Le langage C

Jean-Sébastien Coron

Université du Luxembourg

1 Les boucles

Boucles conditionnelles

- Il est possible de répéter plusieurs fois le même bloc d'instruction, en fonction du résultat d'un test.
- Instruction `while (test) instruction`
  - L'instruction s'exécute tant que le résultat du test est vrai.
  - L'instruction n'est pas exécutée si le résultat du test est faux la première fois.
- Instruction `do (instruction) while (test)`
  - L'instruction est exécutée une fois, puis répétée si le résultat du test est vrai.
  - L'instruction est exécutée au moins une fois.

Instruction `while`

- Exemple: affiche les entiers de 1 à 10.

```
#include <stdio.h>
int main()
{
  int i=1;
  while(i<=10)
  {
    printf("%d\n",i);
    i=i+1;
  }
}
```

Instruction `do while`

- Exemple: on demande à l'utilisateur d'entrer un entier qui doit être positif, sinon on repose la question.

```
#include <stdio.h>
int main()
{
  int a;
  do
  {
    printf("Entrez un nombre positif.\n");
    scanf("%d",&a);
  } while (a<0);
}
```

Boucle for

- Une boucle `for` permet de répéter une instruction plusieurs fois, à l'aide d'une variable de contrôle.
  - `for(init;test;itération)  opération;`

  - `init`: initialiser la variable de contrôle.
  - `test`: test de la variable de contrôle
  - `itération`: opération sur la variable de contrôle
  - `opération`: opération de la boucle
- Exemple: afficher les entiers de 1 à 10.
  - ```
    for (i=1;i<=10;i=i+1)
    printf("%d\n",i);
    ```

Boucle for

- Syntaxe :
  - `for(init;test;itération)  opération;`

- Séquence d'opération :
  - ```
    init
    test:si test=faux, saut à fin du for
    opération
    itération
    retour à test
    fin du for
    ```

- Exemple: somme des entiers de 1 à 10 :
  - ```
    s=0;
    for (i=1;i<=10;i=i+1)  s=s+i;
    ```

Variable de contrôle

- On peut aussi décrémenter la variable :
  - Boucle affichant les nombres de 10 à 0 :
    ```
    for (i=10;i>=0;i=i-1)
    printf("%d\n",i);
    ```
- Explication :
  - `i=10`: valeur initiale de la variable `i`
  - `i>=0`: la boucle continue tant que `i>=0`
  - `i=i-1`: après chaque exécution de l'opération de la boucle, la variable `i` est décrémentée.
    ```

Boucle For

- Exemple :
  - `for (i=0;i<6;i=i+1)`
    `printf("%d\n",i);`

  - La boucle affiche `0,1,2,3,4,5`.
  - Elle commence à `i=0`, vérifie que `i<6`, et affiche `0` à l'écran.
  - La boucle continue avec `i=1,2,3,4,5`.
  - Lorsque `i=6`, la condition `i<6` n'est plus réalisée, et on sort de la boucle.
    * La valeur `6` n'est pas affichée.
    * La boucle s'est exécutée en tout 6 fois, pour `i=0,1,2,3,4,5`

Exemple

- On veut calculer $a \cdot b$ en calculant

$$a \cdot b = b + \ldots + b$$

- Avec une boucle :

```
int mult(int a,int b)
{
  int i,c=0;
  for(i=0;i<a;i++)
  {
    c=c+b;
  }
  return c;
}
```

1) On définit $c = a^b$ (dire *a puissance b* ou *a exposant b*) comme $c = a * \cdots * a$, l'entier a apparaissant b fois dans la multiplication. Par exemple, $3^4 = 3 * 3 * 3 * 3 = 81$. Ecrire un programme qui demande d'entrer deux entiers a et b et qui affiche a^b.

2) On définit $n!$ (dire *factorielle n*) comme $n! = n \cdot (n-1) \ldots 3 \cdot 2 \cdot 1$. Par exemple, $5! = 5 * 4 * 3 * 2 * 1 = 120$. Ecrire un programme qui demande d'entrer n et qui affiche $n!$.

Attention

- Toujours vérifier que la boucle se termine :
  - Incorrect : `for (i=0;i<6;i=i-1)`
- Si nécessaire, utiliser `printf` dans la boucle pour vérifier la valeur du compteur.
- Bien compter le nombre d'exécutions de la boucle :
  - `for(i=0;i<10;i++)` exécute 10 fois la boucle (i de 0 à 9)
  - `for(i=0;i<=10;i++)` exécute 11 fois la boucle (i de 0 à 10)
  - `for(i=1;i<10;i++)` exécute 9 fois la boucle (i de 1 à 9)

Utilisation de `for`

- On utilise généralement une boucle `for` lorsqu'on souhaite exécuter une opération un nombre de fois connu à l'avance.
- Exemple: l'instruction va s'exécuter 10 fois.

```
int n=10;
int i;
for(i=0;i<n;i++)
{
  <instruction(i)>
}
```

3) Suite de Fibonacci. On définit la suite $u_0 = 1$, $u_1 = 1$, $u_n = u_{n-1} + u_{n-2}$ pour $n \geq 2$. Ecrire un programme qui calcule et affiche les n premiers termes de la suite de Fibonacci, pour un n donné. Modifier le programme pour faire afficher les n premiers termes de la suite réelle $v_n = u_{n+1}/u_n$. Que constatez-vous ?

4) Ecrire un programme affichant la décomposition binaire d'un nombre, en commençant par le bit de poids faible.

5) On dit qu'un entier est un nombre *premier* si ses seuls diviseurs sont 1 et lui-même. Par exemple, 1, 2, 3, 5, 7, 11 sont des nombres premiers, mais $15 = 3 * 5$ n'est pas un nombre premier.

Ecrire un programme qui demande un nombre à l'utilisateur et détermine si ce nombre est premier ou pas. Modifiez votre programme pour afficher la liste des nombres premiers de 1 à 100, et déterminez combien il y en a.