

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

Présentation

- ✓ Dr. DJAFRI Ghani
- ✓ ghani.djafri@univ-bejaia.dz



À propos du cours

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

Notion d'algorithme et de programme

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. Mais parfois, il faut :

- **Faire un choix** entre plusieurs actions (exemple : si une condition est vraie, faire ceci, sinon faire autre chose).
- **Répéter une action** plusieurs fois (exemple : faire une tâche tant qu'une condition est vraie).

Pour gérer cela, on utilise **des structures de contrôle** :

- ✓ Structures conditionnelles : pour décider quel chemin suivre selon une condition.
- ✓ Structures répétitives (ou itératives) : pour répéter des instructions plusieurs fois.

Notion d'algorithme et de programme

- Structures de contrôles

- 1. Structures de contrôle conditionnelle

Les **structures conditionnelles** servent à décider si une ou plusieurs instructions doivent s'exécuter en fonction d'une condition.

Il existe **trois types de tests** :

- **Test alternatif simple** : on exécute une action **si la condition est vraie**, sinon on ne fait rien.
- **Test alternatif double** : on exécute une action **si la condition est vraie**, et une autre action **si elle est fausse**.
- **Test alternatif multiple** : Permet de tester plusieurs conditions successives et d'exécuter différentes actions selon celle qui est vraie. Si aucune condition n'est vraie, on peut exécuter une action par défaut.

Test alternatif simple

| Algorithme | Pascal |
|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| si <condition> alors <instruction(s)> finsi ; | if <condition> then begin <instruction(s)>; end ; |

Exemples :

| Algorithme | Pascal |
|-----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| lire(x) si $x > 2$ alors $x \leftarrow x + 3$; finsi ; écrire (x) | read(x); if $x > 2$ then begin $x := x + 3$; end ; write(x); |

Remarque : En Pascal, le bloc d'instructions à exécuter après if (juste après then) DOIT être délimité par un **Begin** et **end**. Si le bloc contient une seule instruction, begin et end sont facultatifs (on peut les enlever).

Test alternatif double

| Algorithme | Pascal |
|------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| si <condition> alors <instruction(s)1> sinon <instruciton(s)2> ; finsi ; | if <condition> then begin <instruction(s)1> ; end else begin <instruction(s)2> ; end; |

Exemples :

| Algorithme | Pascal |
|---------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lire(x) si x > 2 alors x ← x + 3 ; finsi ; écrire (x) sinon x ← x - 2 finsi écrire (x) | read(x); if x > 2 then begin x:= x + 3; end else begin x:= x - 2; end; write(x); |

Remarques :

- En Pascal,
l'instruction qui
précède **else** ne doit
pas contenir un point
virgule (;).

Test alternatif double

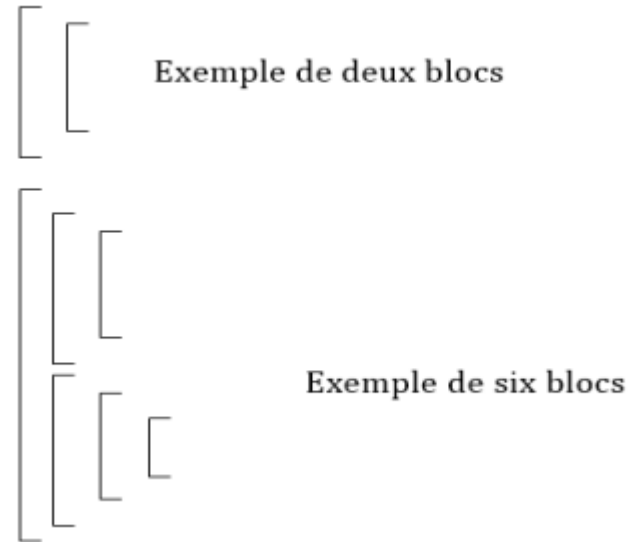
| Algorithme | Pascal |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>si <condition> alors <instruction(s)1> sinon si <condition> alors <instruciton(s)2> sinon action_par_defaut fin si</pre> | <pre>if <condition> then begin <instruction(s)1> ; end else if <condition> then begin <instruction(s)2> ; end Else begin action_par_defaut end;</pre> |

Remarque

Lorsque l'action après THEN, ELSE ou un DO comporte plusieurs instructions, on doit obligatoirement encadrer ces instructions entre BEGIN et END. Autrement dit, on les définit sous forme d'un bloc. Pour une seule instruction, il n'est pas nécessaire (ou obligatoire) de l'encadrer entre BEGIN et END (voir en travaux pratiques). Un ensemble d'instructions encadrées entre BEGIN et END, s'appelle un BLOC ou action composée. On dit qu'un programme Pascal est structuré en blocs.

Remarques importantes

- ✓ Langage Pascal est insensible à la casse, c'est-à-dire, si on écrit begin, Begin ou BEGIN c'est la même chose.
- ✓ En Pascal, on ne met pas de point-virgule ; après « **then** ».
- ✓ Les boucles et blocs doivent en aucun cas chevaucher, ils doivent être imbriqués.



Exemple 1: (– Test alternatif simple)

Écrire un programme qui demande à l'utilisateur un nombre et affiche un message si le nombre est positif.

Exemple 2: (– Test alternatif double)

Écrire un programme qui demande à l'utilisateur un âge et affiche :
« Majeur » si l'âge est supérieur ou égal à 18
« Mineur » sinon

Exemple 3: (– Test multiple)

Écrire un programme qui demande une note (0 à 100) et affiche :
« Excellent » si la note ≥ 90
« Bien » si la note ≥ 75
« Passable » si la note ≥ 50
« Échec » sinon