La programmation Impérative par le Langage C

1. Introduction

1.1 Qu'est-ce que la programmation impérative ?

Un programme est constitué de plusieurs lignes d'*instructions*. Chaque instruction permet d'effectuer une *tâche*. L'exécution du programme permet d'exécuter ces tâches.

En programmation impérative, les tâches (ou instructions) sont exécutées les unes après les autres. On dit que les instructions sont exécutées de manière *séquentielle*.

L'ordinateur commence l'exécution du programme dans un certain état initial, que chaque instruction exécutée modifie, jusqu'à atteindre un état final qui contient le résultat voulu.

Ci-dessous quelques langages de programmation impérative :

Assembleur	C++	Perl	Java	Pascal
C	Objective C	PHP	Fortran	Common Lisp

1.2. Autres types de langages

La programmation déclarative ne dépend d'aucun état interne de l'ordinateur. Elle décrit le problème contrairement à la programmation impérative qui décrit plutôt sa solution.

Il existe plusieurs types de programmation déclarative :

- Programmation logique :
  <Description> Permet de se concentrer sur ce qui doit être calculé sans se soucier de la manière dont le calcul est effectué. Applique des règles de logique dont on associe des conséquences plus ou moins directes. Ce type de langage est essentiellement orienté Intelligence Artificielle.

<Exemples de langages> Prolog, pypy, Oz

- Programmation fonctionnelle :
  <Description> Permet d'effectuer des calculs de fonctions. La notion d'affectation n'existe pas, mais correspond à une égalité qu'aucune fonction ne pourra changer tout comme en mathématiques. La notion de récursivité est très répandue dans ce type de programmation. Les langages de ce type son essentiellement utilisés pour des calculs mathématiques. De ce fait, chaque programme retourne un seul résultat comme la notion de fonction en mathématiques.

  <Exemples de langages> Oz[1], Lisp, Common Lisp, Scheme, OCaml

- Programmation descriptive :
  <Description> C'est une programmation permettant de décrire des structures de données. Les langages de ce type sont à expressivité réduite.

  <Exemples de langages> HTML, XML, LaTeX

- Programmation par contrainte :
  <Description> Permet de poser un problème sous forme de relations logiques entre plusieurs variables. Trouver la solution à un problème de programmation par contraintes (PIC) consiste à décider s'il existe ou non une affectation de toutes les variables telles que toutes les contraintes du problème soient satisfaites. Exemple de problème : le voyageur de commerce.

  <Exemples de langages> Module Python : Python Constraint,
  Librairies Java : Jopt, choco solver, Cream

2. Le langage C

Quelques propriétés du langage C :

- **Langage compilé** : le code source d'un programme écrit en C, doit être *compilé* par un *compilateur* (ex. `gcc`), pour ensuite être exécuté. La compilation génère un fichier binaire qui lui sera exécuté.

- **Langage sensible à la casse** : une variable `a` diffère d'une variable `A`.

- **Une instruction de code se termine toujours par un " ; ".**

- **Un fichier de programme C, a pour extension .c**.

- **Une librairie C a pour extension .h**.

- **Pour commenter une ligne d'instruction il faut la précéder de " // ".**

- **Les commentaires sont compris entre les caractères " /* " et " */ ".**

- **Une fonction f s'écrit en langage C comme suit :**

```
type              f(liste_paramètres){
     instruction1 ;
     …
     instructionN ;
}
```

- **Mots clé[1] du langage C :**

```
auto      default   float    register  struct    volatile
break     do        for      return    switch    while
case      double    goto     short     typedef char      else
```

[1] Liste non exhaustive

```
if      signed   union const    enum     int     sizeof
unsigned conitnue  extern   long     static   void
```

En général, vous devriez vous efforcer de rendre votre code aussi portable que possible, sans quoi vous risquez de devoir récrire entièrement votre programme par la suite pour qu'il fonctionne ailleurs.

```
% gcc –W -Wall -pedantic -o foobar foobar.c
```

Cela générera un exécutable *foobar* après avoir vérifié que *foobar.c* respecte la norme.

Pour plus d'options voir la documentation de gcc : `$man gcc`

2.3. La structure d'un programme C

Un programme C peut être composé de plusieurs fichiers :

- Un fichier principal contenant la fonction main().
- Librairies standard C.
- Librairies spécifiques au programme. La création de ce type de librairies nécessite la création d'un fichier portant le nom de la librairie avec l'extension .c et un fichier du nom de la librairie avec l'extension .h. Nous expliquons dans ce qui suit le contenu de chacun de ces fichiers.

Le fichier principal du programme C suit la structure ci-dessus, à savoir en début de fichier des commentaires doivent décrire « ce que fait le programme dans sa globalité ». D'autres informations peuvent être précisées comme le nom de chaque développeur ayant participé à la création du projet. La date de création, la date de dernière modification ou encore la version du programme.

main.c

```c
/*
        Description du programme
        nom developpeur(s)
        date de creation
        date de dernière modification
        version
*/

#includes "mes_librairies"
#includes <librairiesC>

/* commandes preprocesseur */

/* definitions macros privees              */
/* definitions constantes privees          */
/* definitions types prives                */
/* definition structures privees           */
/* definition variables globales privees   */
/* definition fonctions privees            */
/* definition fonctions publiques          */
/* definition variables globales publiques */

int main(){
        /* declaration (variables, fonctions locales) */
        /* prototypes de fonctions */
        /* instructions          */
        return 0;
}
```

BibNonStandardC.h

```c
/* commandes preporcesseur                */
/* declaration macros                */
/* declaration constantes            */
/* declaration types                 */
/* declaration structures            */
/* declaration  fonctions            */
/* declaration  variables globales   */
```

BibNonStandardC.c

```c
/* inclusion headers necessaires */
/* definition  fonctions  */
```

Notez que seul *main.c* doit contenir la fonction `main()`. Il est strictement interdit de faire appel à cette fonction dans les autres fichiers .h ou .c dans un même projet. Ainsi, nous avons la règle suivante :

un projet = un fichier principal contenant int main().

2.4. Un exemple de programme

Vous pouvez écrire un programme C dans un éditeur de votre choix : Emacs ou vi (ou tout autre). Nous donnons ci-dessous le contenu d'un programme nommé *toto.c*.

```c
#include <stdio.h>

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

toto.c

Ce programme permet d'afficher à l'écran :

```
"Hello World !
"
```

L'éditeur vi que nous avons utilisé ci-dessus permet la coloration syntaxique du code. Ce qui signifie que les mots clé du langage C sont colorés de manière à permettre une meilleure lisibilité du code.

La norme prévoit que tout programme C débute toujours par la fonction `main()`. Tout programme C doit donc contenir une et une seule fonction `main()`.

2.5. La directive #include

La directive `#include` est une commande particulière en C qui permet de charger dans le programme des librairies standard du langage. Dans l'exemple donné cidessus, la librairie chargée est *stdio.h*.

Notons deux choses :

- Le nom de la librairie est compris entre les caractères "<" et ">". Ceci doit toujours être de règle pour les librairies standard du C dont fait partie *stdio.h*. Lorsqu'on inclue une librairie définie par le programmeur, l'instruction s'écrira : `#include "nomMaLib.h"`.
- Il n'y a pas de " `;` " à la fin de l'instruction `#include`. En effet, toute directive en langage C ne doit pas être suivie d'un " `;` ".

2.6. La fonction main()

La fonction `main()` est une fonction réservée en C. La norme impose d'écrire cette fonction comme suit :

```
int main() { instruction(s) }           /* forme usuelle    */
```

```
int main(int nbargs, char *argv[])      /* forme permettant de
{ instruction(s) }                      récupérer des
                                        informations fournies
                                        par l'environnements */
```

Ainsi, la fonction `main()` doit toujours être déclarée de type `int` (i.e. entier).

2.7. La fonction printf()

Cette fonction permet d'afficher une chaine de caractères à l'écran.
L'appel à cette fonction ne peut se faire qu'après l'inclusion de la librairie *stdio.h* dans laquelle elle est définie. Nos verrons plus en détail l'utilisation de cette fonction et autres fonctions de cette librairie.

2.8. La fonction return

La fonction return permet d'évaluer une expression, de la convertir dans le type de la valeur de retour tel qu'il est indiqué dans l'entête et de la fonction.

Dans cet exemple la fonction return est écrite sans parenthèses, cela est correct particulièrement pour cette fonction, mais non pour les fonctions que vous écrirez par la suite.

Il est équivalent d'écrire `return 0;` et `return(0);` ou encore `return (0);`