



MINDS' ACADEMY

Niveau	1A&3B
Chapiter 1	Les structures conditionnelles
Objectif	<ul style="list-style-type: none">• Connaître et appliquer les structures conditionnelles (if ... else) et if imbriqué.• Connaître et appliquer la structure conditionnelle switch.

Table des matières

1 Introduction

1.1 Instructions de test

2 L'instruction if

2.1 Test simple

2.1.1 Quelques erreurs à éviter

2.1.2 Instruction if imbriqués

2.2 Test avec alternative

2 L'instruction switch

Introduction

Dans le précédent chapitre, nous avons étudié les trois instructions de base que sont l'affectation, la lecture et l'écriture. Jusqu'ici, nous ne les avons, utilisées, que pour réaliser des programmes dans lesquels les instructions étaient exécutées séquentiellement, c'est-à-dire tout simplement dans l'ordre dans lequel elles apparaissaient dans le programme .

L'intérêt et la puissance de l'ordinateur sont essentiellement dus à la possibilité d'effectuer des choix et des répétitions au sein du programme. La plupart des langages évolués, et en particulier le C, disposent à cet effet de structures de contrôle : on parle suivant les cas de structures de choix ou de structures de répétition. Ici, nous allons étudier l'instruction permettant de réaliser des structures de choix, à savoir ***if et switch***. Les instructions permettant de réaliser des structures de répétition seront étudiées dans le chapitre suivant.

1.1 Instructions de test

Le langage C permet trois types de tests différents : simple, avec alternative, et multiple. Pour chacun, on distingue deux parties :

- Une ou plusieurs conditions .
- Un ou plusieurs blocs.

Condition : *expression dont la valeur est interprétée de façon booléenne : la condition peut être soit vraie, soit fausse.*

Le principe est d'associer une condition à un bloc. Autrement, si une condition donnée est vraie, alors une certaine partie du programme sera exécutée. Sinon, c'est une autre partie qui sera exécutée, voire aucune partie.

Remarque : *une fois l'instruction de test exécutée, le programme reprend son cours normalement*

L'instruction if

2.1 Test Simple

Avec le test simple, on associe une seule condition à un seul bloc de code. Si la condition est vraie, ce bloc est exécuté. Sinon, il n'est pas exécuté.

Cette instruction est notée if, et sa syntaxe est la suivante :

```
if(condition)
{
    instruction 1;
    ...
    instruction n;
}
```

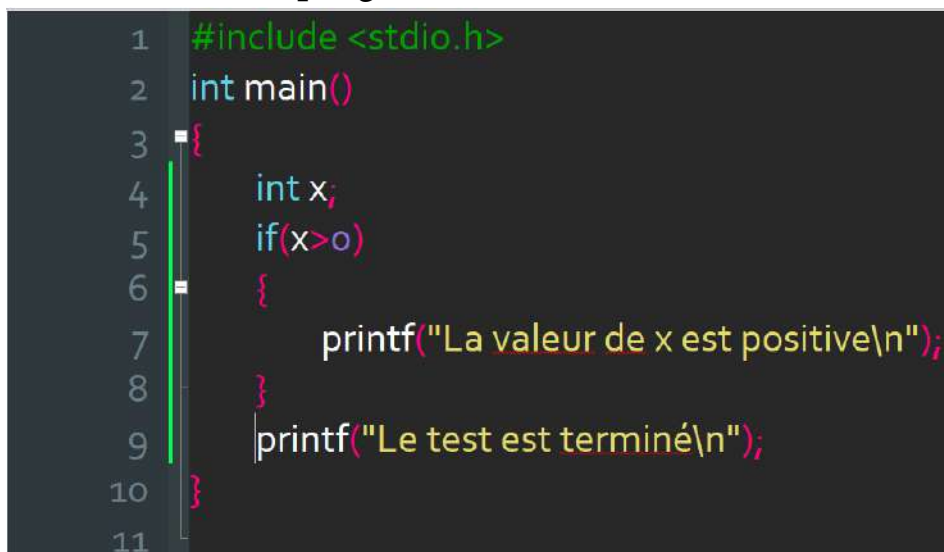
Notez l'absence de point-virgule ";" après la condition.

Le fonctionnement est le suivant. Tout d'abord, l'expression condition est évaluée :

- Si elle est vraie (si sa valeur est un entier non-nul) alors le bloc est exécuté
- Si elle est fausse (si sa valeur est zéro) alors le bloc n'est pas exécuté.

Comme indiqué précédemment, dans les deux cas, l'exécution du programme continue normalement ensuite.

exemple : considérons le programme suivant :



```
1  #include <stdio.h>
2  int main()
3  {
4      int x;
5      if(x>0)
6      {
7          printf("La valeur de x est positive\n");
8      }
9      printf("Le test est terminé\n");
10 }
11
```

Pour x=4 on va obtenir l'affichage suivant :

La valeur de x est positive

Le test est terminé

Pour $x=-8$, on aura l'affichage suivant:

Le test est terminé

Remarque : si le bloc ne contient qu'une seule instruction, on peut omettre les accolades.

exemple : le if de l'exemple précédent peut se simplifier de la façon suivante :

```
5   if(x>0)
6       printf("La valeur de x est positive\n");
7
```

2.1.1 Quelques erreurs à éviter

Il faut faire très attention avec la syntaxe du dernier exemple, car si vous l'utilisez, toute instruction supplémentaire sera considérée comme extérieure au if.

exemple : supposons qu'on a le code source suivant :

```
5   if(x>0)
6       printf("La valeur de x n'est pas négative\n");
7       printf("Et je dirais même qu'elle est positive\n");
8       printf("Le test est terminé\n");
9   }
```

Alors, une valeur $x=4$ produira bien le résultat attendu :

La valeur de x n'est pas négative

Et je dirais même qu'elle est positive

Le test est terminé

Mais une valeur négative comme $x=-8$ affichera également la ligne supplémentaire (en italique dans le code source) :

Et je dirais même qu'elle est positive

Le test est terminé

En effet, il s'agit de la deuxième instruction après le if, et aucune accolade ne vient délimiter le bloc, donc cette ligne n'appartient pas au if.

Une autre erreur classique des programmeurs débutant consiste à placer un point-virgule juste après la condition, de la façon suivante :

```
5   if(x>0);
6       printf("La valeur de x est positive\n");
7
```

Il ne s'agit pas d'une erreur de compilation, la syntaxe est correcte. En effet, le compilateur considère que le bloc du if est constitué d'une seule instruction, qui est... rien du tout. Donc, si $x > 0$, ici on ne fera rien de spécial. Cela signifie aussi que le printf ne se trouve pas dans le if, mais à l'extérieur. Et par conséquent, il sera exécuté quelle que soit la valeur de x.

2.1.2 Instruction if imbriquées

Il est possible de combiner plusieurs if, de manière à effectuer des séquences de tests. On dit alors qu'on imbrique les if :

Imbrication : fait d'utiliser une instruction dans le bloc d'une autre instruction.

exemple :

```
5   if(x>0)
6   {
7       printf("La valeur de x est positive\n");
8       if(x>10)
9           printf("Je dirais même que la valeur de x est supérieure à 10\n");
10  }
11
```

Pour $x=4$, on a l'affichage suivant :

La valeur de x est positive

Alors que pour $x=14$, on a l'affichage suivant :

La valeur de x est positive

Je dirais même que la valeur de x est supérieure à 10

2.2 Test avec alternative

Avec le test simple, on exécute un bloc seulement si une condition est vraie, et on ne l'exécute pas si la condition n'est pas vraie. Il est possible de proposer un bloc alternatif, à exécuter quand la condition n'est pas vraie (plutôt que de ne rien faire). On parle alors de test avec alternative.

Ce bloc additionnel est introduit par le mot clé else, en respectant la syntaxe suivante :

```
if(condition)
{
    instruction 1a;
    ...
    Instruction na;
}
else
{
    instruction 1b;
    ...
    instruction nb;
}
```

Exemple:

```
5      if(x>0)
6          printf("La valeur de x est positive\n");
7      else
8      {
9          printf("La valeur de x n'est pas positive\n");
10     }
11     printf("Le test est terminé\n");
```

Pour $x=4$, on aura exactement le même affichage que pour le test simple.
Par contre, pour $x=-8$, on aura l'affichage suivant :

La valeur de x n'est pas positive

Le test est terminé

Comme pour le if, il est possible d'omettre les accolade si le bloc du else ne contient qu'une seule instruction. On peut donc simplifier l'exemple précédent de la façon suivante :

```
5      if(x>0)
6          printf("La valeur de x est positive\n");
7      else
8          printf("La valeur de x n'est pas positive\n");
9      printf("Le test est terminé\n");
```

Il est aussi possible d'imbriquer d'autres if dans le bloc du else

L'instruction switch

Dans certains cas, on a besoin de comparer une variable donnée à plusieurs valeurs différentes, et non pas une seule comme précédemment.

exemple : soit une variable entière x. On veut effectuer un traitement différent suivant que la valeur de la variable est 1, 2, 3 ou plus. Si on utilise if, on va devoir procéder par imbrications successives :

```
4      if(x==1)
5      {...
6      }
7      else
8      {
9          if(x==2)
10         {...
11         }
12         else
13         {
14             if(x==3)
15             {...
16             }
17             else
18             {...
19             }
20         }
21     }
22 }
```

L'instruction switch permet d'effectuer le même traitement, mais avec une syntaxe plus compacte. Le mot-clé switch reçoit entre parenthèses la variable à tester. Chaque valeur à comparer est ensuite indiquée dans le bloc du switch, en utilisant le mot-clé case, de la façon suivante :


```

4      switch(variable)
5      {
6          case valeur_a:
7              instruction 1a;
8              ...
9              instruction na;
10         case valeur_b:
11             instruction 1b;
12             ...
13             instruction nb;
14             ...
15     }

```

Le fonctionnement est le suivant. La variable spécifiée va être comparée successivement à chaque valeur proposée, jusqu'à ce qu'il y ait égalité. Si la variable n'est égale à aucune valeur, alors on sort simplement du switch sans rien faire de plus. Si on trouve une valeur égale, alors on exécute toutes les instructions situées en-dessous du case correspondant. Attention : cela inclut non seulement les instructions du case associé à la valeur, mais aussi celles de tous les autres case suivants !

Or, il arrive fréquemment qu'on désire que seules les instructions placées directement en dessous du case correspondant soient exécutées, mais pas les suivantes. Pour arriver à ce comportement, il faut ajouter l'instruction `break` juste avant le case suivant, comme indiqué ci-dessous :

```

6      case valeur_a:
7          instruction 1a;
8          ...
9          instruction na;
10         break;
11     case valeur_b:
12         instruction 1b;
13         ...
14         instruction nb;
15         break;
16     ...

```

exemple : on veut que le traitement soit le même si x a la valeur 2 ou la valeur 3, mais on veut aussi un traitement spécifique pour la valeur 1 :

```
4      switch(x)
5      {
6          case 1:
7              instruction1-1;
8              instruction1-2;
9              break;
10         case 2:
11         case 3:
12             instruction23-1;
13             instruction23-2;
14             break;
15     }
```

Il est possible de définir un cas par défaut, c'est-à-dire de préciser un comportement si la variable ne correspond à aucune des valeurs spécifiées dans les case. On utilise pour cela le mot-clé `default` à la place de `case`, de la façon suivante :

```
...
case ...:
    ...
    break;
default:
    ...
}
```

```
4      switch(x)
5      {
6          case 1:
7              ...
8              break;
9          case 2:
10             ...
11             break;
12          case 2:
13             ...
14             break;
15          default:
16              ...
17     }
```

À noter que ce cas par défaut se place tout à la fin du switch.

exemple : on reprend le programme qui utilisait des `if`, au début de cette sous-section, et on les remplace par un switch équivalent

Remarque : la structure de contrôle switch s'utilise uniquement sur des valeurs entières (donc aussi sur des caractères).

Notez que l'instruction break est utilisable dans d'autres situations, comme par exemple dans des boucles. Mais cela rend les programmes peu lisibles et difficiles à comprendre. Par conséquent, dans le cadre de ce cours, vous n'avez le droit d'utiliser break que dans les switch.