Partie I : Initiation au langage C

HISTORIQUE

- Le langage C a été mis au point au début des années 1970 et normalisé (ANSI) en 1988.
- Depuis cette date le langage C a conquis le monde des entreprises et des universités malgré ses quelques défauts (syntaxe parfois illisible, pas assez de contrôle sémantique …).
- C'est un langage de haut niveau qui génère un code très rapide grâce à un compilateur très performant.
- C'est un langage tolérant, permissif !

PORTABILITE

- Un programme écrit en langage C peut être exécuté sur n'importe quel système d'exploitation sans aucune modification. C'est pour cela qu'on dit qu'il est portable.

APPRENTISSAGE

- Le langage C n'est pas difficile à apprendre malgré son apparence car il dispose de peu d'instructions et ses structures de données sont limitées.
- Comme ça a été dit le langage C n'est pas difficile, il est différent !
- Une fois l'apprentissage du langage C est achevé, cela nous facilite la familiarisation avec d'autres langages tels que C++ et Java qui sont dérivés du C.

QUELQUES CONCEPTS DE BASE

- Narration est un langage très simple qu'on apprendra avant l'algorithme, il nous permet de voir les différentes étapes pour solutionner un problème donné. Voir cours sur Narration.
- Algorithme est un outil nécessaire pour n'importe quel programmeur (informaticien ou autre) pour avoir la suite logique des instructions pour solutionner un problème donné. Avant on utilisait des organigrammes.
- On peut dire qu'un algorithme est le résultat de la décomposition d'un problème complexe en opérations élémentaires qui seront exécutées en plusieurs étapes successives. C'est le raffinement d'un problème.
- Pour obtenir un programme il suffit de traduire l'algorithme obtenu dans n'importe quel langage de programmation. C'est pour cela qu'un algorithme doit être écrit indépendamment du langage de programmation.

TYPOLOGIE DES LANGAGES

- Langages machine utilisant que des 0 et des 1 (langage binaire). Il est compris directement par la machine d'où son nom.
- Langages d'assemblage utilisant des mnémoniques (un semblant de mot) donc il n'est pas compris directement par la machine par conséquent il lui faut un traducteur (un outil logiciel) qu'on appelle « Assembleur ».
- Langages évolués ou de haut niveau, pour qu'ils soient compris par la machine ils doivent passés par l'étape de la compilation. C'est pour cela qu'on parle par exemple du compilateur C.

COMPILATION

- Tout langage évolué possède un compilateur, celui-ci sert à traduire le « code source » d'un programme écrit en un langage de haut niveau comme le C en « code objet » exécutable par un ordinateur.
- Avant la traduction le compilateur passe par l'étape d'analyse du programme source (Analyse syntaxique et analyse sémantique).

PRE-PROCESSEUR

- Il est aussi appelé pré-compilateur c'est un utilitaire qui traite le fichier source avant la compilation.
- Tous les langages évolués ne possèdent pas ce pré-compilateur, le langage C en possède un.

ASSEMBLEUR

- Le code généré par la compilation est traité pour générer un fichier en format relogeable (0 & 1). Ce fichier généré n'est pas encore exécutable, il reste encore l'étape de l'édition de lien surtout si votre programme est composé de plusieurs modules.

EDITEUR DE LIEN

- L'éditeur de lien prend le ou les fichier(s) en format relogeable et fait le lien entre eux pour créer un programme chargeable c'est-à-dire exécutable.

DIFFERENTES ETAPES DE CONSTRUCTION D'UN PROGRAMME

- On construit un programme en passant par plusieurs étapes.
  - **Etape 1 :** On pose un problème donné dans une langue donnée (Français par exemple)
    Analyse (analyse préalable)
  - **Etape 2 :** On écrit les différentes étapes de résolutions
    Formalisme
  - **Etape 3 :** On écrit une narration, un organigramme ou bien un algorithme
    Traduction
  - **Etape 4 :** On traduit notre algorithme en un langage de programmation (C par exemple)
- Une fois le programme est écrit en un langage évolué, on passe aux étapes suivantes :
  - **Etape 1 :** compilation du programme source
  - **Etape 2 :** exécution du programme compilé

NB : D'une manière générale, un programme (logiciel) est composé de plusieurs modules (sous programmes). Dans ce cas, il est nécessaire de lier ces différents fichiers entre eux après la compilation : c'est l'édition de lien.

STRUCTURE D'UN PROGRAMME C

- Tout comme la narration et l'algorithme, le programme C a une structure (un squelette).
- Le langage C est un langage structuré et un programme C de base est constitué de :
  - Partie entête (Directive)
  - Programme principal (Fonction principale)
  - Partie traitement (Corps du programme = déclarations + instructions)

EXEMPLE DE BILAN

On essaie d'écrire une narration puis un algorithme correspondant et sa traduction en langage C. Le problème est la somme de deux nombres entiers.

➢ Ecriture de la narration

Narration somme

Objet A, B, S

1- Lire A
2- Lire B
3- Calculer S = A+B
4- Ecrire S
5- Arrêter

➢ Ecriture de l'algorithme correspondant

```
Algorithme  somme ;
Déclaration
A, B, S : entier ;
Début
        Lire(a) ;
        Lire(b) ;
        S ←A+B ;
        Ecrire(S) ;
Fin.
```

➢ Ecriture du programme en C

```
#include <stdio.h>          /* Partie entête*/
Main()                      /*Programme principale*/
{
        Int A, B, S ;       /*Déclaration*/
        Scanf("%i" , &A);   /*Lecture de A*/
        Scanf("%i" , &B);   /*Lecture de B*/
        S = A+B ;           /*Calcul de S*/
        Printf("%i" , S);   /*Ecriture de S*/
}
```

Remarque : La parenthèse ouvrante **{** joue le rôle de **Début** et la fermante **}** celui de **Fin**.

Essayons de comprendre ce programme sans trop rentrer dans les détails.

- Ce programme principal **main()** utilise dans la partie traitement (corps du programme) deux différentes opérations **scanf()** et **printf(),** ce sont des opérations (fonctions) d'entrée/sortie (input/output),il utilise aussi une déclaration **int A,B,S ;** et une affectation **S = A+B ;**
- Les deux dernières opérations (déclaration et affectation) sont reconnues par le compilateur par contre les opérations d'input/output ne le sont pas, donc il faut les inclure à l'aide de la directive **#include** qui précise au compilateur dans quel fichier se trouve la définition de ces fonctions.
- Que signifie **stdio.h** ?
  - **std** : Standard
  - **io** : input/output
  - **.h** : header (entête)
- L'inclusion qui est une opération appartenant aux instructions du pré processeur se fait comme suit : **#include <stdio.h>**
- Que signifie **%i** et **&** dans les opérations d'input/output **scanf()** et **printf()** ?
  - **%i** : veut dire que c'est le format entier (i comme integer), %i est équivalent à **%d**
  - **%c :** Pour les caractères (c comme character).
  - **%f :** Pour les réels (f comme float).
  - **&** : cet opérateur doit précéder toute variable de type **scalaire** (entier, réel, caractère …).

Remarques

- Attention un mot écrit en minuscule est différent de celui qui est écrit en majuscule. Si vous écrivez **main()** et **Main()** ce n'est pas la même chose.
  L'écriture du programme C doit être en minuscule !
- Le langage C permet la déclaration des variables à n'importe quel endroit dans le programme, il n'a pas une zone précise appelée zone déclarative. Il est permissif.
- Par contre la déclaration des constantes est dans l'entête (partie du pré compilateur), à l'aide de **#define**.
- En C, toute déclaration et/ou toute instruction se termine par un point-virgule **;** sauf les instructions destinées au préprocesseur comme par exemple #include …
- A chaque fois qu'on utilise une fonction standard, on doit inclure son fichier entête !

BIBLIOGRAPHIE

1- Langage C : Cours et références
Pierre NERZIC Mars 2003

2- Langage C : Support de cours
Patrick CORDE Mars 2006

3- Programmation en langage C
M.C. BELAID