac{n(a + a(n-1)r)}{2}$$

7 Validation des compétences acquises à l'issue de cette séance

Je maitrise les compétences demandées à l'issue de cette séance si **je suis capable** de :

- ☐ expliquer brièvement le rôle d'un compilateur
- ☐ écrire un programme en C à l'aide d'un éditeur de texte et le compiler avec `gcc` en ligne de commande
- ☐ faire afficher à l'écran une chaine de caractère (= une petite phrase)
- ☐ déclarer une variable de type entier ou une variable de type décimal
- ☐ faire afficher à l'écran la valeur d'une ou plusieurs variables entière ou décimale au sein d'une chaine de caractère
- ☐ récupérer dans une variable un nombre entier saisi par l'utilisateur au clavier
- ☐ récupérer dans une variable un nombre décimal saisi par l'utilisateur au clavier
- ☐ permuter les valeurs de deux variables de même type