Ministère de l'enseignement supérieur et de la recherche scientifique
Université Abderrahmane MIRA de Bejaia

Faculté de Technologie
Département de Technologie



Structure des ordinateurs et applications

À propos du cours

- **Crédits**: 02
- **Coefficients**:02

Notion d'algorithme et de programme

Structures de contrôles

En générale, les instructions d'un programme sont exécutés d'une manière séquentielle : la première instruction, ensuite la deuxième, après la troisième et ainsi de suite. Cependant, dans plusieurs cas, on est amené soit à choisir entre deux ou plusieurs chemins d'exécution (un choix entre deux ou plusieurs options), ou bien à répéter l'exécution d'un ensemble d'instructions, pour cela nous avons besoins de structures de contrôle pour contrôler et choisir les chemins d'exécutions ou refaire un traitement plusieurs fois.

Les structures de contrôle sont de deux types :

- Structures de contrôle conditionnelle (Tests)
- Structures de contrôle répétitives (itératives, boucles).

Structures de contrôle conditionnelle

Ces structures sont utilisées pour décider de l'exécution d'un bloc d'instruction : est ce que ce bloc est exécuté ou non. Ou bien pour choisir entre l'exécution de deux blocs différents. Nous avons trois types de structures conditionnelles :

Test alternatif simple

Un test simple contient un seul bloc d'instructions. Selon une condition (expression logique), on décide est ce que le bloc d'instructions est exécuté ou non. Si la condition est vraie, on exécute le bloc, sinon on l'exécute pas.

Tableau II.14 : La syntaxe d'un test alternatif simple

Algorithme	Langage C

si <condition> alors
<bloc_instructions_si> ;
finsi;
	if (condition){
	<bloc_instructions_si>;
	}

Remarque :

Dans le langage C, un bloc est délimité par les deux accolades **{** et **}**. Si le bloc contient une seule instruction, les accolades **{** et **}** sont facultatifs (on peut les enlever)

Test alternatif double

Un test double contient deux blocs d'instructions : on est amené à décider entre le premier bloc ou le seconds. Cette décision est réalisée selon une condition (expression logique ou booléenne) qui peut être vraie ou fausse. Si la condition est vraie on exécute le premier bloc, sinon on exécute le second.

Algorithme	**Langage C**
si <condition> **alors** <bloc_instructions_si>**sinon**<bloc_instrucitons_sinon> ;**finsi**	**if** (condition) **{**<bloc_instructions_si>;**} else** **{**<bloc_instructions_sinon>;**}**

Remarques :

- Dans le langage C, il faut jamais mettre de point virgule après la condition (erreur logique).

Structure conditionnelle multiple

La structure conditionnelle multiple permet de tester plusieurs conditions et d'exécuter les instructions correspondant à la première condition vraie.

Syntaxe de la structure conditionnelle multiple

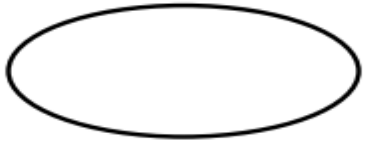
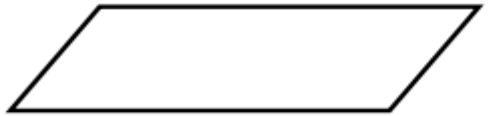
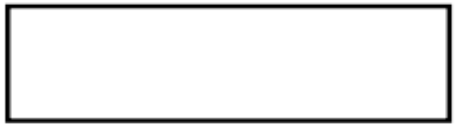
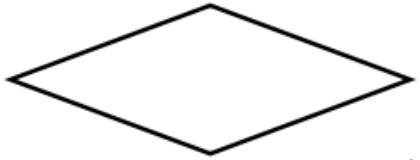
Algorithme	Langage C

Si <condition1> Alors
 	
<instruction(s)1>
 	
Sinon Si <condition2> Alors
 	
<instruction(s)2>
 	
Sinon <action_par_défaut>
 	
FinSi;
	if (condition1) {
	  // Instruction(s) 1
	}
	else if (condition2) {
	  // Instruction(s) 2
	}
	else {
	  // Action par défaut
	}

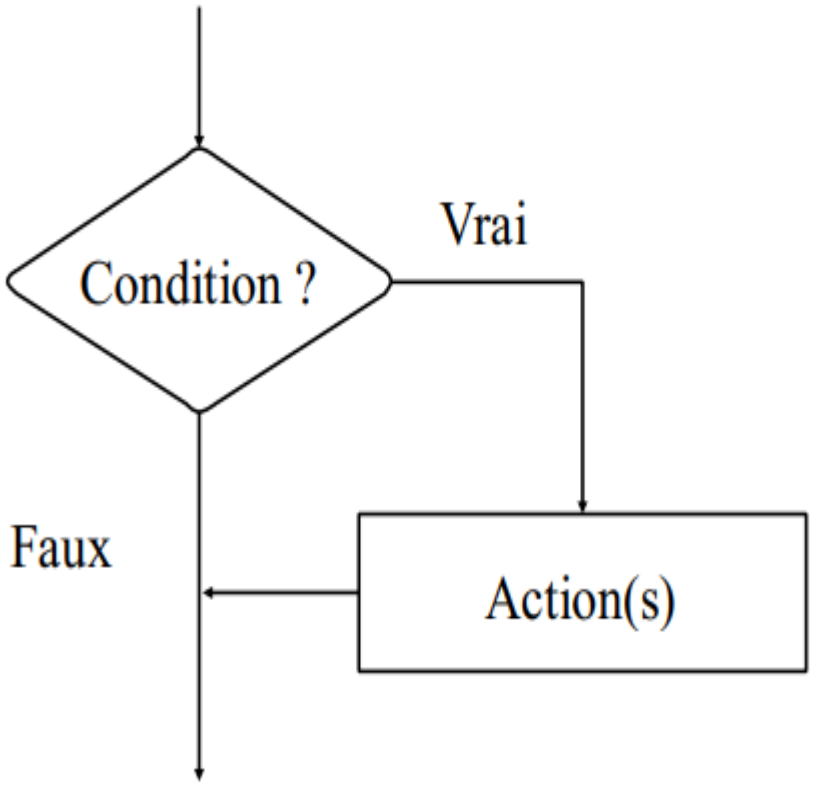
Représentation en organigramme

Un organigramme est la représentation graphique de la résolution d'un problème. Il est similaire à un algorithme. Chaque type d'action dans l'algorithme possède une représentation dans l'organigramme.

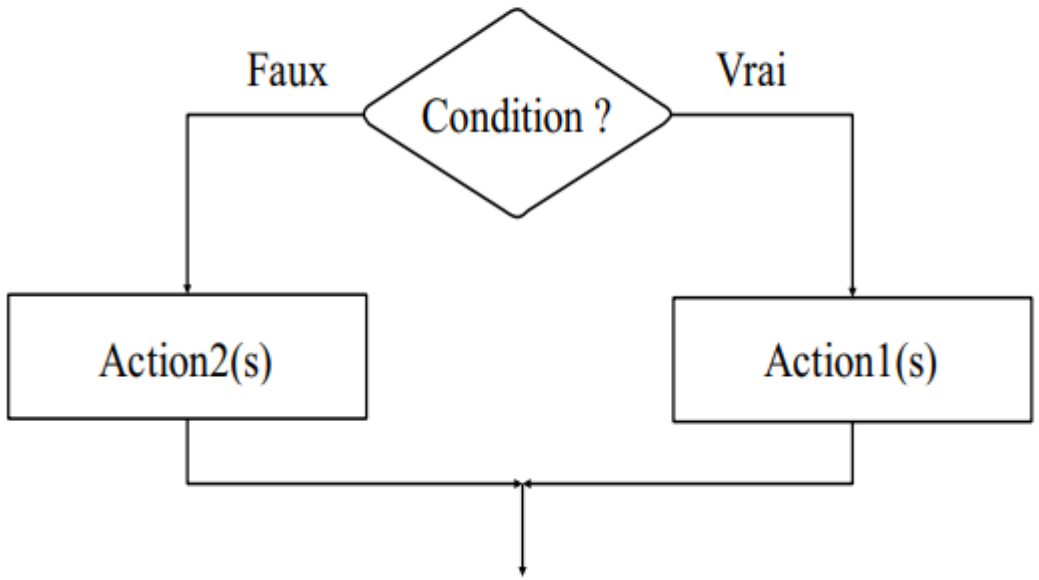
Les symboles d'organigramme

 ovale	Représente le début et la Fin de l'organigramme
 (parallélogramme)	Entrées / Sorties : Lecture des données et écriture des résultats.
	Calculs, traitements
 losange	Tests et décision : on écrit le test à l'intérieur du losange
	Ordre d'exécution des opérations (Enchaînement)
	Connecteur

La syntaxe et l'organigramme de la structure alternative simple

Représentation algorithmique	Représentation sous forme d'organigramme
si <condition> **alors** <Action(s)>; **finsi**; Si la condition est vérifiée, le bloc <*action(s)*> sera exécuté, sinon rien, et on continue l'exécution de l'instruction après fin si.	

La syntaxe et l'organigramme de la structure alternative double

Représentation algorithmique	Représentation sous forme d'organigramme
si <condition> **alors** <Action1(s)> **sinon** <Action2(s)>; **finsi**; Si la condition est vérifiée, le bloc *<action1(s)>* sera exécuté, sinon (si elle est fausse) on exécute *<action2(s)>*.	

Exercice 01

Traduire l’algorithme en un programme en langage C.

Il existe trois cas :

1. Si a est différent de 0 : une seule solution, $x=-b/a$.
2. Si a = 0 et b = 0 : une infinité de solutions.
3. Si a = 0 et b différent de 0 : aucune solution.

```
Algorithme Resolution_Equation;
Variables
   a, b, x : Réel;
Début
   Ecrire("Donner la valeur de a : ");
   Lire(a);
   Ecrire("Donner la valeur de b : ");
   Lire(b);
   Si (a <> 0) Alors
      x ← -b / a;
      Ecrire("La solution est : ", x);
   Sinon
      Si (a = 0 ET b = 0) Alors
         Ecrire("L'equation admet une infinite de solutions");
      Sinon
         Ecrire("L'equation n'admet aucune solution");
      FinSi;
   FinSi;
Fin
```

Exercice 02: (– Comparaison entre deux nombres)

1/ Écrire l'algorithme puis le programme **C** qui lit deux nombres entiers a et b, et affiche :

- 'a est plus grand que b' si a > b;
- ‘b est plus grand que a' si a < b;
- 'a est égal à b' si a = b.