Université Paris-Nord — L1
Institut Galilée — Année 2010-2011
Licence 1 - 1er semestre

Travaux dirigés 4 : la structure de contrôle `for`

L'objectif de ce TD est de vous familiariser avec la notion d'itération en programmation. On parle communément de "boucle". Cette notion sera illustrée sur des problèmes de comptage et de répétition d'actions.

1 Itération : l'instruction `for`

1.1 Rappel

Soit le programme suivant :

```
1   /* déclaration de fonctionalités supplémentaires */
2   #include <stdlib.h> /* EXIT_SUCCESS */
3   #include <stdio.h> /* printf */
4
5   /* déclaration constantes et types utilisateurs */
6
7   /* déclaration de fonctions utilisateurs */
8
9   /* fonction principale */
10  int main()
11  {
12      /* déclaration et initialisation variables */
13      int i; /* var. de boucle */
14
15      for(i = 0;i < 5;i = i + 1)
16      {
17          printf("i = %d\n",i);
18      }
19      /* i >= 5 */
20
21      printf("i vaut %d après l'exécution de la boucle.\n",i);
22
23      return EXIT_SUCCESS;
24  }
25
26  /* implantation de fonctions utilisateurs */
```

1. Quelle est la signification de chaque argument du `for` ? Quelles instructions composent le corps de la boucle ?
2. Faire la trace du programme. Qu'affiche le programme ?
3. Modifiez le programme afin que la séquence affichée soit exactement (on passera à la ligne juste avant la fin du programme) :
   - `0 1 2 3 4`
   - `1 2 3 4`
   - `1 2 3 4 5`
   - `1 3 5`

– `(0,0) (1,1) (2,2)`.

4. Modifiez le programme afin que la séquence affichée soit :
   – `0 1 2 0 1 2`.
   – `0 1 2 0 1 2 3`.
   De combien de boucles avez-vous besoin ? De combien de variables de boucles avez-vous besoin ?
5. Modifiez le programme afin que la séquence affichée soit :
   – `(0,0) (0,1) (0,2) (1,0) (1,1) (1,2) (2,0) (2,1) (2,2)`.
   De combien de boucles avez-vous besoin ? De combien de variables de boucles avez-vous besoin ? Quelle est la différence de structuration des boucles entre le point 4 et le point 5 ?

1.2 Exercice type : calcul de $\Sigma_1^n i$

Écrire un programme qui calcule et affiche la somme des entiers de 1 à n : $\Sigma_1^n i$. n est un entier quelconque.

1.3 Affichage de n fois "Coucou"

Écrire un programme qui affiche n fois la chaîne de caractères `"Coucou\n"`.

1.4 Calcul de la somme d'une série d'entiers saisie par l'utilisateur

Écrire un programme qui demande à l'utilisateur combien d'entiers composent sa série, lit la série d'entiers et affiche la somme des valeurs de la série.

Rappel : l'instruction `scanf("%d", &a)` permet de réaliser une saisie utilisateur d'un entier dont la valeur sera affectée à la variable `a` (comme toute variable, `a` doit être préalablement déclarée).

Université Paris-Nord — L1
Institut Galilée — Année 2010-2011
Licence 1 - 1er semestre

Travaux pratiques 4 : la structure de contrôle "for"

L'objectif de ce TP est de vous familiariser avec les notions d'itération et de boucles imbriquées.

Vous allez mettre tous vos programmes écrits dans ce TP dans le répertoire TP4.

1. À partir du début de votre arborescence, créez le répertoire TP4 : `mkdir TP4`
2. Allez dans ce répertoire pour y mettre des fichiers : `cd TP4`

L'étape suivante est à répéter pour chaque nouveau programme (exo1, exo2 etc..) :

3. Créez un nouveau fichier source pour le langage C ou une nouvelle copie d'un programme existant.

   Création `gedit exo1.c &` (vous pouvez utiliser `emacs` ou `kwrite` au lieu de `gedit`)

   Copie Il est plus rapide de repartir d'une copie de votre programme `bonjour.c` du TP2 pour éviter de retaper tout le squelette. Dans le terminal :
   ```
   cp ../TP2/bonjour.c exo1.c
   gedit exo1.c &
   ```
   Vous pouvez-aussi ouvrir `bonjour.c` et utiliser la fonction *Enregistrer sous...* de votre éditeur mais attention à enregistrer la nouvelle copie dans le bon répertoire.

Vous pouvez utiliser à tout moment la commande `ls` (list directory) pour voir la liste des fichiers d'un répertoire.

Les trois étapes suivantes seront à répéter autant de fois que nécessaire pour la mise au point de chaque programme (apprenez à utiliser les raccourcis clavier).

4. Après avoir fini d'écrire votre programme, enregistrez le.
5. Créez un programme exécutable à partir de votre fichier source :
   `gcc -Wall exo1.c -o exo1.exe`
6. Quand l'étape précédente a réussi (il faut lire attentivement les messages affichés), exécutez le programme pour vérifier qu'il fonctionne : `exo1.exe` (ou `./exo1.exe`).

1 Affichage de figures géométriques

Les exercices suivants utilisent le caractère `*` (étoile) pour dessiner des figures géométriques simples, appelées figures d'étoiles.

1.1 Exercice type : affichage d'un rectangle d'étoiles

Écrire un programme qui, étant données deux variables, `longueur` et `largeur`, initialisées à des valeurs strictement positives quelconques,affiche un rectangle d'étoiles ayant pour longueur `longueur` étoiles et largeur `largeur` étoiles. Deux exemples d'exécution, avec deux initialisations différentes, sont les suivants :

```
Affichage d'un rectangle d'etoiles de longueur 10 et largeur 5.
**********
**********
**********
**********
**********
```

```
Affichage d'un rectangle d'etoiles de longueur 6 et largeur 3.
******
******
******
```

1.2 Exercice type : affichage d'un demi-carré d'étoiles

Écrire un programme qui affiche, étant donnée la variable, `cote`, initialisée à une valeur quelconque, un demi-carré d'étoiles (triangle rectangle isocèle) ayant pour longueur de côté `cote` étoiles. Deux exemples d'exécution, avec deux initialisations différentes, sont les suivants :

```
Affichage d'un demi-carre d'etoiles de cote 6.
*
**
***
****
*****
******

Affichage d'un demi-carre d'etoiles de cote 2.
*
**
```

1.3 Affichage d'un demi-carré droit d'étoiles (optionnel)

Écrire un programme qui affiche un demi-carré droit d'étoiles de côté spécifié par l'utilisateur. Exemple d'exécution :

```
Entrer la taille du demi-carré :
5
Affichage d'un demi-carre droit d'etoiles de cote 5.
    *
   **
  ***
 ****
*****
```