Cours: Algorithmes et Structures des Données

Chapitre 3: Les instructions de contrôle

Réalisé par:

Dr. Sakka Rouis Taoufik

I. Introduction

Les instructions de contrôle servent à contrôler le déroulement de l'enchaînement des instructions à l'intérieur d'un programme.
Ces instructions peuvent être des instructions:

- **conditionnelles** : permettent de réaliser des tests, et suivant le résultat de ces tests, d'exécuter des parties de code différentes.

➔C distingue deux types de structures conditionnelles:

- la structure simple **if**
- la structure à choix multiples **switch.**

- **itératives:** sont commandées par trois types de boucles : le **for**, le **while** et le **do while**

II. Instructions conditionnelles

1) L'instruction de teste : if

L'opérateur de test du langage C se présente généralement sous la forme suivante :

Syntaxe :

```
if ( expression ) {
    ..... ;        /*bloc d'instructions*/
}
else {
    ..... ;        /*bloc d'instructions*/
}
```

Dans le cas où aucun traitement n'est évoqué, si l'expression logique est fausse, la structure conditionnelle devient :

```
if ( expression ) {
    ..... ;        /*bloc d'instructions*
}
```

II. Instructions conditionnelles

1) L'instruction de teste : if

Les { } ne sont pas nécessaire lorsque les blocs ne comporte qu'une seule instruction.

L'expression n'est pas forcément un test qui retourne la valeur 0 ou +1.

L'expression peut être **un calcul ou une affectation**, car, comme nous l'avons déjà dit dans le chapitre 2, il n'y a pas de type booléen.

Une expression est **vraie** si elle ramène un résultat **non nul** ; elle est **fausse** si le résultat **est nul**.

II. Instructions conditionnelles

1) L'instruction de teste : if

Exercise 1:

Réaliser en C un algorithme 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.

Exercise 2:

Réaliser en C un algorithme 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.

Exercise 3:

Réaliser en C un algorithme qui lit trois valeurs entières (A, B et C) au clavier. Trier les valeurs A, B et C par échanges successifs de manière à obtenir (A <= B <= C) puis afficher les trois valeurs.

II. Instructions conditionnelles

1) L'instruction de teste : if

Exercise 4:

On veut déterminer si une année A est bissextile ou non.
Si A n'est pas divisible par 4 l'année n'est pas bissextile.
Si A est divisible par 4, l'année est bissextile sauf si A est divisible par 100 et pas par 400.

Exemples :

1980 et 1996 sont bissextiles car elles sont divisibles par 4
2000 est une année bissextile car elle est divisible par 400
2100 et 3000 ne sont pas bissextiles car elles ne sont pas divisibles par 400.

Réaliser en C un algorithme qui effectue la saisie de la donnée, détermine si l'année est bissextile ou non et affiche le résultat.

II. Instructions conditionnelles

2) L'instruction de sélection multiple : switch

Syntaxe :

```
switch (expression) {
    case constante1 :
            liste d'instructions1;
            break;
    case constante2 :
            liste d'instructions2;
            break;
      ...
    default :
            liste d'instructionsN;
            break;
}
```

II. Instructions conditionnelles

2) L'instruction de sélection multiple : switch

L'instruction **switch** permet d'éviter les imbrications d'instructions **if**

L'instruction **switch** prend la valeur de la variable « expression » et compare à chacune des étiquettes **case**, dès qu'elle trouve celle qui correspond, les instructions qui suivent sont exécutées:

- soit jusqu'à la rencontre d'une instruction **break**,
- soit jusqu'à la fin du corps de l'instruction **switch**.

II. Instructions conditionnelles

2) L'instruction de sélection multiple : switch

Exemple 1 :

On suppose que «choix» est une variable de type caractère, une instruction **switch** typique est la suivante :

```
switch (choix) {
        case 'R' : printf ("Rouge") ; break ;
        case 'B' : printf ("Bleu") ;    break ;
        case 'J' : printf ("Jaune") ;   break ;
}
```

II. Instructions conditionnelles

2) L'instruction de sélection multiple : switch

Exemple 2 :

On suppose que «jour» est une variable de type entier, une instruction **switch** typique est la suivante :

```
switch (jour) {
        case 0 : case 1: case 2: case 3 : case 4:
                printf ("Au travail !") ; break ;
        case 5 : printf ("Aujourd'hui est samedi") ; break ;
        case 6 : printf ("Aujourd'hui est dimanche") ; break ;
        default: printf ("erreur") ;

}
```

II. Instructions conditionnelles

2) L'instruction de sélection multiple : switch

Remarques

Les instructions qui suivent la condition **default** sont exécutées lorsqu'aucune constante des **case** n'est égale à la valeur retournée par l'expression.

Le dernier **break** est facultatif. Il vaut mieux le laisser pour la cohérence de l'écriture, et pour ne pas avoir de surprise lorsqu'un **case** est ajouté.

Plusieurs valeurs de case peuvent aboutir sur les mêmes instructions.

II. Instructions conditionnelles

2) L'instruction de sélection multiple : switch

Exercice 1:

Écrire un programme C qui lit une date sous la forme N° du jour, N° du mois et l'année. Il affiche ensuite la date avec le nom du mois.

Exercice 2:

Réaliser en C un algorithme qui permet de saisir un numéro de couleur de l'arc-en-ciel et d'afficher la couleur correspondante :

1: rouge, 2 : orangé, 3 : jaune, 4 : vert, 5 : bleu, 6 : indigo et 7 : violet.

II. Instructions conditionnelles

```
#include <stdio.h>
void main() {
unsigned int j, m, a;
printf (“donner le jour puis le moi puis l’année”);
scanf(“%2u %2u %4u”, &j, &m, &a);
   switch (m) {
        case 1: printf (“%u janvier %u”, j, a); break ;
        case 2: printf (“%u fevrier %u”, j, a); break ;
        ...
        ...
        case 12: printf (“%u decembre %u”, j, a); break ;
        default: printf ("erreur") ;
   }
}
```

III. Instructions itératives

Les instructions itératives sont commandées par trois types de boucles :

- **le for**
- **le while**
- **le do while**

III. Instructions itératives

1) La boucle for

Syntaxe :

```
for ( exp1 ; exp2; exp3 ) {
    /*bloc d'instructions*/
}
```

Remarques :

- Le for s'utilise avec trois expressions, séparées par des points virgules, **qui peuvent être vides.**

- Les { } ne sont pas nécessaires lorsque le bloc ne comporte qu'une seule instruction.

III. Instructions itératives

1) La boucle for

1. **l'expression expr1** est réalisée une seule fois lors de l'entrée dans la boucle, nous l'appellerons expression d'initialisation ;
2. **l'expression expr2** est la condition d'exécution de l'instruction. Elle est testée à chaque itération, y compris la première. Si l'expression expr2 prend la valeur vraie l'instruction contrôlée par le for est exécutée, sinon la boucle se termine ;
3. **l'expression expr3** contrôle l'avancement de la boucle. Elle permet de manière générale de calculer la prochaine valeur avec laquelle la condition de passage va être ré-testée, elle est exécutée après l'instruction à chaque itération avant le nouveau test de passage.

III. Instructions itératives

1) La boucle for

Exemples:

```
for (i = 0 ; i < 10 ; i++) {
        printf("%d",i);

}
```

```
i=0 ;
for ( ; i < 10 ; ) {
    printf("%d",i);
    i++;
}
```

```
for (i = 0 , j = 10 ; i < j ; i++ , j--) {
        printf("%d %d", i, j);
}
```

```
i=0 ;
for (  j=10; i < j ; j-- ) {
   printf("%d %d", i, j);
   i++ ;
}
```

III. Instructions itératives

1) La boucle for

Exercice 1:

Un nombre réel X et un nombre entier N étant donnés, proposer un programme C qui fait calculer X^N. Étudier tous les cas possibles (N positive ou négative).

Exercice 2:

Un entier naturel de trois chiffres est dit cubique s'il est égal à la somme des cubes de ses trois chiffres.

Exemple :

153 est cubique car $153 = 1^3 + 5^3 + 3^3$

Réaliser en C un algorithme qui cherche et affiche tous les entiers cubiques de trois chiffres.

III. Instructions itératives

2) La boucle while

Le **while** répète le bloc d'instructions tant que la valeur de l'expression s'interprète comme vraie (**différente de zéro**).

Si expression **est nulle** le bloc d'instructions **ne sera jamais exécuté**.

Syntaxe :

```
while (expression) {
                ……. ;
                ……;  /*bloc d'instructions*/
                ……;
 }
```

III. Instructions itératives

2) La boucle while

Exemple :

```
int i=1;
while (i < 10) {
    printf ("i = %d \n ", i) ;
    i++;
}
```

Remarques :

- Les { } ne sont pas nécessaires lorsque le bloc ne comporte qu'une seule instruction.
- Le traitement s'exécute 0 ou n fois !

III. Instructions itératives

2) La boucle while

Exercice 1:

Réaliser en C un algorithme qui permet de déterminer la somme des chiffres d'un nombre entier positif donné.

Exemple :

pour N= 25418, on aura 2+5+4+1+8 = 20

III. Instructions itératives

3) La boucle do while

- A l'inverse du **while**, le **do while** place son test en fin d'exécution, ceci assure que l'instruction est réalisée **au moins une fois**.
- Il ressemble au **REPEAT UNTIL** du langage **PASCAL**.

Syntaxe :

```
do {
    ..... ;
    .....;  /*bloc d'instructions*/
    .....;
} while (expression) ; /* !!! N'oublier pas le ; */
```

III. Instructions itératives

3) La boucle do while

Exercice 1:

En utilisant la boucle do while, réaliser en C un algorithme qui cherche et affiche tous les entiers cubiques de trois chiffres.

Exercice 2:

Écrire un programme C qui permet de lire un entier positif et déterminer tous ses facteurs premiers.

Exemples:

30= 2 * 3 * 5 | 36= 2 * 2 * 3 * 3 | 99= 3 * 3 * 11

IV. Les instructions de branchement non conditionnel

1) L'instruction continue

- Elle provoque le **passage à l'itération suivante** de **la boucle** en sautant à la fin du bloc.

➔ Elle provoque la non-exécution des instructions qui la suivent à **l'intérieur du bloc**.

Exemple :

```
void main() {
   int i;
   for (i = 0; i < 6; i++) {
         if (i==3)
                  continue;
         printf ("i = %d \t", i);
   }
   printf ("\n La valeur de i a la sortie de la boucle = %d", i);
}
```

Cet exemple imprime

i= 0 i=1 i=2 i = 4 i = 5

La valeur de i a la sortie de la boucle = 6

IV. Les instructions de branchement non conditionnel

2) L'instruction break

- L'instruction break permet de sortir d'un bloc d'instruction associé à une instruction **répétitive** ou **alternative** contrôlée par les instructions **if**, **switch**, **for**, **while** ou **do while.**

Exemple :

```
void main () {
  int i;
  for (i = 0; i < 6; i++) {
     printf ("i = %d \t ", i) ;
     if (i==3)
         break ;
  }
 printf ("\n La valeur de i a la sortie de la boucle = %d", i);
}
```

Cet exemple imprime
i= 0 i=1 i=2 i=3
La valeur de i a la sortie de la boucle = 3

IV. Les instructions de branchement non conditionnel

3) L'instruction goto

- L'instruction **goto** permet de sauter à un point précis du programme que nous aurons déterminé à l'avance.
- Pour ce faire, le langage C nous permet de marquer des instructions à l'aide d'étiquettes (labels en anglais).

Exemple :

```
void main( ) {
int i = 0;
condition:
        if (i < 5) {
                printf("La variable i vaut %d \n", i);
                i++;
                goto condition;
        }
}
```

IV. Exercices d'application

Exercice 1 :

Un nombre est dit palindrome s'il est écrit de la même manière de gauche à droite ou de droite à gauche.

Exemples : 101 ; 22 ; 3663 ; 10801, etc.

Écrire un programme C permettant de déterminer et d'afficher tous les nombres palindromes compris dans l'intervalle [100..9999].

Exercice 2 :

Réaliser en C un algorithme qui affiche la suite de tous les nombres parfaits inférieurs ou égaux à un nombre naturel non nul donné noté n. Un nombre est dit parfait s'il est égal à la somme de ses diviseurs autre que lui-même.

Exemple: 28 = 1 + 2 + 4 + 7 + 14

Voici la liste des nombres parfaits inférieurs à 10000 : 6, 28, 496, 8128.

IV. Exercices d'application

Exercice 3 :

Les nombres de Fibonacci sont donnés par la récurrence :

$F_n = F_{n-2} + F_{n-1}$ avec $F_0 = 1$ et $F_1 = 1$.

Écrire un programme C qui affiche les 20 premiers nombres de Fibonacci.

Exercice 4 :

Deux entiers N1 et N2 sont dits frères si chaque chiffre de N1 apparaît au moins une fois dans N2et inversement.

Exemples :

Si N1 = 1164 et N2 = 614 alors le programme affichera :

N1 et N2 sont frères

Si N1 = 405 et N2 = 554 alors le programme affichera :

N1 et N2 ne sont pas frères

Écrire un programme C qui saisit deux entiers N1 et N2, vérifie et affiche s'ils sont frères ou non.

IV. Exercices d'application

Exercice 5 :

Dans une entreprise, le calcul des jours de congés payés s'effectue de la manière suivante :

- Si une personne est rentrée dans l'entreprise depuis moins d'un an, elle a le droit à deux jours de congés par mois de présence, sinon à 28 jours au moins.
- Si c'est un cadre, s'il est âgé d'au moins 35 ans et si son ancienneté est supérieure ou égale à 3 ans, il lui est accordé deux jours supplémentaires. Si ce cadre est âgé d'au moins 45 ans et si son ancienneté est supérieure ou égale à 5 ans, il lui est accordé 4 jours supplémentaires, en plus des 2 accordés pour plus de 35 ans.

Réaliser en C l'algorithme qui calcule le nombre de jours de congés à partir de l'âge, de l'ancienneté et de l'appartenance au collège cadre d'un employé. Afficher le résultat.