TP 1 : Ecriture, compilation et exécution de programmes C simples

Objectifs :

- Découvrir la syntaxe et la sémantique du langage C.
- Etre capable d'écrire en C un algorithme simple.

Stéphanie CHOLLET – Ioannis PARISSIS

El Mehdi KHALFI – Ayoub SOURY

2015/2016

Syntaxe du langage C

Ces travaux pratiques sont à effectuer sous Linux.
Pour créer/éditer des programmes, utiliser un éditeur tel que gedit
(menu Accessoires>Editeur de texte>gedit)).
Pour lancer une compilation ou exécuter votre programme, utiliser un terminal (Accessoires>Terminal).
On se réfère au polycopié « Introduction au langage C » de B. Cassagne.

1. Introduction

1.1. Mon premier programme C

Créer un fichier prog1.c et y ajouter le texte suivant :

```
#include <stdio.h>
int main() {
     printf("Bonjour\n"); /* \n signifie " passage à la ligne " */
     return(0) ;
}
```

Ensuite exécuter la commande suivante :

```
gcc prog1.c -Wall -ansi -pedantic –o prog1
```

Vous venez de compiler votre premier programme C. A l'issue de l'exécution de cette commande, le fichier `prog1` a été ajouté dans votre répertoire courant. Il s'agit du programme exécutable correspondant à `prog1.c`. Pour exécuter ce programme, taper simplement :

```
./prog1.
```

1.2. Notions de base du langage C : types, constantes, variables, expressions, instructions conditionnelles

Ce paragraphe est un résumé très compact du polycopié, chapitre 1, sections 1.1 à 1.13, dont la lecture est nécessaire.

Les *identificateurs* du langage (noms de variables, fonctions etc.) sont des suites formées de lettres, chiffres (pas en première position) et du caractère '_' (p.ex. bonjour, b3jour, bon_jour, _bonjour).

Un *commentaire* est compris entre /* et */ (/* Ceci est un commentaire */ et ceci une faute de syntaxe */).

Les *types de base* du langage sont : caractère (char), entier (int, short int, long int), flottants (float, double, long double).

L'écriture de *valeurs constantes* se fait naturellement pour les cas simples : 2, 2.6, 'a', « toto ». De nombreuses écritures spécifiques sont disponibles (p.ex. 4L désigne l'entier 4 représenté comme un entier long ; '\n' correspond au caractère *newline*…) à découvrir dans

le polycopié. Le moyen le plus simple de définir une *constante nommée* est d'utiliser la directive `#define` :

```
#define PI 3.14159 /* sans ';' à la fin */
```

Cette ligne fait de sorte que toute occurrence de l'identificateur `PI` dans le programme est remplacée par `3.14159`.

L'affectation en C s'effectue à l'aide de l'opérateur = (égal).

```
x = 5 ; /* affectation à x de la valeur 5 */
```

Une particularité du langage C est qu'une affectation est une expression retournant une valeur :

```
int k = 1 ;
int i = (j = k) + 1 ; /* j prend la valeur de k, soit 1 ; i
prend la valeur 2, soit la valeur de (j = k) + 1 */
```

Les opérateurs de comparaison sont == (égalité), >, <, >=, <=, != (différence).

L'instruction conditionnelle `if` :

```
if(condition) {
     instruction
}
/* noter la parenthèse obligatoire autour de la condition*/
if(condition) {
     instruction
} else {
     instruction
}
```

Une instruction peut prendre la forme d'un *bloc d'instructions* délimitées par des accolades (qui correspondent à « début » et « fin » en langage algorithmique).

```
if(x > 0) {
   x = x+1 ;
   y = 0 ;
} else {
   x = y = 1;
}
```

L'instruction de choix `switch` (voir polycopié 3.15.1) :

```
/* x, y sont des int */
switch (x) {
     case 0: y = 1; break;
     case 1: y = 2; break;
     default: y = 0;
}
```

1.3. Entrées sorties élémentaires

L'affichage à l'écran ou dans un fichier, la lecture depuis le clavier ou depuis un fichier se font au moyen d'appels à des fonctions des bibliothèques de C (en d'autres termes, il n'existe pas d'instruction de base du langage permettant d'effectuer une action d'entrée ou sortie). Avant une étude plus approfondie de ces fonctions, on se contente ici d'utiliser deux

d'entre elles, `printf` et `scanf`. Leur utilisation est rapidement expliquée dans les sections 1.16 et 3.14 du polycopié. L'exemple ci-dessous en illustre quelques utilisations simples :

```
#include <stdio.h>

#define INC 2

int main() {
   int i;
   printf("Donnez un entier : ");
   scanf ("%d", &i); /* &i : i par adresse (voir plus loin) */
   i = i + INC ;
   printf("valeur de i=%d et son successeur i+1=%d\n", i,
i+1) ;
}
```

Exercices

Exercice 1 : Appréciation

Écrivez un programme qui lit une note exprimée sous la forme d'un chiffre entre 1 et 5, puis affiche un message correspondant à la lettre saisie (« Très bien », « Bien », « Assez Bien », « Passable » et « Insuffisant »). Faire trois versions de programme :

- Une version utilisant des instructions `if` imbriquées ;
- Une version utilisant des `if` en séquence ;
- Une version utilisant l'instruction `switch`.

Exercice 2 : Année bissextile

Une année bissextile comprend un jour de plus que les années normales. On dit couramment que les années bissextiles reviennent tous les quatre ans, ce qui n'est pas tout à fait exact. La définition complète est la suivante :
Une année bissextile est divisible par 4 ; si elle est également divisible par 100, alors elle est aussi divisible par 400.

1. Écrivez un programme qui lit un entier correspondant à une année et affiche un message approprié en fonction (« L'année 1996 est bissextile », « L'année 2001 n'est pas bissextile »).
2. Testez ce programme en lui fournissant différentes années en entrée. Comment choisissez-vous les valeurs d'entrée? Ajouter, sous forme de commentaire dans le programme, les tests effectués avec leur justification.

1.4. Itérations

Il existe trois instructions permettant de réaliser des itérations en langage C : `while`, `do`, `for` (lire les sections 2.2 et 2.3 du polycopié).

```
/* condition est une expression booléenne */
while(condition) {
     instruction
}
```

```
do {
     instruction
} while (condition) ;
```

```
for(expression1 ; expression2 ; expression3) {
     instruction
}
```

Exercices

Exercice 1 : Somme des n premiers entiers

Ecrire un programme calculant la somme des n premiers entiers. La valeur de n est fournie par l'utilisateur. Réaliser une version de ce programme utilisant l'instruction `while` et une autre utilisant `do`.

Exercice 2 : Opérateurs de post et pré incrémentation et décrémentation

Exécuter et commenter le programme suivant :

```
#include <stdio.h>

int main() {
     int i, j, k, l;

     i = j = k = l = 0;

     while(i < 9) {
          printf("i++ = %d, ++j = %d, k-- = %d, --l = %d\n",
                  i++, ++j, k--, --l);
     }

     printf ("i = %d, j = %d, k = %d, l = %d\n", i, j, k, l);
}
```