1ière Année Licence Informatique – Algorithmique et structure de données 1 (ASD1)

Chapitre 3 : Les structures conditionnelles (en langage algorithmique et en langage C)

Contenu du Chapitre :

1. Introduction

Nous avons vu dans le chapitre 2 qu'il y a 2 types d'actions (algorithmique et langages de programmation, ici langage C) :

- Les actions simples : Lecture, Ecriture, Affectation
- Les actions de contrôle : Séquence naturelle, les actions conditionnelles (rupture de la séquence naturelle par saut en avant) et les boucles (rupture de la séquence naturelles par saut en arrière).

Dans ce chapitre, nous abordons les actions conditionnelles.

2. Les actions conditionnelles (on dit aussi, structures algorithmique conditionnelles) :

Définition : Une action *Action1* est dite conditionnelle, si elle est exécutée seulement dans le cas où une condition de test *Cond* est vraie.

La condition de test *Cond* est une expression *logique* ou *arithmétique et logique*. Pour des raisons de simplicité, nous dirons : expression logique.

Il y a alors trois formes de structures algorithmiques conditionnelles :

a. *Si Simple* :

Syntaxe en langage algorithmique :

Si (*Cond*) alors **// Condition est une expression logique**

Action1 ;

Action2 ;

….

Finsi

Effet :

Lorsque *Cond* == vrai : Il y a exécution des actions entre *alors* et *finsi*

Autrement, il y a poursuite de l'exécution après le *finsi*, sans exécution des actions entre alors et finsi.

Nous disons qu'il y a un saut du point d'exécution en avant. Comme déjà signalé dans les chapitres précédents, ceci constitue une forme de rupture de la séquence naturelle des actions de l'algorithme (ou programme).

Exemple :

L'algorithme suivant vérifie si la température lue dépasse 29 degrés, auquel cas il affiche « Température élevée ». Autrement, il ne fait rien (nous disons qu'il y a continuation en séquence).

```
Algorithme AlerteHighTemp
Données  t : réel ;
Résultats
Variables
  Début
      Lire(t) ;
      Si (t>29) alors                    // Cond : (t>29)
         Ecrire(''High Temperature'')     // Action conditionnelle
      Finsi
Fin
```

Syntaxe en langage C :

```
If (Cond) {                          // les actions conditionnelles sont entre { et }
    Action1 ;
    ….
}
```

Traduction de l'algorithme précédent en langage C:

```
#include <stdio.h>
int main() {
   float t;
   printf("Exemple de if en Langage C. \n");
   printf("Temprature: ");
   scanf("%f", &t);

   // Message d'alerte lorsque t>29
   if (t > 29) {
      printf("High Temperature");
   }
  }
```

Exécution :

Exemple de if en Langage C.
Temprature: 31

High Temperature

Forme Organigramme :

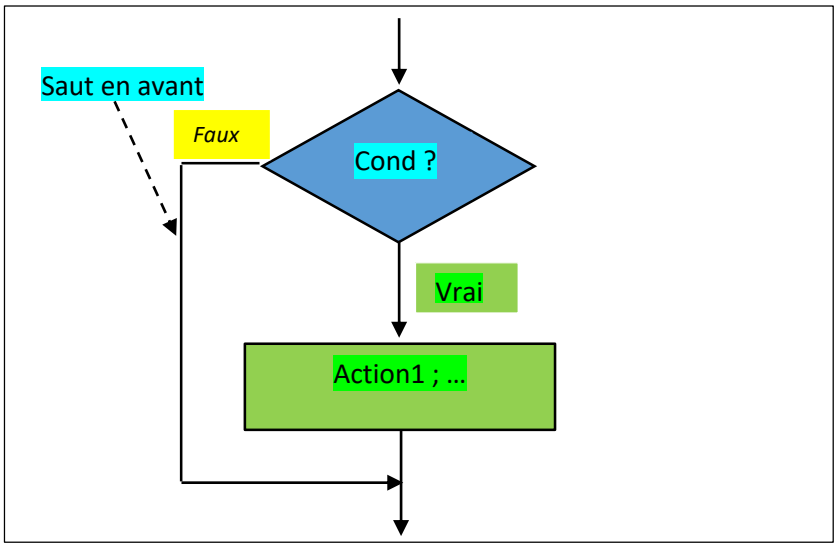

Figure 3.1 Forme Organigramme de l'action conditionnelle : Si..alors…Finsi

b. Si double alternative : (Si …alors…sinon…finsi) :

Syntaxe en langage algorithmique :

Si (*Cond*) alors

Action a1 ;

Action a2 ;

….

Sinon

Action b1 ;

Action b2 ;

…..

Finsi

Effet :

Lorsque *Cond* == vrai, il y a exécution des actions entre *alors* et *sinon* *(action a1 ; …)*.

Lorsque la *Cond* == faux, il y a exécution des actions entre *sinon* et *finsi* *(action b1 ; …)*.

Exemple :

Nous reprenons l'exemple de la température, avec envoi de message dans les deux cas : t>29 et t≤29.

```
Algorithme AlerteHighTemp2
Données  t : réel
Résultats
Variables
  Début
      Lire(t) ;
      Si (t>29) alors
          Ecrire("High Temperature")
      Sinon
          Ecrire(" Il fait frais")
      Finsi
Fin
```

Syntaxe en langage C :

```
If (Cond) {
    Action a1 ;
    Action a2 ;
     ….
}
Else {
 Action b1 ;
   Action b2 ;
…..
}
```

Exemple : Programme de la température en langage C :

```
#include <stdio.h>
int main()
{
  float t;
  printf("Entrez la température: ");
  scanf("%f", &t);
  if (t>29.0)
  {
     printf("\n Il fait chaud ! ");
  }
  else
  {
     printf("\n Il fait frais");
  }
  }
```

Exécution :

Entrez la température: 25.2
Il fait frais

Forme Organigramme (Si Cond Alors Actiona1 ; …Sinon Actionsb1 ;…Finsi)

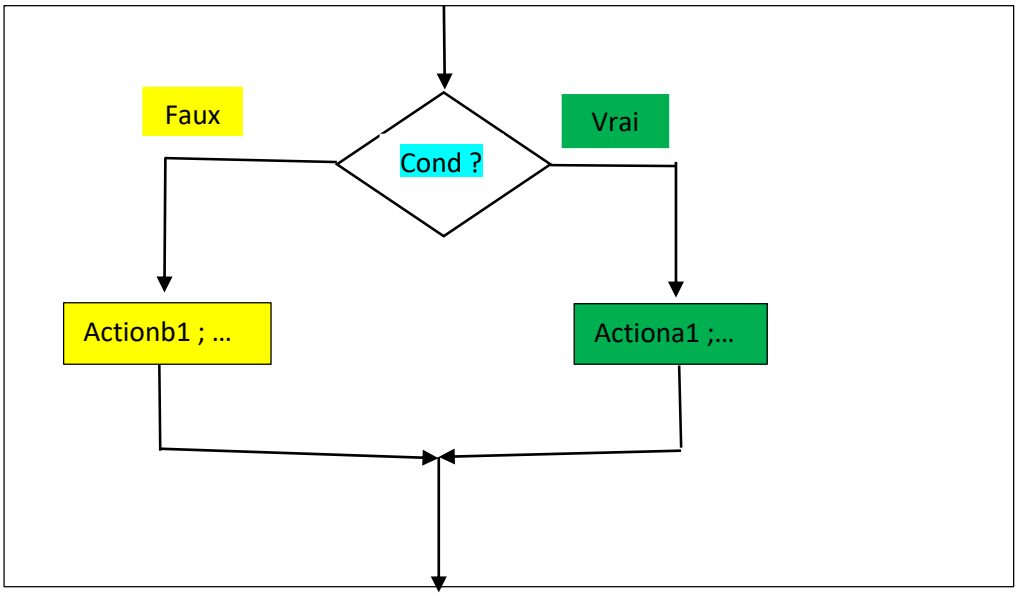

Figure 3.2 Forme Organigramme de l'action conditionnelle : Si..alors…Sinon…Finsi

c. Structure conditionnelle à choix multiples (ou Si à plusieurs alternatives)

Ce type de structure algorithmique est basé sur l'existence de plusieurs blocs d'actions B1, B2, …, Bn, en plus d'un bloc Bn+1 (facultatif : non obligatoire). Chaque bloc d'action Bi est conditionné par une valeur vi d'un sélecteur de type discret : (Caractère, Entier) (version1) ou une condition logique Ci (version2).

Remarque : Les conditions Ci doivent être mutuellement exclusives. De même, les valeurs vi doivent être aussi mutuellement exclusives.

Version1 : Le choix basé sur la valeur d'un *selecteur* :

Syntaxe en langage algorithmique :

Selon *selecteur* ***faire***
Valeur1 : action 1 ;
Valeur2 : action 2 ;

Valeur n : action n ;
sinon
action n+1 ;
finselon ;

Où :

- Sélecteur est une variable qui prend des valeurs discrète (entier ou caractère)
- valeur1, valeur2, … : les valeurs possibles pour sélecteur.
- action 1 ; action 2 ; … ; action n : les actions à exécuter correspondants aux valeurs vi.
- Action n+1 : une action à exécuter lorsqu'aucune valeur vi n'est égale au sélecteur.

Effet :

Lorsque selecteur == valeuri, il y a exécution de Actioni, i=1 :n ; puis, il y a saut après ***finselon***

Lorsque la valeur du selecteur est différente de toutes les valeurs valeur1, .., valeurn, i=1..n, il y a exécution de *action*(n+1), si elle existe (Le bloc sinon est facultatif).

Chapitre 3 : Les structures conditionnelles (en langage algorithmique et en langage C)

Exemple : Ecrire un algorithme *Mois123* qui saisit une valeur M de type entier et affiche le mois correspondant en forme texte, lorsque : 1≤ M ≤ 3 et le message, « Valeur de M hors intervalle », autrement.

```
Algorithme Mois123
Données  M : entier
Résultats
Variables
Début
Ecrire ("Donner le numero du Mois: 1 a 3 ")
Lire (M)
Selon M faire
1 : Ecrire ("Janvier")
2 : Ecrire ("Fevrier")
3 : Ecrire ("Mars")
Sinon
   Ecrire ("Valeur de M hors intervalle  ")
FinSelon
Fin
```

Syntaxe de l'action si à choix multiple en langage C :

```
switch(selecteur){
case valeur1:
  Action1;
    break;
case valeur2:
  Action2;
 break;
.....

default:
Action(k+1);
}
```

En langage C, la même démarche est appliquée avec l'action *switch* (selecteur). Le rôle de l'action *break* est de terminer l'action switch, suite à l'exécution d'une action i, i=1..n. Lorsque, aucune valeur i ne correspond au sélecteur, il y a exécution de l'action (n+1), si elle existe.

Exemple en langage C : Algorithme Mois123

```
#include <stdio.h>
int main() {
int m;
printf("Donnez numero du mois: ");
scanf("%d",&m);
switch (m) {
case 1:
printf("\n le mois est : Janvier \n");
break;
case 2:
printf("\n le mois est : Fevrier \n");
break;
case 3:
printf("\n Le mois est : Mars \n");
break;
default:
printf("La valeur est Hors intervalle 1..3");
break;
}
}
```

Exécution :

Donnez numero du mois: 3

Le mois est : Mars

Regroupement de plusieurs valeurs dans un intervalle dans l'instruction switch :

En langage C, plusieurs valeurs du sélecteur peuvent être regroupées dans un intervalle (des valeurs de type entier ou caractère, consécutives. Exemples : 0 … 10 , 'a' … 'z', etc.) associées à une seule action particulière.

Voici un exemple de programme en langage C où le sélecteur prend des valeurs dans des intervalles.

Problème : Lire un entier (x) et trouver son intervalle par palier de 10, jusqu'à un maximum de 40 : 1 à 10, 11 à 20, 21 à 30, 31 à 40.

Les valeurs x<1 ou x>40 seront déclarées « hors intervalles 1 à 40 ».

Voici un programme en langage C qui réalise ce travail :

```c
#include <stdio.h>
int main() {
  int x;
  printf(" Enter x = ");
  scanf("%d",&x);
  printf(" \n x= %d \n",x);
  switch (x) {
      case 1 ... 10:                    // remarquez les espaces entre 1 et … et … et 10
        printf("  1<= x <=10");
      break;
      case 11 ... 20:
        printf("  11<= x <=20");
      break;
      case 21 ... 30:
        printf("  21<= x <=30");
      break;
      case 31 ... 40:
        printf("  31<= x <= 40");
      break;
      default:
        printf(" x<1 ou x>40 ");
      break;
    }
 }
```

Exécution1 :

Enter x = 18

x= 18

11<= x <=20

Exécution 2 :

Enter x = -2

x= -2

x<1 ou x>40

Exécution 3

Enter x = 89

x= 89

x<1 ou x>40

Regroupement de valeurs du sélecteur selon plusieurs valeurs non consécutives :

L'instruction switch du langage C, permet aussi de regrouper plusieurs valeurs non obligatoirement consécutives, associées à la même action. Voici un exemple.

Problème : Ecrire un programme en langage C qui saisit un nombre entier x ≤ 10 et affiche un message informant si x est un nombre pair ≤ 10 ou x est un nombre impair ≤ 10.

Autrement, il affiche : x ≤ 0 ou x ≥ 10.

```
#include <stdio.h>
int main(void) {
   int x;
   printf(" Donnez  x = ");
   scanf("%d",&x);
   switch(x)   {
      case 1:
      case 3:
      case 5:
      case 7:
      case 9:
         printf("x = %d  est Nombre impair <= 10 \n", x);
         break;
      case 0:
      case 2:
      case 4:
      case 6:
      case 8:
      case 10:
         printf("x= %d  est Nombre pair <= 10 \n", x);
         break;
      default:
         printf("x = %d : x<0 ou x>10 \n", x);
   }
}
```

Exécution 1 :

Donnez x = 5

x = 5 est Nombre impair <= 10

Exécution 2 :

Donnez x = 8

x= 8 est Nombre pair <= 10

Exécution 3 :

Donnez x = 89

x = 89 : x<0 ou x>10

Version2: Dans cette version du si à alternatives multiples, il y a utilisation de structures *if-else imbriquées*.

Syntaxe en langage C :

```
If (cond1) {
    Action1;
  }
Else if (cond2) {
     Action2;
....
Else
   Action(k+1);
```

Exemple en langage C :

Nous reprenons l'exemple de la température avec trois paliers :

- t<=29 : Temps Frais
- 29< t ≤ 35 : Temps Chaud
- t>35 : Grande Chaleur.

Voici un programme en langage C qui réalise le travail demandé :

```
#include  <stdio.h>
int main() {
   float t;
   printf("Quelle est la temperature ? \n: ");
   scanf("%f",&t);
   if (t <= 29.) {
      printf("Temps Frais \n");
   } else {
      if (t <= 35.) {
         printf("Temps Chaud \n");
        }
      else {
         printf("Grande Chaleur \n");
        }
     }
  }
```

Exécution :

```
Quelle est la temperature ? : 25
Temps Frais
```

d. L'action *goto* (*allerà*) :

L'action *goto* permet de se brancher, directement, à une action particulière dans le programme. Cette action est repérée par une variable de type *étiquette* (*label*), associée à l'action désirée où l'on veut 'sauter' pour continuer l'exécution du programme. Ces sauts sont de même type que les sauts des actions d'alternatives (Saut en avant) et de boucles (saut en arrière) :

- **Saut en avant par l'action *goto* :**

Syntaxe en langage algorithmique :

Action 1 ;
Action 2 ;
Si (Cond) alors ***goto*** à etiq1
Action 3 ;
Finsi
etiq1 : Action 4 ;

Exemple : Avec la structure algorithmique *allerà* (goto), écrire un algorithme qui saisit un nombre entier signé et calcule sa valeur absolue.

Algorithme en langage Algorithmique :

```
Algorithme AbsEntier
Données A : entier
Résultats A : entier                    // ou Données/Résultats  A : entier
Début
Ecrire(''Donnez A :'')
Lire(A)
Si(A>0) alors goto etiq1
   A ← -A
Finsi
Etiq1 : Ecrire(''Valeur absolue de A='',A)
Fin AbsEntier
```

Algorithme équivalent en langage C :

```
#include <stdio.h>
int main(){
  int a;
  printf("a= ");
  scanf("%d",&a);
  if(a>0) goto etiq1;
  a*=-1;                                  // a=-a ou a= a*(-1)
etiq1:
  printf("abs(a)= %d",a);
}
```

Exécution :

```
a= -2
abs(a)= 2
```

- **Saut en arrière par l'action *goto***

  Dans ce cas, il y a branchement à une *action antérieure*, suite à la satisfaction d'une condition Cond.

Syntaxe en langage algorithmique :

Etiq1 : Action 1 ;

Action 2 ;

Si (Cond) alors **goto** à **etiq1**

finsi

Action 2 ;

Exemple : Avec la structure algorithmique *goto*, écrire un algorithme qui saisit un nombre entier positif n et calcule la somme : 1+2+3+ … + n

```
Algorithme SomArithm
  Données n: Entier
  Résultats  Som : entier
  début
  Ecrire('Donnez n :')
  Lire(n)
  Som ← 0
etiq1 :  Som ← Som + n ;
  n ← n-1;
  Si(n>0) alors goto etiq1                // Saut en arrière
  Ecrire(''Som: n + n-1 +…3+2+1='',Som)
  Fin SomArithm
```

Programme équivalent en langage C :

```
#include <stdio.h>
int main(){
  int n, som=0;
  printf("Donnez n :");
  scanf("%d",&n);
  etiq1 :
  som +=n;                          // som = som +n
  --n;                              // n=n-1 ;      préfixé
  if(n>0){
       goto etiq1;
  }
  printf("Som: 1 + 2 + …+ n =  %d ",som);
}
```

Exécution :
Donnez n : 6
Som: 1 + 2 + …+ n = 21

Remarque : L'action *goto* est généralement déconseillée (à cause des règles imposées par la programmation structurée) : Elle peut compliquer la logique et la compréhension des programmes, ce qui conduit à un suivi et une maintenance plus difficile des programmes.