TD1: Introduction à la programmation en C

Le compilateur

Pour allons écrire un premier programme en `C`. Pour cela, nous devons utiliser un éditeur de texte pour écrire le texte du programme tout en respectant les règles de syntaxe du langage.

Le texte de notre programme sera valide syntaxiquement si un autre programme, appelé le 'compilateur', le reconnait et le transforme dans une série de formats intermédiaires de plus en plus proches du langage machine.

La commande pour lancer le compilateur C, sous Linux, est `cc`

TODO

- [ ] Ouvrez un Shell et lancez la commande suivante: `cc --help`
- [ ] Regardez attentivement les options suivantes: `-c` et `-o`
- [ ] Est-ce que `cc` produit nécessairement un programme exécutable ?

Un premier programme

```c
// section pour inclure les en-tete
#include <stdio.h>
#include <stdlib.h>

//section pour delarer les fonctions et variables globales
int main();

// section coder les fonctions
int main()
{
    printf("Bonjour!");
}
```

Voici un premier programme C minimal; Il est composé de 3 sections: - une section pour inclure les en-têtes de librairies; - une section de déclaration des fonctions et variables globales; - et une section qui donne le code des différentes fonctions (dans cet exemple une seule fonction est déclarée et implémentée)

La fonction déclarée dans ce programme, joue un rôle particulier: elle est reconnue, par convention, comme le point d'entrée du programme. Elle sera donc la première fonction exécutée lors du lancement du programme.

TODO

- [ ] Ouvrez votre éditeur de texte et recopiez le code source du programme
- [ ] Compilez votre programme avec la commande `cc` pour produire un fichier objet uniquement
- [ ] Compilez votre programme avec la commande `cc` pour produire un exécutable
- [ ] Lancez le programme produit par la commande `cc`
- [ ] Renommez la fonction `main` en `main1` et essayez de produire un exécutable. Que remarquez vous ?
- [ ] Le programme utilise la fonction `printf` où se trouve la déclaration de cette fonction ?

Les variables

```c
// section pour inclure les en-tete
#include <stdio.h>
#include <stdlib.h>
//section pour delarer les fonctions et variables globales
int main();
void addition(int i, int j);
// variable globale
int registre;
//section pour delarer les fonctions
// la fonction main
int main(){
    int a = 0;
    int b = 0;
    printf("donnez un entier ?\n");
    scanf("%d",&a);
    printf("donnez un entier ?\n");
    scanf("%d",&b);
    addition(a,b);
    printf("%d + %d = %d\n",a,b,registre);
}
// la fonction addition
void addition(int i, int j)
{
    registre = i + j;
}
```

Une variable en `C` est un emplacement en mémoire qui possède une adresse et qui va contenir une valeur.

Pour nous aider à programmer sans commettre trop d'erreurs, nous allons donner un type à nos variables pour avoir une indication sur la nature de la valeur

stockée.

Par exemple, dans le programme proposé, nous savons que `registre` est une variable qui va contenir un entier. Or un entier en `C` est codé sur 4 octets. Donc nous savons que l'espace mémoire occupé par cette variable est de exactement 4 octets.

Une variable `C` peut être déclarée à différents endroits. Voici les différentes situations que vous pouvez rencontrer:

- la variable est déclarée en dehors de toute fonction; elle est dans ce cas considérée comme une variable globale. Elle sera accessible depuis n'importe quelle fonction. Dans notre exemple, `registre` est une variable globale. Les fonctions `main` et `addition` peuvent accéder à cette variable en lecture et en écriture.
- une variable peut être déclarée dans une fonction. Dans ce cas c'est une variable locale (ou automatique) qui n'existe que si la fonction est exécutée. Elle cesse d'exister quand la fonction se termine.
- une variable paramètre est déclarée dans les paramètres de la fonction; elle sera traitée comme une variable automatique.

TODO

- [ ] Compilez et exécuter le code du programme
- [ ] Combien de variables globales, automatiques et paramètres comptez dans le programme ?
- [ ] Que se passe-t-il si la ligne de déclaration `void addition(int i, int j);` est retirée ?
- [ ] Si nous ne souhaitons pas utiliser la variable globale `registre` comment procéder pour récupérer le résultat de la fonction ?
- [ ] Complétez le programme pour traiter la multiplication et la division des entiers

Les tableaux

```c
// section pour inclure les en-tete
#include <stdio.h>
#include <stdlib.h>
//section pour delarer les fonctions et variables globales
#define TAILLE 5
int main();
void doesnotknow();
void affiche();
```

```c
// variable globale
int tableau[TAILLE];
//----------------------
//section pour realiser les fonctions
// la fonction main
int main(){
    int a = 0;
    int i = 0;

    for(int i = 0 ; i < TAILLE; i ++){
    printf("donnez un entier ?\n");
    scanf("%d",&a);
    tableau[i] = a;
    }
    affiche();
    tri();
    affiche();
}
void affiche(){
    int i ;
    printf("DEBUT TABLEAU\n");
    for(i=0;i<TAILLE;i++){
    printf("[%d] = %d \n",i,tableau[i]);
    }
    printf("FIN TABLEAU\n");
}
// la fonction de tri
void doesnotknow()
{
    int i = 0;
    int j = 0;
    int c = 0;
    for(j=0;j< TAILLE;j++)
    {
        for(i=0;i< TAILLE-1;i++){
        if ( tableau[i] > tableau[i+1] )
            {
                c = tableau[i];
                tableau[i] = tableau[i+1];
                tableau[i+1] = c;
            }
        }
    }
}
```

Dans un tableau nous allons pouvoir ranger plusieurs valeurs dans un espace

mémoire contigu. Pour accéder à une valeur nous allons utiliser un index. Cet index va calculer un déplacement à effectuer depuis l'adresse de départ.

TODO

- [ ] Compilez et exécutez le code proposé
- [ ] Que fait la fonction `doesnotknow` ?