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

La structure principale d'un programme

```
#include <stdio.h>
int main() {
    // instructions
    return 0;
}
```

#include <stdio.h> : pour les fonctions d'entrée/sortie (printf, scanf).
int main() : point d'entrée du programme.
return 0; : indique que le programme s'est terminé correctement.

Les expressions logiques et comparateurs

Comparateurs : ==, !=, <, >, <=, >=

•Opérateurs logiques : && (ET), || (OU), ! (NON)

•Important : **= est une affectation**, **== est une comparaison**

- **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.

- **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	En langage C
si <condition> **alors** <instruction(s)> **finsi**;	**if (condition) {** **// instructions ;** **}**

Exemple

lire(x)
si x > 2 **alors**
x ← x + 3 ;
finsi ;
Ecrire (x)

```
printf("Donner la valeur de x : ");
    scanf("%d", &x);
if (x > 2) {
        x = x + 3;
    }
printf("La valeur de x est : %d\n", x);
```

Test alternatif double

Algorithme	En langage C
si <condition> alors <instruction(s)1> **sinon** <instruciton(s)2> ; **finsi ;**	**if (condition) {** **// instructions ; }** **else {** **// instructions ;** **}**

Exemples :

Si x > 2 Alors
    x ← x + 3
    Ecrire("La valeur de x après ajout : ", x)
  Sinon
    x ← x - 2
    Ecrire("La valeur de x après soustraction : ", x)
  FinSi;

```
if (x > 2) {
        x = x + 3;
        printf("La valeur de x
après ajout : %d\n", x);
    } else {
        x = x - 2;
        printf("La valeur de x
après soustraction : %d\n", x);
    }
```

Test alternatif multiple

Algorithme	En langage C

Si condition1 Alors
instruction(s)1
SinonSi condition2 Alors
instruction(s)2
Sinon
action_par_defaut
FinSi;
	if (condition1)
	{// instructions1;}
	else if (condition2)
	{// instructions2;}
	else
	{// instructions ;}

Remarque :la condition d'un test (comme pour if, while ou for) doit toujours être mise entre parenthèses.

Exemples :

```
Si x > 2 Alors
   x ← x + 3
   Ecrire("x après ajout : ", x)
Sinon
   Si x = 2 Alors
      x ← x * 2
      Ecrire("x doublé : ", x)
   Sinon
      x ← x - 2
      Ecrire("x après soustraction : ", x)
   FinSi
FinSi
```

```
if (x > 2) {
        x = x + 3;
        printf("x après ajout : %d\n", x);
    } else if (x == 2) {
        x = x * 2;
        printf("x doublé : %d\n", x);
    } else {
        x = x - 2;
        printf("x après soustraction : %d\n", x);
    }
```

Les opérateurs logiques dans les conditions

&& : AND → toutes les conditions doivent être vraies;
|| : OR → au moins une condition doit être vraie;
! : NOT → inverse la condition (true devient false et vice versa).

Règle essentielle : Les conditions doivent être entre parenthèses;

Règle : le résultat de la condition doit être logique (vrai ou faux)
•En C, la condition d'un if **doit être une expression qui renvoie soit vrai (true), soit faux (false)**.

Le langage C est sensible à la casse (case-sensitive)
•Cela signifie que les majuscules et les minuscules sont considérées comme différentes.

Les mots-clés du langage doivent toujours être en minuscules (if, else, while, for, return, etc.).
Chaque instruction se termine par un point-virgule ;

Les commentairesPermettent d'expliquer le code sans l'exécuter.Commentaires sur une ligne : **// Ceci est un commentaire. /* Ceci est un commentaire sur plusieurs lignes */**

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 100 > =note ≥ 90
« Bien » si la note ≥ 75
« Passable » si la note ≥ 50
« Échec » sinon

1. Écrire un programme en langage ***C*** qui permet de résoudre l'équation suivante :

$ax^2 + bx + c = k$ avec a $\neq 0$;

2. Écrire un programme qui lit un entier n et affiche :

- "Pair et positif" si le nombre est pair ET supérieur à zéro
- "Pair mais négatif ou nul" si le nombre est pair mais ≤ 0;
- " Impair" sinon