Introduction au langage C

7 janvier 2010

Table des matières

Chapitre 1

Les instructions de contôle

On appelle `instructions de contrôle` ou `structures conditionnelles` les instructions qui permettent de tester la véracité ou non d'une condition. Ces structures conditionnelles peuvent être associées à des structures qui se répètent suivant la réalisation de la condition, on appelle alors ces structures des `structures de boucle`.

1.1 L'instruction if

1.1.1 Syntaxe

```
if (expression)
   instruction1
[else
   instruction2]
```

– La valeur de `expression` est évaluée et, si elle est non nulle, `instruction1` est exécutée sinon c'est `instruction2` qui est exécutée (si elle existe) ;
– `instruction1` et `instruction2` peuvent être des instructions simples ou des blocs d'instruction ;
– La clause `else` peut être omise et se rapporte toujours au dernier `if` visible.

1.1.2 Diagramme syntaxique

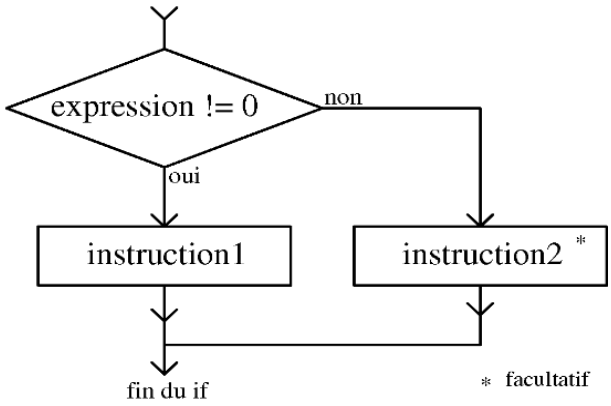

FIG. 1.1 – Diagramme Syntaxique du If

Exemples :

1. ```
   if (i < 10)
   i++;
   ```
2. ```
   if (delta > 0)
   printf("2 racines reelles\n";
   else
   if (delta == 0)
   printf("racine double\n");
   else
   printf("pas de racines reelles\n");
   ```
3. ```
   if (a)        /* equivalent :  if ( a != 0 )  */
   i++;
   ```
4. ```
   if (an % 4 == 0 && an % 100 != 0 || an % 400 == 0)
   printf("annee bissextile\n");
   ```

Exercices

1. Ecrivez un programme qui lit trois valeurs entières (A, B et C) au clavier et qui affiche la plus grande des trois valeurs, en utilisant :
   a) `if ... else` et une variable d'aide MAX ;
   b) `if ... else if ... else ...` sans variable d'aide ;
   c) les opérateurs conditionnels et une variable d'aide MAX ;
   d) les opérateurs conditionnels sans variable d'aide.
2. Ecrivez un programme qui lit trois valeurs entières `A`, `B` et `C` au clavier. Triez les valeurs `A`, `B` et `C` par échanges successifs de manière à obtenir : `val(A) val(B) val(C)`. Affichez les trois valeurs ;

3. Ecrivez un programme qui lit deux valeurs entières `A` et `B` au clavier et qui affiche le signe du produit de `A` et `B` sans faire la multiplication ;
4. Ecrivez un programme qui lit deux valeurs entières `A` et `B` au clavier et qui affiche le signe de la somme de `A` et `B` sans faire l'addition. Utilisez la fonction `fabs` de la bibliothèque `<math>` ;
5. Ecrivez un programme qui calcule les solutions réelles d'une équation du second degré $ax^2 + bx + c = 0$ en discutant la formule. Utilisez une variable d'aide `D` pour la valeur du discriminant $b^2 - 4ac$ et décidez à l'aide de `D`, si l'équation a une, deux ou aucune solution réelle. Utilisez des variables du type `int` pour `A`, `B` et `C` ; Considérez aussi les cas où l'utilisateur entre des valeurs nulles pour `A` ; pour `A` et `B` ; pour `A`, `B` et `C`. Affichez les résultats et les messages nécessaires sur l'écran.

1.2 L'instruction while

Cette instruction permet de répéter une instruction (ou un bloc) `tant qu'une condition est vraie`.

1.2.1 Syntaxe

```
while(expression)
instruction ou bloc d'instructions
```

– `instruction` est exécutée de façon répétitive aussi longtemps que le résultat de expression est non nul ;
– `expression` est évaluée avant chaque exécution de `instruction`.

1.2.2 Diagramme syntaxique

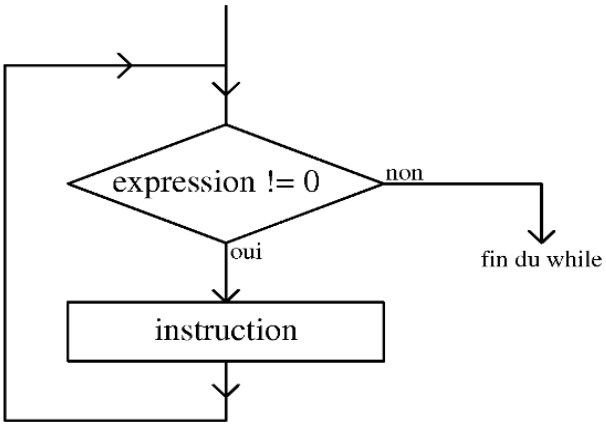

FIG. 1.2 – Diagramme Syntaxique du while

Exemple :

```
int i = 0 ;
/* affiche les nombres de 0 - 9 */
while (i != 10)
{
   printf("%d ", i);
   i++;
}
```

1.3 L'instruction do ... while

1.3.1 Syntaxe

```
do
      instruction ou bloc d'instructions;
while (expression != 0);
```

Cette instruction est similaire à la précédente, le test a lieu après chaque exécution de `instruction` ; de fait `instruction` est au moins exécutée une fois.

1.3.2 Diagramme syntaxique

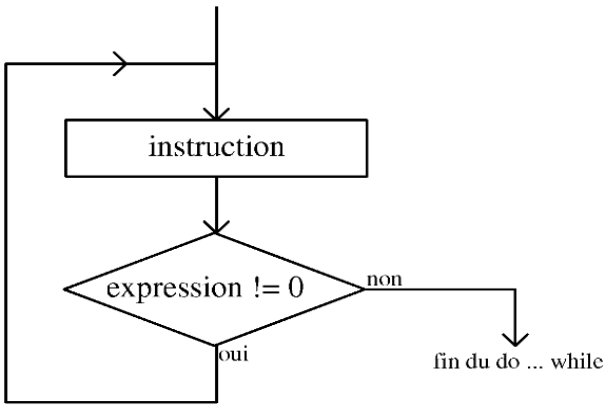

FIG. 1.3 – Diagramme Syntaxique du do while

1.4 L'instruction for

1.4.1 Syntaxe

```
for ([expression1]; [expression2]; [expression3])
      instruction (ou bloc d'instructions);
```

- `instruction` est répétée tant que la valeur de `expression2` est non nulle ;
- Avant la première itération, `expression1` est évaluée ; en général, elle sert à incrémenter les variables de la boucle ;
- Après chaque itération de la boucle, `expression3` est évaluée. En général elle sert à incrémenter le compteur de la boucle.

La boucle `for` est équivalente à la structure suivante :

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

Remarques :

Toutes les expressions sont facultatives. Si `expression2` n'existe pas, elle sera supposée vraie (valeur 1).

1.4.2 Diagramme syntaxique

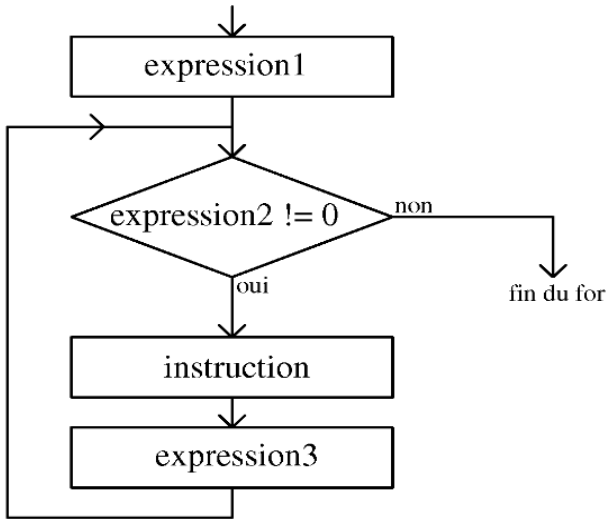

FIG. 1.4 – Diagramme Syntaxique du for

Exemples :

1. ```
   for (i=0; i<10; i++)
           somme += tab[i];
   ```
2. ```
   for( ; ; )
           ;          /* boucle infinie ne faisant rien ! */
   ```
3. ```
   for(i = 0, j = 0; i < 10; i++, j++)
           ...
   ```

1.5 L'instruction switch

L'instruction `switch` permet un choix multiple en fonction de l'évaluation d'une expression.

1.5.1 Syntaxe

```
switch (expression)
{
      case e1 : instruction1 ...
      case e2 : instruction2 ...
      ...
      case e3 : instruction3 ...
      default : instruction_default ...
}
```
```

– L'évaluation de `expression` doit donner pour résultat une valeur de type `int` ;
– `e1`, `e2`, `e3`, ..., sont des expressions constantes qui doivent être un entier unique de type `int` ou `char` ;
– La valeur de `expression` est recherchée successivement parmi les valeurs des différentes expressions constantes `e1`, `e2`, `e3` ;
– En cas d'égalité les instructions (facultatives) correspondantes sont exécutées jusqu'à une instruction `break` ou jusqu'à la fin du bloc du `switch`, et ceci indépendamment des autres conditions `case` ;
– S'il n'y a pas de valeur correspondante, on exécute les instructions du cas `default` (s'il existe).

1.5.2 Diagramme syntaxique

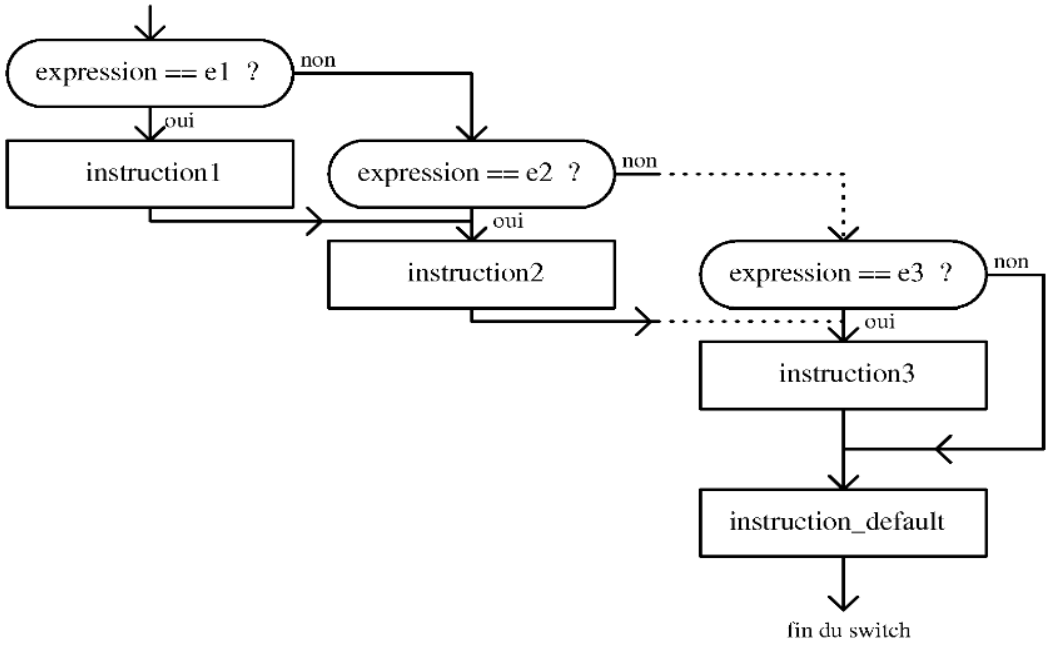

FIG. 1.5 – Diagramme Syntaxique du switch

Exemple

```
switch (car)
{
      case 'a' :
      case 'A' :
      case 'e' :
      case 'E' : v++;
            break;
      case ' ' : espace++;
            break;
      default  : c++;
}
```

1.6 L'instruction break

Cette instruction s'utilise obligatoirement au sein d'une boucle. Une fois lancée cette instruction stoppe irrémédiablement toutes itérations restantes et provoque le passage à l'instruction qui suit immédiatement le corps de la boucle `while`, `do ... while`, `for` ou `switch`. Afin que le programme ne perde pas tout son sens, il est recommandé de placer l'instruction `break` dans une instruction `if` (ou dans une instruction `switch`), sinon on ne fera toujours qu'une itération, et toutes les instructions de la boucle qui suivraient le `break` ne seraient jamais exécutées.

1.7 L'instruction continue

Cette instruction est quasiment équivalente à la précédente, à la nuance près que l'exécution ne se poursuit pas à la première instruction suivant la boucle, mais recommence une nouvelle itération. Elle s'utilise à l'intérieur d'une boucle `while`, `do ... while` ou `for`. Tout code, à l'intérieur d'une boucle, placé après l'instruction `continue` ne sera jamais exécuté. L'instruction `continue` est aussi équivalente à un `goto` à la fin du bloc. De même que pour la clause `break`, il est recommandé de placer cette instruction dans une instruction conditionnelle ou de sélection. Sinon, le programme perd inévitablement de son sens.

Exemple :

```
for (i = -10; i <= 10; i++)
{
if (i == 0)
continue;  // pour éviter la division par zéro
tab[i] = 1 / i;
}
```

Pour la boucle `for`, l'instruction `continue` fait passer à l'évaluation de `expr3`.

Exercices :

1. Soit le programme suivant :

```
#include <stdio.h>
int main(void)
{
      int i,n,som;
      som = 0;
      for(i=0;i<4;i++)
      {
            printf(‘’donnez un entier ’’);
            scanf(‘’ %d ’’,&n);
            som+=n;
      }
      printf(‘’somme : %d\n’’,som);
      return 0;
}
```

   Ecrire un programme réalisant exactement la même chose, en employant :

   a) une instruction `while` ;

   b) une instruction `do ... while`.

2. Calculer la moyenne de notes fournies au clavier. Le nombre de notes n'est pas connu à priori et l'utilisateur peut en fournir autant qu'il le désire. Pour signaler qu'il a terminé on convient de fournir une note négative non comptée dans la moyenne.

3. Afficher un triangle rempli d'étoiles, s'étendant sur un nombre de lignes fourni en donnée et se présentant ainsi :

```
*
**
***
****
```

4. Déterminer si un nombre entier fourni en donnée est premier ou non ;

5. Ecrire un programme qui détermine la nième valeur de la suite de Fibonacci définie comme suit :

```
u1 = 1;
u2 = 1;
un = un-1 + u n-2 pour n > 2
```

6. Ecrire un programme qui affiche la table de multiplication des nombres de 1 à 10.