

Programmation Langage C

Youssef ALJ

26 février 2020

- ① Concepts de base :
 - ① Forme d'un programme C (compilation, exécution).
 - ② Types de base, variables et opérateurs en C.
 - ③ Structures de contrôle.
 - ④ Lecture/écriture.
- ② Types composés :
 - ① Les tableaux.
 - ② Les structures.
 - ③ Le mot clé **typedef**.
- ③ Les pointeurs.
- ④ Les fonctions.
 - ① Définition, déclaration et appel d'une fonction.
 - ② Appel d'une fonction.
- ⑤ Gestion des fichiers.
- ⑥ Programmation modulaire.

Pré-requis :

- Initiation au langage C (cf cours du premier semestre).
- Mathématiques de base (Terminale S : un peu d'arithmétique).

Organisation :

- 15 % participation (rendu de TP).
- 15 % contrôle continu.
- 70 % examen

- Le langage C a été développé aux laboratoires AT&T dans les années 1970 par Dennis Ritchie.

Le cycle de création d'un programme en langage C est le suivant :

- 1 Concevoir un algorithme.
- 2 Utiliser un éditeur pour écrire le code source.
- 3 Compilation à partir du code source.
- 4 Édition des liens.
- 5 Éventuellement corriger les erreurs de compilation.
- 6 Exécuter le programme et le tester.
- 7 Éventuellement corriger les bugs.
- 8 Éventuellement Recommencer depuis le début.

On n'a pas besoin de mémoriser ces étapes. Ceci viendra avec la pratique.

Étape 1 : Créer un algorithme

Exemple : on veut écrire un algorithme qui affiche "bonjour tout le monde" à l'écran.

Début

Variables :

Écrire("Bonjour tout le monde")

Fin

Étape 2 : Saisie et sauvegarde du programme

- Avec un éditeur (par exemple notepad++) on saisit le texte suivant :

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("bonjour tout le monde");
6      return 0;
7  }
```

- On sauvegarde (souvent avec les touches CTRL-s).
- Durant la sauvegarde, on fait attention à ce que le nom du fichier finisse par '.c' pour qu'il soit reconnu comme étant un programme C.
- Y a-t-il des mots qui vous sont familiers ?

Étape 3 : Compilation

Le compilateur est un programme spécial qui :

- lit les instructions enregistrées dans le code source "bonjour.c"
- analyse chaque instruction.
- traduit ensuite l'information en langage machine compréhensible par le micro-processeur de l'ordinateur.

Étape 3 : Compilation (suite)

- Il existe plusieurs compilateurs :
 - gcc : GNU Compiler Collection.
GNU : acronyme récursif qui veut dire GNU's Not Unix.
 - Le compilateur : Microsoft Visual Studio.
 - ...
- On va utiliser le compilateur gcc. Voir TP.
- On compile en ligne de commandes windows (ou linux) via la commande suivante : `>gcc -c bonjour.c -o bonjour.o`
- Le résultat est la création d'un nouveau fichier appelé fichier objet qui a comme extension '.o' et qui a comme nom bonjour.o.

Étape 4 : Édition des liens

L'éditeur des liens (en anglais le linker) permet de :

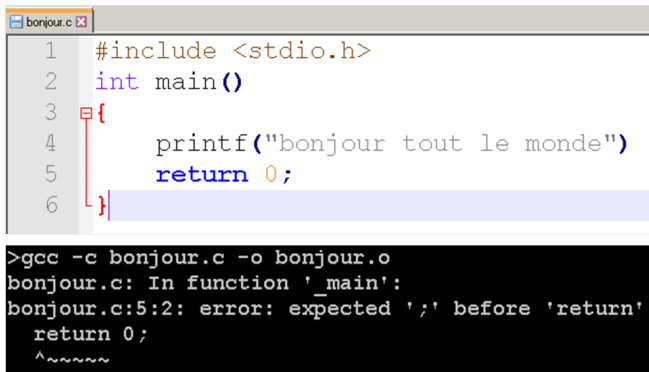
- faire le liens entre les différents fichiers ".o"
- cette étape de link est aussi faite en utilisant gcc.
`>gcc bonjour.o -o bonjour`
- on reviendra plus en détails sur le fonctionnement du linker dans les prochaines séances.

Fusion de la compilation et de l'édition des liens

- Plusieurs compilateurs font la compilation et l'édition des liens en même temps.
- Cela se fait avec la commande suivante :
`>gcc bonjour.c -o bonjour`

Étape 5 : Erreurs de compilation

- Plusieurs erreurs de compilations peuvent apparaître.
- Elles sont dues par exemple à une mauvaise syntaxe (oubli du ";", mot réservé mal écrit, etc.).
- Il faudra les corriger en éditant le fichier source, sauvegarder et recompiler puis ré-exécuter.

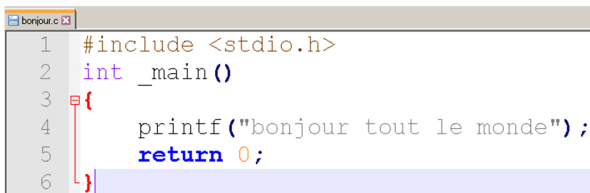


```
1  #include <stdio.h>
2  int main()
3  {
4      printf("bonjour tout le monde")
5      return 0;
6  }
```

```
>gcc -c bonjour.c -o bonjour.o
bonjour.c: In function '_main':
bonjour.c:5:2: error: expected ';' before 'return'
    return 0;
    ^~~~~~
```

Étape 5 : Erreurs de link

- Elles sont dues au fait que le linker n'arrive pas à faire le lien entre les fonctions définies par les programmes écrits.
- Une fonction principale doit toujours exister qui sert de point d'entrée aux autres fonctions.
- cette fonction principale est appelée la fonction main.



```
1 #include <stdio.h>
2 int _main()
3 {
4     printf("bonjour tout le monde");
5     return 0;
6 }
```



```
>gcc -c bonjour.c -o bonjour.o

>gcc bonjour.o -o bonjour
c:/mingw/bin/../../lib/gcc/mingw32/6.3.0/../../libmingw32.a(main.o):(.text.start
up+0xa0): undefined reference to `WinMain@16'
collect2.exe: error: ld returned 1 exit status
```

Étape 6 : Exécuter

- Si la compilation s'est effectuée sans erreur on verra nouveau fichier qui apparait.
- Ce fichier est appelé un exécutable.

```
>gcc bonjour.c -o bonjour
```

win

lin

```
>dir  
bonjour.exe  
bonjour.c
```

```
>ls  
bonjour  
bonjour.c
```

Étape 7 : Correction des bugs à l'exécution

- On utilise un programme appelé gdb.
- On reviendra sur la correction des bugs dans les prochaines séances.

Synthèse des principales étapes :

1

Edition

```
bonjour.c
1  #include <stdio.h>
2  int main()
3  {
4      printf("bonjour tout le monde");
5      return 0;
6  }
```

2

Compilation
+
Edition des liens

```
>gcc bonjour.c -o bonjour
```

win

```
>dir
  bonjour.exe
  bonjour.c
```

lin

```
>ls
  bonjour
  bonjour.c
```

3

Exécution

win

```
>bonjour
```

```
bonjour tout le monde
```

lin

```
>./bonjour
```

```
bonjour tout le monde
```

```
1  #include <stdio.h>
2  // ceci est un commentaire
3
4  int main()
5  {
6      // un autre commentaire
7
8      /*
9      Une autre facon
10     d ecrire un commentaire sur plusieurs lignes
11     */
12     printf("bonjour tout le monde");
13     return 0;
14 }
```


- Les commentaires sont un outil de base pour documenter son programme.
- Cette documentation n'est pas utile seulement pour les autres mais pour vous aussi.
- Les commentaires améliorent la lisibilité du code écrit.
- Sans commentaire la lecture est difficile.
- Il est donc recommandé d'ajouter des commentaires autant que possible.

Toujours par souci de lisibilité il est recommandé respecter les règles d'indentation :

- Décaler d'un cran (= trois espaces) vers la droite tout bloc inclus dans un précédent.
- Cela permet de repérer qui dépend de quoi et si tous les blocs ouverts sont

Exemple d'un programme C mal écrit

```
1  #include <stdio.h>
2
3  void main()
4  {
5      int a = 0;
6      if (a == 0)
7
8
9      {
10     printf("a est egal a 0");
11     }
12     else
13     {
14     printf ("a est different de 0");
15     }
16 }
```

Exemple d'un programme C mieux écrit

```
1  #include <stdio.h>
2  // fonction main indispensable pour chaque programme C
3  void main()
4  {
5      int a = 0;
6      // si a est nul on affiche : "a est nul"
7      if (a == 0)
8      {
9          printf("a est egal a 0");
10     }
11     // sinon on affiche : "a n est pas nul"
12     else
13     {
14         printf ("a est different de 0");
15     }
16 }
```

Manipulation d'informations

- Quand on programme avec n'importe quel langage de programmation on veut utiliser des données.
- On veut stocker ces données dans ce qu'on appelle des variables.
- On veut programmer un jeu vidéo.
- On a un joueur de ce jeu vidéo qui veut manipuler un personnage.
- Ce personnage va avoir une position dans une carte.
- Cette position va changer en fonction du mouvement du personnage.
- On veut stocker cette information pour pouvoir interagir avec d'autres éléments du jeu.

Il y en a six : **void**, **int**, **char**, **float**, **double**, **long double**.

- **void** : c'est le type vide. Il est surtout utilisé pour définir les fonctions sans arguments ou sans valeur de retour.
- **int** : c'est un type qui se décline sous plusieurs formes.
 - **short** : un entier court.
 - **long** : un entier long.
 - **signed** pour dire que l'entier qu'on manipule peut être uniquement positif.
 - **unsigned** pour dire que l'entier qu'on manipule peut être positif ou négatif.Si on n'indique rien, le qualificatif **signed** est appliqué.

Les types de bases

- **char** : ce type permet de stocker les caractères. Il peut aussi représenter les entiers sur 8 bits. On, peut aussi trouver un char signé **signed char** ou non signé **unsigned char**.
- **float** : ce type permet de représenter les réels.
- **double** : même chose que **float** mais avec une précision plus importante.
- **long double** : pareil que **double** mais avec une précision encore plus grande.

- La fonction printf est très utile car permet d'afficher des messages et les valeurs des variables.

- Exemple :

```
1 printf("Bonjour tout le monde!");  
2 printf("%d kilogramme vaut %d grammes", 1, 1000):
```

- L'argument de la fonction printf dit **format** est une chaîne de caractère qui détermine ce qui sera affiché par printf et sous quelle forme.
- Dans l'exemple 1 c'est "Bonjour tout le monde!".

Exemple

```
1 printf("Bonjour tout le monde!");  
2 printf("%d kilogramme vaut %d grammes", 1, 1000):
```

- Dans l'exemple 2 c'est "%d kilogramme vaut
- Dans l'exemple 2 : la chaîne de caractère qu'on veut afficher est composée d'un texte normal et de séquence de contrôle permettant d'inclure des variables.
- le premier %d sera donc remplacé par la valeur 1.
- le deuxième %d sera remplacé par la valeur 1000.
- On aura comme résultat en sortie l'affichage suivant :
"1 kilogramme vaut 1000 grammes".

Les séquences de contrôle commencent par le caractère % suivi d'un caractère parmi les suivants :

- d ou i pour afficher un entier signé.
- f pour afficher un réel (float ou double).
- c pour afficher comme un caractère.
- s pour afficher une chaîne de caractères.

- Cette fonction permet de lire des données formatées à partir de l'entrée standard (clavier).
- `scanf` utilise les mêmes formats que `printf` mais on fait précéder le nom de la variable du caractère `&`.

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

- seul le format est passé en paramètre.
- Il ne faut ajouter de message ni aucun autre caractère.

Exemple

```
1  /* Exemple pour tester "scanf" */
2  #include <stdio.h>
3  int main () {
4      int nb1 ;
5      float nb2 ;
6      printf("Saisissez une valeur entiere (positive ou
           negative) pour nb1 : ") ;
7      scanf("%d",&nb1) ;
8      printf("Saisissez une valeur reelle pour nb2 : ") ;
9      scanf("%f", &nb2) ;
10     printf("nb1 vaut %d ; nb2 vaut %f\n", nb1, nb2) ;
11     return 0;
12 }
```

- Éditer, sauvegarder puis compiler ce code.
- Faites en sorte que le programme affiche "hello monde" après avoir affiché les valeurs saisies par l'utilisateur.