Structures itératives

Fondements du langage C

Le langage C présente plusieurs types de structures itératives, c'est-à-dire, des structures de contrôle permettant de réaliser des traîtements cycliques en boucle. Il s'agit alors d'exécuter plusieurs fois une portion de code, généralement jusqu'à ce qu'une condition soit fausse. Il convient donc d'être rigoureux dans l'écriture de la condition afin de s'assurer qu'elle ne restera pas toujours vraie et que l'exécution du programme n'entrera pas dans une boucle infinie. Tout comme les structures conditionnelles, les structures itératives utilisent les opérateurs relationnels et logiques afin d'exprimer la condition à évaluer.

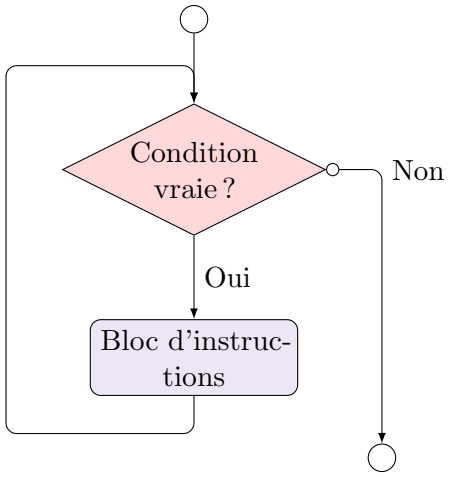

Fig. 1 – *Déroulement d'une structure* `while`

Boucle `while`

La boucle `while` existe dans la plupart des langages de programmation. Elle consiste à tester une condition et à exécuter de manière itérative un bloc d'instructions tant que cette condition reste vraie. Sa syntaxe générale est la suivante :

```
while (/* Condition à évaluer */)
{
  /* Suite d'intructions à traiter */
  /* en boucle tant que la         */
  /* la condition est vraie.       */
}
```

Le bloc d'instructions à exécuter est borné par des accolades. Néanmoins, s'il s'agit de n'exécuter qu'une seule instruction, les accolades sont facultatives.

La figure 1 montre la boucle `while` sous la forme d'un algorigramme. L'évaluation de la condition est effectuée préalablement à l'exécution du bloc d'instructions.

Le code ci-dessous montre un exemple d'utilisation d'une boucle `while` :

```
int compteur = 0;

while (compteur < 10)
{
  printf("Compteur : %d\n", compteur);
  compteur++;
}
```

Il est important de noter que si l'incrémentation de la variable `compteur` n'est pas effectuée, la condition de la boucle `while` qui en dépend ne pourra jamais être fausse et le programme entrera dans une boucle infinie.

Boucle `do...while`

La boucle `do..while` se distingue de la boucle `while` par le fait que la condition est évaluée au terme du traitement du bloc d'instructions. Ainsi,

et même si la condition est initialement fausse, le bloc d'instructions est au minimum exécuté une fois. La syntaxe générale de la boucle `do...while` est la suivante :

```
do
{
  /* Suite d'intructions à traiter */
  /* en boucle tant que la         */
  /* la condition est vraie.       */
} while (/* Condition à évaluer */);
```

La figure 2 montre la boucle `do...while` sous la forme d'un algorigramme. L'évaluation de la condition est effectuée après l'exécution du bloc d'instructions.

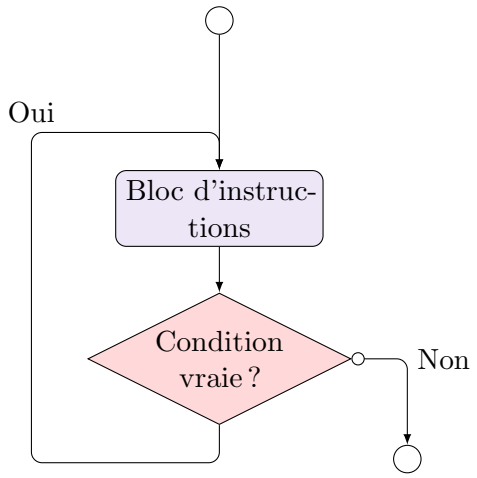

Fig. 2 – *Déroulement d'une structure* `do...while`

Il existe également un risque de boucle infinie si la condition reste systématiquement vraie.

Le code ci-dessous montre un exemple d'utilisation d'une boucle `while` :

```
int compteur = 0;

do
{
  printf("Compteur : %d\n", compteur);
  compteur++;
} while (compteur < 10);
```

Boucle `for`

La boucle `for` est une structure très couramment utilisée dans les programmes écrits en langage C. Elle est généralement mise en œuvre lorsqu'il est nécessaire de répéter un bloc d'instruction durant un nombre de fois préalablement déterminé. Sa syntaxe générale prend en compte trois champs distincts séparés par le caractère `';'`. Elle est la suivante :

```
for (/* Initialisation*/;
     /* Condition */;
     /* Itération de contrôle */)
{
  /* Suite d'intructions à traiter */
  /* en boucle tant que la         */
  /* la condition est vraie.       */
}
```

L'initialisation consiste à fixer la valeur initiale d'une variable utilisée dans le contrôle de la boucle `for`. Le second champ concerne l'expression de la condition à prendre en compte. Le troisième champ s'applique à la mise à jour de la variable de contrôle de boucle. Cette mise à jour se décline le plus souvent sous la forme d'une incrémentation ou d'une décrémentation.

Le code ci-dessous présente un exemple classique d'implémentation d'une boucle `for` :

```
int i;

for (i = 0; i <= 10; i++)
  printf("Valeur : %d\n", i);
```

D'un point de vue algorithmique, la boucle `for` se comporte de façon similaire à la boucle `while`.

Imbrication de boucles

Dans le cadre de traîtements particuliers, il est parfois nécessaire d'imbriquer plusieurs boucles `for`. Le code suivant montre un cas d'application :

```
int i, j;

for (i = 0; i < 10; i++)
  for (j = 0; j < 10; j++)
    printf("Addition : %d\n", i + j);
```