Structures conditionnelles

Fondements du langage C

Les structures conditionnelles sont très utilisées en programmation pour conditionner l'exécution d'un programme. Du point de vue algorithmique, il s'agit de mécanismes qui permettent d'exprimer une décision et de sélectionner des suites d'instructions à traîter en fonction d'une condition (prédicat) qui peut être vraie ou fausse. Une condition est ainsi une expression dont la valeur va déterminer les actions réalisées dans la suite de l'exécution d'un programme.

Opérateurs relationnels et logiques

Dès lors que le principe d'une structure conditionnelle repose sur l'évaluation de conditions, il est nécessaire de faire appel à des opérateurs permettant d'établir des expressions relationnelles et logiques. Par défaut, une telle expression est associée au type `int` et prend pour valeur 0 si elle est fausse et 1 si elle est vraie. Par ailleurs, toute expression non nulle est logiquement évaluée comme étant vraie.

Les opérateurs relationnels et logiques sont les listés dans le tableau 1 :

Opérateur	Signification
>	Strictement supérieur
<	Strictement inférieur
>=	Supérieur ou égal
<=	Inférieur ou égal
==	Égal
!=	Non égal
&&	ET logique
||	OU logique
!	NON logique

Tab. 1 – *Liste des opérateurs relationnels et logiques*

Il est alors possible de combiner plusieurs expressions relationnelles au moyen d'opérateurs logiques tel que le montre l'exemple ci-dessous :

```
(age >= 8) && (age <= 12)
```

Dans cet exemple, la condition à évaluer n'est vraie que si les deux expressions relationnelles sont vraies.

L'utilisation de l'opérateur NON logique permet d'évaluer l'inverse de l'expression :

```
!(age >= 18)
```

L'exemple ci-dessus exprime une condition qui est vraie si la valeur de la variable `age` n'est pas supérieure ou égale à 18, c'est-à-dire qu'elle est inférieure à 18.

Structure `if...else`

La structure conditionnelle la plus courante et la plus basique est `if...else`. Son principe repose sur l'évaluation booléenne d'une condition. Sa syntaxe générale est la suivante :

```
if (/* Condition à évaluer */)
{
  /* Suite d'intructions à       */
  /* traiter si la condition est */
  /* vraie.                      */
}
else
{
  /* Autre suite d'intructions à */
  /* traiter si la condition est */
  /* si la condition est fausse. */
}
```

La figure 1 présente sous forme d'un algorigramme le déroulement d'une structure `if...else`. Elle schématise le fait que l'exécution du programme ne peut s'appliquer qu'à l'un des deux blocs d'instructions sélectionné en fonction du résultat de l'évaluation de la condition.

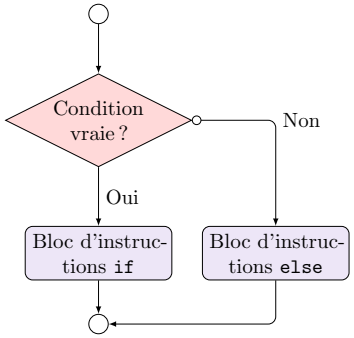

Fig. 1 – *Déroulement d'une structure `if...else`*

Lorsqu'il est nécessaire d'exécuter plusieurs instructions selon que la condition est vérifiée ou non, il faut créer un bloc de code, c'est-à-dire un ensemble d'instructions borné par des accolades ({ et }). Cependant et même lorsqu'une seule instruction doit être traitée, il est conseillé d'utiliser systématiquement les accolades. Le code ci-dessous présente un exemple d'utilisation de la structure `if...else` :

```
if (age >= 18)
{
  printf("Vous êtes majeur !");
}
else
{
  printf("Vous êtes mineur !");
}
```

Le cas `else` n'est pas directement associé à condition. Il correspond en effet au traitement par défaut, c'est-à-dire qu'il est pris en compte uniquement lorsque la condition booléenne exprimée au niveau du `if` n'est pas vérifiée.

L'exemple ci-dessous montre l'utilisation du `if` considérant deux expressions relationnelles liées par l'opérateur ET logique :

```
if ((speed >= 150) && (speed < 200))
{
  printf("Rapide !");
}
```

Un autre exemple présenté ci-dessous permet d'exclure les cas limites :

```
if ((speed == 0) || (speed >= 300))
{
  printf("Hors limites !");
}
```

Dans cet exemple, c'est l'opérateur OU logique qui est utilisé.

Il est également possible d'imbriquer les cas d'évaluation d'une condition en utilisant le test `else if`. Il s'agit alors d'une structure `if...else` généralisée permettant de prendre en compte un plus grand nombre de cas. Le code suivant montre un exemple d'implémentation :

```
if (vitesse >= 200)
{
  printf("Très rapide !");
}
else if (vitesse >= 150)
{
  printf("Rapide !");
}
else if (vitesse >= 100)
{
  printf("Moyen !");
}
else
{
  printf("Lent !");
}
```

L'algorigramme présenté par la figure 2 montre le déroulement d'une structure de type `if...else` `if...else` basée sur un ensemble de trois évaluations de condition. Ces différentes évaluations sont successivement considérées jusqu'à ce que l'une d'entre-elles soit vérifiée. Le cas échéant, le bloc d'instructions correspondant est traité. Dans le cas contraire, c'est le bloc d'instructions par défaut et associé au `else` qui est exécuté.

Structure switch...case

La structure `switch...case` s'applique lorsqu'un choix multiple doit être effectué. En termes d'usage et de représentation algorithmique, elle s'apparente à une structure `if...else` généralisée. L'instruction `switch` permet de faire plusieurs tests de valeurs sur le contenu d'une même variable. Ce branchement conditionnel simplifie beaucoup le test de plusieurs valeurs d'une variable en évitant l'utilisation d'une

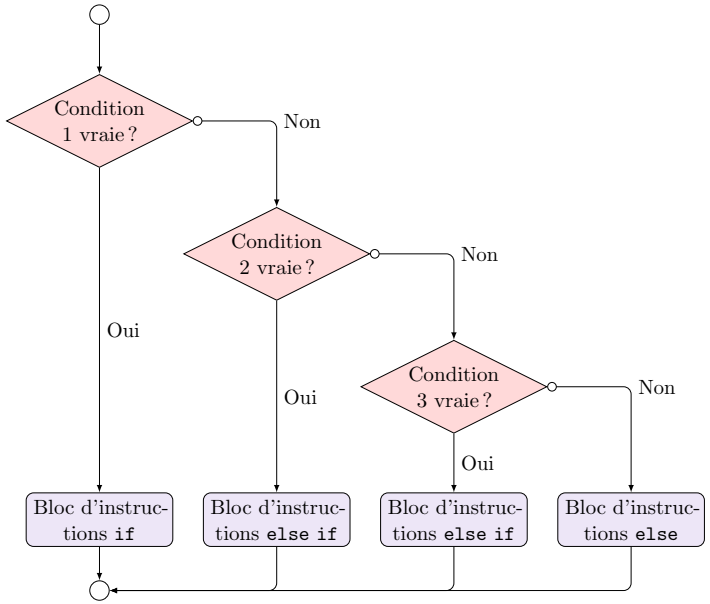

Fig. 2 – *Déroulement d'une structure* `if...else if...else`

structure de type `if...else if...else`. Sa syntaxe générale est la suivante :

```
switch (/* Variable à évaluer */)
{
  case /* Valeur 1 */ :
  /* Suite d'instructions */
  /* à traiter           */
  break;

  case /* Valeur 2 */ :
  /* Suite d'instructions */
  /* à traiter           */
  break;

  case /* Valeur 3 */ :
  /* Suite d'instructions */
  /* à traiter           */
  break;

  default :
  /* Suite d'instructions */
  /* à traiter           */
}
```

Chaque cas correspond à une confrontation de la valeur de variable à évaluer avec une valeur constante. Lorsque l'égalité est vérifiée, le bloc d'instructions associé au cas est exécuté.

L'instruction `break` provoque la sortie du bloc `switch`. Si elle n'est pas précisée, le cas suivant est également évalué. Cette situation n'a généralement aucun intérêt et il convient le plus souvent d'utiliser l'instruction `break` afin d'éviter d'exécuter des tests inutiles.

Le cas `default` est facultatif comme peut l'être celui du `else` dans une structure `if...else`. Il est exécuté seulement si la valeur de la variable à évaluer ne correspond à aucun des cas précémment évalués.

```
switch (commande)
{
  case 'M' :
  printf("Démarrage !\n");
  break;

  case 'A' :
  printf("Arrêt !\n");
  break;

  default :
  printf("Ordre non valide !\n");
}
```