

Chapitre III

LES INSTRUCTIONS DE BRANCHEMENT CONDITIONNEL

Objectifs

Général

⇒ Se familiariser avec les instructions de branchement conditionnel et comprendre leur utilité.

Spécifiques

- ❑ Comprendre les structures "if - else", "if sans else" et "if - else - if";
 - ❑ Comprendre la structure "switch - case";
 - ❑ Connaître les opérateurs conditionnels.
-

1. Introduction

Les structures de contrôle définissent la suite dans laquelle les instructions sont effectuées. Dans ce chapitre, nous allons voir comment les instructions de sélection connues fonctionnent en C et nous allons faire connaissance d'un couple d'opérateurs spécial qui nous permet de choisir entre deux valeurs à l'intérieur d'une expression.

Constatons déjà que la particularité la plus importante des instructions de contrôle en C est le fait que les 'conditions' en C peuvent être des *expressions quelconques qui fournissent un résultat numérique*. La valeur zéro correspond à la valeur logique faux et toute valeur différente de zéro est considérée comme vrai.

2. La structure if - else

→ La structure alternative en langage algorithmique

```
si (<expression logique>)  
  alors  
    <bloc d'instructions 1>  
  sinon  
    <bloc d'instructions 2>  
  Finsi
```

- ✓ Si l'<expression logique> a la valeur logique *vrai*, alors le <bloc d'instructions 1> est exécuté,
- ✓ Si l'<expression logique> a la valeur logique *faux*, alors le <bloc d'instructions 2> est exécuté.

→ La structure alternative en C

```
if ( <expression> )  
    <bloc d'instructions 1>  
else  
    <bloc d'instructions 2>
```

- ✓ Si l'<expression> fournit une valeur différente de zéro, alors le <bloc d'instructions 1> sera exécuté,
-

- ✓ Si l'<expression> fournit la valeur zéro, alors le <bloc d'instructions 2> est exécuté.

La partie <expression> peut désigner :

- ✓ Une variable d'un type numérique,
- ✓ Une expression fournissant un résultat numérique.

La partie <bloc d'instructions> peut désigner :

- ✓ un (vrai) bloc d'instructions compris entre accolades,
- ✓ une seule instruction terminée par un point-virgule.

Exemple 1

```
if (a > b)
    max = a;
else
    max = b;
```

Exemple 2

```
if (EGAL)
    printf("A est égal à B\n");
else
    printf("A est différent de B\n");
```

Exemple 3

```
if (A-B) printf("A est différent de B\n");
else printf("A est égal à B\n");
```

Exemple 4

```
if (A > B)
{
    AIDE = A; A = C; C = AIDE;
}
else
{
    AIDE = B; B = C; C = AIDE;
}
```

3. La structure if sans else

La partie **else** est facultative. On peut donc utiliser **if** de la façon suivante:

```
if ( <expression> )
    <bloc d'instructions>
```

Attention !

Comme la partie **else** est optionnelle, les expressions contenant plusieurs structures **if** et **if - else** peuvent mener à des confusions.

Exemple 1

L'expression suivante peut être interprétée de deux façons:

```
if (N>0)
    if (A>B)
        MAX=A;
    else
        MAX=B;
```

```
if (N>0)
    if (A>B)
        MAX=A;
    else
        MAX=B;
```

Pour $N=0$, $A=1$ et $B=2$,

→ Dans la première interprétation, MAX reste inchangé,

→ Dans la deuxième interprétation, MAX obtiendrait la valeur de B.

Sans règle supplémentaire, le résultat de cette expression serait donc imprévisible.

Convention

En C une partie **else** est toujours liée au dernier **if** qui ne possède pas de partie **else**. Dans notre exemple, C utiliserait donc la première interprétation.

Solution

Pour éviter des confusions et pour forcer une certaine interprétation d'une expression, il est recommandé d'utiliser des accolades `{ }`.

Exemple

Pour forcer la deuxième interprétation de l'expression ci-dessus, nous pouvons écrire:

```
if (N>0)
{
    if (A>B) MAX=A;
}
else
    MAX=B;
```

Exercice 3.1

Considérez la séquence d'instructions suivante:

```
if (A>B) printf ("premier choix \n"); else
    if (A>10) printf ("deuxième choix \n");
    if (B<10) printf ("troisième choix \n");
    else printf ("quatrième choix \n");
```

a) Copier la séquence d'instructions en utilisant des tabulateurs pour marquer les blocs **if - else** appartenant ensemble.

b) Déterminer les réponses du programme pour chacun des couples de nombres suivants et vérifiez à l'aide de l'ordinateur.

- ✓ A=10 et B=5
- ✓ A=5 et B=5
- ✓ A=5 et B=10
- ✓ A=10 et B=10
- ✓ A=20 et B=10
- ✓ A=20 et B=20

Solution

a)

```
if (A>B)
    printf ("premier choix \n");
else
    if (A>10)
        printf ("deuxième choix \n");
```

```

    if (B<10)
        printf ("troisième choix \n");
    else
        printf ("quatrième choix \n");

```

b)

- ✓ A=10 et B=5 : premier choix et troisième choix
- ✓ A=5 et B=5 : troisième choix
- ✓ A=5 et B=10 : quatrième choix
- ✓ A=10 et B=10 : quatrième choix
- ✓ A=20 et B=10 : premier choix et premier choix
- ✓ A=20 et B=20 : deuxième choix et quatrième choix

4. La structure “if - else if - ... – else”

En combinant plusieurs structures **if - else** en une expression nous obtenons une structure qui est très courante pour prendre des décisions entre plusieurs alternatives: ***if - else - ... – else.***

```

if ( <expr1> )
    <bloc1>
else if (<expr2>)
    <bloc2>
else if (<expr3>)
    <bloc3>
else if (<exprN>)
    <blocN>
else <blocN+1>

```

Les expressions <expr1> ... <exprN> sont évaluées du haut vers le bas jusqu'à ce que l'une d'elles soit différente de zéro. Le bloc d'instructions y lié est alors exécuté et le traitement de la commande est terminé.

Exemple 1

```

#include <stdio.h>
main()
{
    int A,B;
    printf("Entrez deux nombres entiers :");
    scanf("%d %d", &A, &B);
    if (A > B)
        printf("%d est plus grand que %d\n", A, B);
    else if (A < B)
        printf("%d est plus petit que %d\n", A, B);
    else
        printf("%d est égal à %d\n", A, B);
    return 0;
}

```

La dernière partie **else** traite le cas où aucune des conditions n'a été remplie. Elle est optionnelle, mais elle peut être utilisée très confortablement pour détecter des erreurs.

Exemple 2

```

...
printf("Continuer (O)ui / (N)on ?");
getchar(C);
if (C=='O')

```

```

    {
        ...
    }
else if (C=='N')
    printf("Au revoir ...\n");
else
    printf("\aErreur d'entrée !\n");
...

```

Exercice 3.2

Considérez la séquence d'instructions suivante:

```

if (A>B)
    if (A>10)
        printf ("premier choix \n"); else if (B<10)
        printf ("deuxième choix \n"); else
        if (A==B) printf ("troisième choix \n");
        else printf ("quatrième choix \n");

```

- Copier la séquence d'instructions en utilisant des tabulateurs pour marquer les blocs if - else appartenant ensemble.
- Pour quelles valeurs de A et B obtient-on les résultats: premier choix, deuxième choix, ... sur l'écran?
- Pour quelles valeurs de A et B n'obtient-on pas de réponse sur l'écran?

Solution

a)

```

if (A>B)
    if (A>10)
        printf ("premier choix \n");
    else if (B<10)
        printf ("deuxième choix \n");
    else if (A==B)
        printf ("troisième choix \n");
    else
        printf ("quatrième choix \n");

```

b) Le résultat:

"premier choix" : apparaît pour (A>B) et (A>10)

"deuxième choix" : apparaît pour (10A>B)

"troisième choix" : apparaît pour (10A>B10) et (A=B)

10>10 impossible

A>B et A=B impossible → "troisième choix" n'apparaît jamais

"quatrième choix" : apparaît pour (10A>B10) et (AB)

10>10 impossible → "quatrième choix" n'apparaît jamais

- On n'obtient pas de réponses pour (A>B). Si (A>B) alors la construction if - else if - ... - else garantit que toutes les combinaisons sont traitées et fournissent un résultat.

5. La structure switch-case

C'est un moyen qui permet de décider dans une structure à plusieurs cas. Cette instruction teste si une expression prend une valeur parmi un ensemble de constante et fait le branchement en conséquence.

```

Switch (expression)
{
    case expr_cst1 : liste d'instructions 1
    case expr_cst2 : liste d'instructions 2
    ...
    default : liste d'instructions N
}

```

→ Le choix de l'ensemble d'instructions à exécuter est calculé par l'évaluation de **expression** qui doit envoyer un type entier.

→ *expr_cste* doit être un **int** et doit être unique

Exemple

```

int mois ;
scanf ("%d", &mois) ;
switch (mois)
{
    case 1 : printf (janvier) ; break ;
    case 2 : printf (fevrier) ; break ;
    ...
    case 12 : printf (décembre) ; break ;
    default : printf (erreur) ;
}

```

L'instruction **break** permet de sortir de l'instruction **switch**. Elle est importante car si on ne la met pas après chaque cas d'exécution alors toutes les instructions après ce cas seront exécutées (bien sur s'ils ne sont pas suivis d'une autre instruction **break**).

Remarque

Il existe des instructions de branchement non conditionnel : *goto*, *continue* et *break*.

6. Les opérateurs conditionnels

Le langage C possède une paire d'opérateurs un peu exotiques qui peut être utilisée comme alternative à **if - else** et qui a l'avantage de pouvoir être intégrée dans une expression:

```
<expr1> ? <expr2> : <expr3>
```

→ Si *<expr1>* fournit une valeur différente de zéro, alors la valeur de *<expr2>* est fournie comme résultat,

→ Si *<expr1>* fournit la valeur zéro, alors la valeur de *<expr3>* est fournie comme résultat.

Exemple 1

La suite d'instructions

```

if (A>B)
    MAX=A;
else
    MAX=B;

```

peut être remplacée par:

```
MAX = (A > B) ? A : B;
```

Employés de façon irréflechis, les opérateurs conditionnels peuvent nuire à la lisibilité d'un programme, mais si on les utilise avec précaution, ils fournissent des solutions très élégantes.

Exemple 2

```
printf("Vous avez %i carte%c \n", N, (N==1) ? ' ' : 's');
```

Les ***règles de conversion de types*** s'appliquent aussi aux opérateurs conditionnels `?:`. Ainsi, pour un entier `N` du type **int** et un rationnel `F` du type **float**, l'expression :

```
(N>0) ? N : F
```

va toujours fournir un résultat du type **float**, n'importe si `N` est plus grand ou plus petit que zéro!

7. Exercices d'application

Exercice 3.3

Ecrivez un programme C 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.

Exercice 3.4

Ecrire un programme C qui permet de lire la note d'un candidat et le déclarer « ajourné » s'il a moins de 8, « convoqué à l'oral » s'il a de 8 compris à 10 non compris, ou « admis » s'il a 10 ou plus.

Exercice 3.5

Ecrire un programme C qui permet de lire trois nombres et dire s'ils sont tous différents (il n'y a pas au moins deux qui sont égaux), ou si deux au moins sont égaux, ou s'ils sont tous égaux.

Exercice 3.6

Ecrire un programme C qui permet de lire deux nombres `A` et `B` et échanger éventuellement leurs valeurs de telle sorte qu'à la fin `A < B`.

Exercice 3.7

Ecrire un programme C qui permet de lire trois nombres `A`, `B` et `C`, puis les échanger, autant de fois que cela sera nécessaire, de telle sorte qu'à la fin `A < B < C`. On pourra chercher d'abord à amener la plus petite des trois valeurs dans `A`, puis la plus petite parmi `B` et `C` dans `B` : la plus grande se retrouvera dans `C`.

Exercice 3.8

Soient deux bornes `Min` et `Max` fixées en constantes dans le programme. Ecrire un programme C qui permet de lire un nombre `N` et d'afficher l'un des trois messages : « Inférieur au Min », « Compris entre les bornes » ou « Supérieur au Max », selon la valeur de `N`.

Exercice 3.9

Ecrire un programme C qui, étant donné le temps de départ d'un train (heures `Hd` et minutes `Md`) et la durée du voyage (`Hv` et `Mv`), donne le temps d'arrivée (`Ha` et `Mf`).

Exercice 3.10

Ecrire un programme C qui affiche un menu de choix pour calculer la surface d'un dessin (Rectangle, Carré, Cercle, Triangle et trapèze) et calcule selon le choix de l'utilisateur la surface d'un dessin

Exercice 3.12

Ecrire un programme C qui calcule :

- le discriminant DELTA d'un trinôme de second degré,
- la ou les racines réelles du trinôme.

N.B : utiliser tous les cas possibles.

Exercice 3.13

Ecrire un programme C qui calcule le module et l'argument d'un nombre complexe $Z = A + iB$ que l'on saisira en entrant successivement sa partie réelle et sa partie imaginaire.

$$r = |Z| = \sqrt{A^2 + B^2}$$

$$\theta = \arg(Z) = a \tan\left(\frac{B}{A}\right)$$

N.B : utiliser tous les cas possibles.
