# Programmation algorithmique

## Objectifs:

- concevoir un programme indépendamment du langage qui sera utilisé pour le coder (pérennité de la conception)

**Bases d'un langage algorithmique**

=> langage générique permettant de traiter des problèmes par concaténation d'instructions élémentaires.

## Organisation d'un programme algorithmique:

```
1: Nom du programme
2: Déclaration de variables
3: Déclaration de fonctions
4: Début
5:     ...
6:     Liste des instructions
7:     ...
8: Fin
```

## **Organisation d'un programme algorithmique**:

Les commentaires

Commentaire général (ex. : rôle d'un programme, ..):

{ceci est un commentaire général}

Commentaire sur un raffinage, le rôle d'une instruction, ..:

-- ceci est un commentaire spécifique

ou // ceci est un commentaire spécifique

# Déclaration des Variables et des types

Qu'est-ce qu'une variable?

•un espace mémoire (adresse)

• de taille fixe (nbre octets connu défini par le type)

•Pouvant prendre au cours de l'exécution de l'algorithme un nombre indéfini de valeurs différentes

Les types de variables:

•Entier: 5, 345, -87, … (le nombre d'octets dépend du processeur)

•Réel ou Flottant: 1.31415, 1.65 E+5,

•Booléen: VRAI ou FAUX

•Caractère: a, b, U,?, …

•chaîne: '' tension efficace= ''

Déclaration d'une variable:

En début du programme principal si elle est globale. En début du sous-programme (ou fonction) si elle est locale.

•Var a, b: entier

•Var Res: réel

# Variables et types

Affectation d'une valeur à une variable (la valeur est du même type que le type de la variable)

- a ← 123
- Res ← 1.75
- b← a

Cas d'une constante:

**Constante** PI: réel ← 3.1415

Tableaux de variables:

Ensemble de cases où chaque case contient une valeur

**VAR** tab : entier[100] est la définition d'un tableau nommé tab qui peut stocker 100 valeurs de type entier au maximum.

**VAR** z : réel[20]

# Les instructions

peuvent être:

- des opérations arithmétiques (sur variables)

Addition : +

Soustraction : -

Multiplication : *

Division : /

Modulo : %

- ….

- des affectations:

a ←b

# Les instructions

peuvent être:

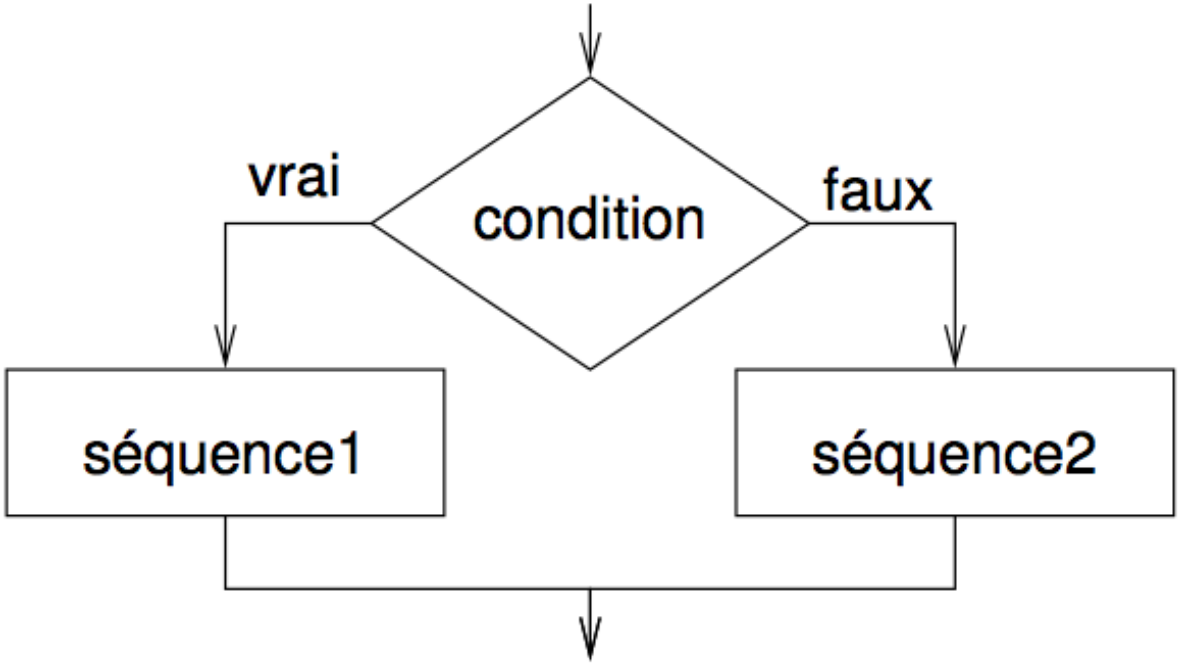
- des fonctions
- des phrases avec verbes d'action
- des tests sur des variables pour les structures de contrôle

| Nom | Utilisation | Rôle | Résultat |
|---|---|---|---|
| = | valeur1 = valeur2 | Égalité | VRAI si les deux valeurs testées sont égales |
| ≠ | valeur1≠ valeur2 | Inégalité | VRAI si les deux valeurs testées sont différentes |
| > | valeur1 > valeur2 | Supérieur strictement | VRAI si valeur1 strictement supérieure à valeur2 |
| < | valeur1 < valeur2 | Inférieur strictement | VRAI si valeur1 strictement inférieure à valeur2 |
| ⩾ | valeur1 ⩾ valeur2 | Supérieur ou égal | VRAI si valeur1 supérieure ou égale à valeur2 |
| ⩽ | valeur1 ⩽ valeur2 | Inférieur ou égal | VRAI si valeur1 inférieure ou égale à valeur2 |

# Les Structures de contrôle

Exécution conditionnelle d'instructions:
Structure **SI**…**ALORS**…**Sinon… FinSI**

**SI** condition **ALORS**
séquence 1
**Sinon**
séquence 2
**FinSI**

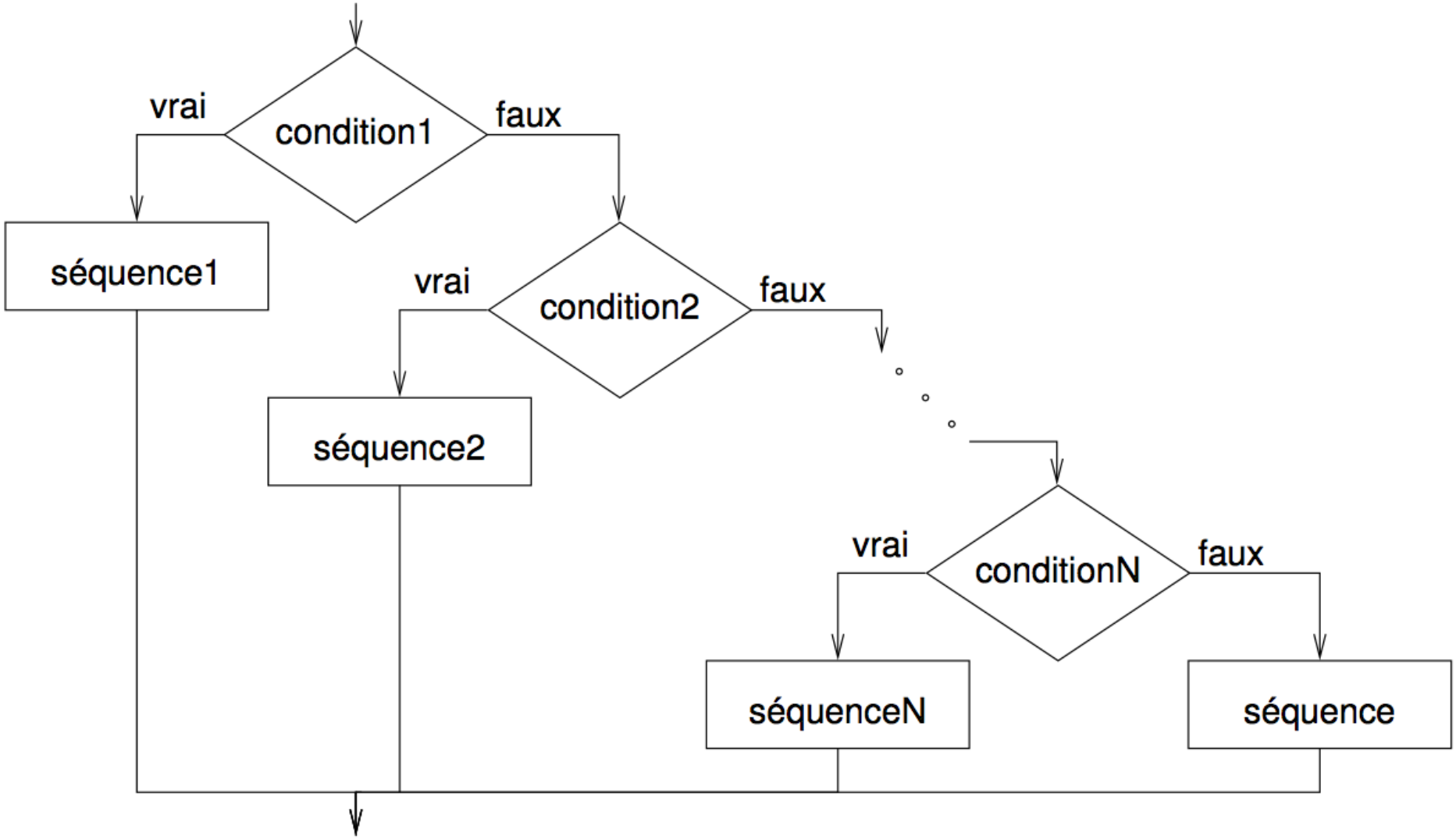


# Les Structures de contrôle

Exécution conditionnelle d'instructions: clause **SinonSi**

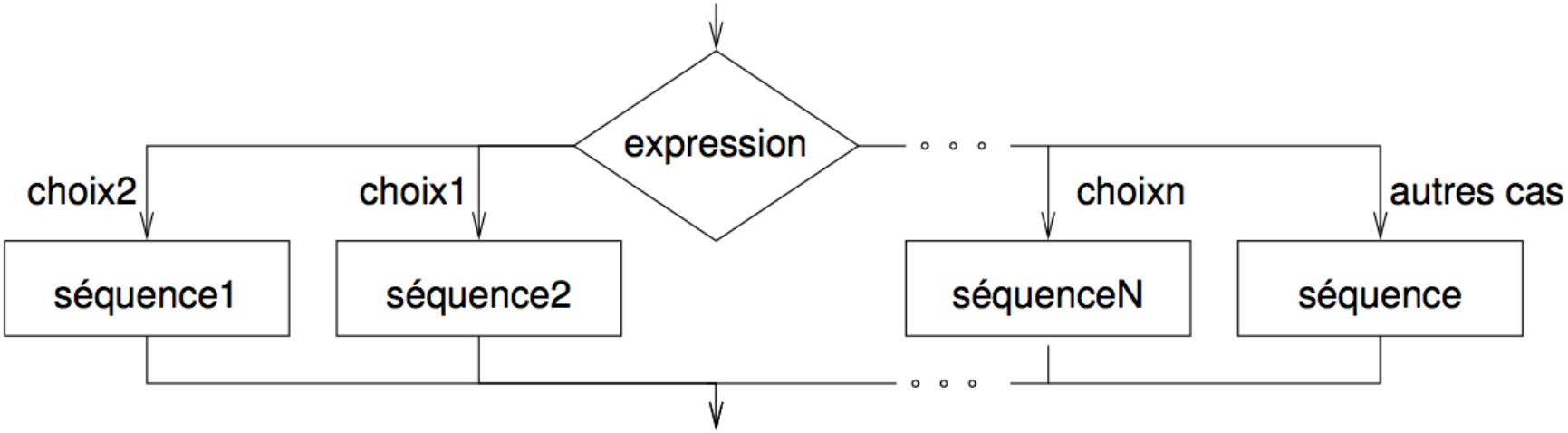
**SI** condition_1 **ALORS**
séquence 1
**SinonSi** condition_2
séquence 2
…
**SinonSi** condition_n
séquence n

**FinSI**


vrai
condition1
faux
séquence1
vrai
condition2
faux
séquence2
vrai
conditionN
faux
séquenceN
séquence

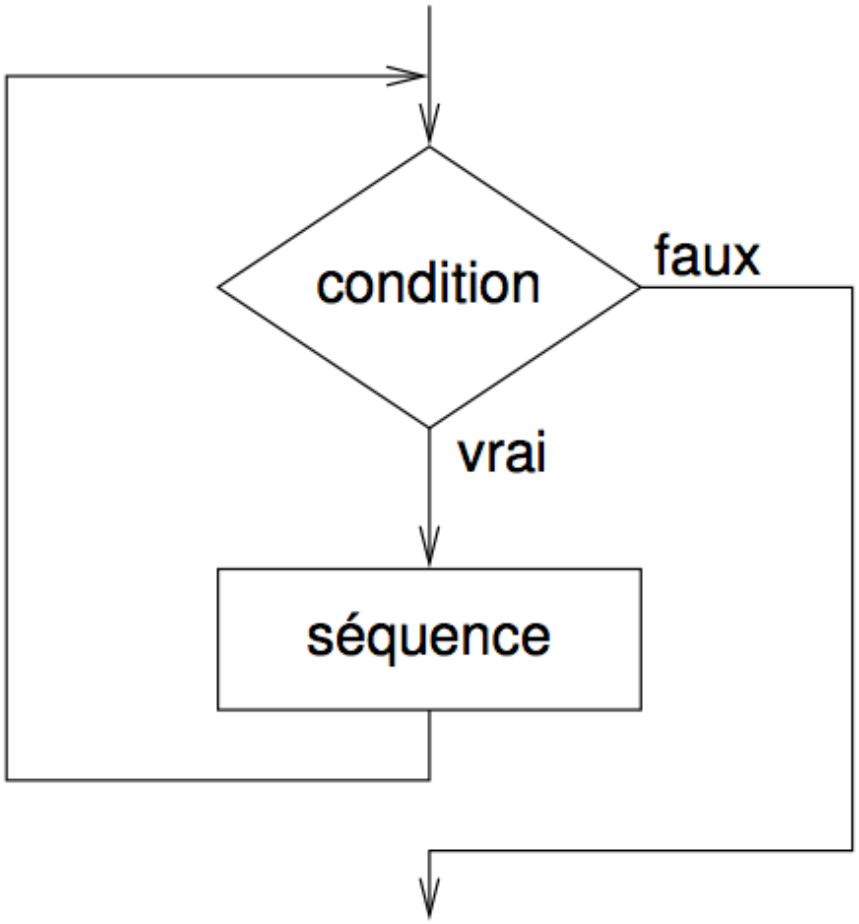
Structure Conditionnelle Selon:

Selon expression Dans
 choix $_1$ : séquence $_1$
 choix $_2$ : séquence $_2$ ...
choix $_N$ : séquence $_N$
Sinon séquence
FinSelon



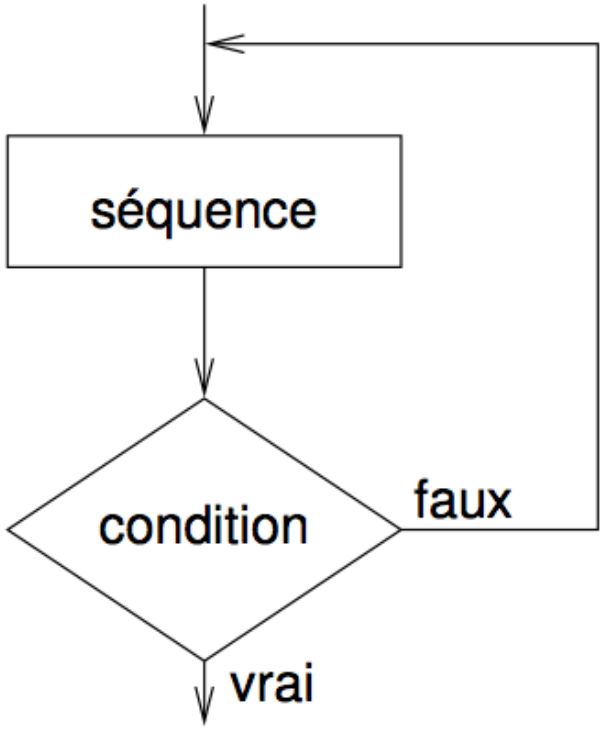
# Itérations et Boucles: Boucle TANTQUE

TANTQUE condition FAIRE
 instruction 1
...
instruction n
FinTQ



# La boucle FAIRE...TANT QUE

FAIRE
instruction 1
…
instruction n
TANTQUE condition



# La boucle POUR

**POUR** variable **DE** valeur1 **A** valeur2 **FAIRE**
instruction 1
…
instruction n
**FinPOUR**

```
PROGRAMME recherche_max
VAR maxi :entier              // stocke la valeur du maximum
VAR tabloval :entier [50]     // un tableau stockant des valeurs
VAR t_ut : entier             // la taille utile du tableau
VAR cpt : entier              // index des éléments du tableau
VAR rep : caractere           // pour la saisie des valeurs
// étape numéro 1 : initialisation des variables
t_ut ← 0  // à ce stade, pas de variable dans le tableau
FAIRE
  AFFICHER("entrez une valeur dans le tableau : ")
           // valeur rangée dans la case d'indice t_ut
  SAISIR(tabloval[t_ut])
           // incrémentation de t_ut (car ajout d'un élément)
  t_ut ← t_ut+1 ;
  AFFICHER("une autre saisie ? (o/n) :")
  SAISIR(rep);
  // la boucle reprend s'il reste de la place
  // dans le tableau ET si l'utilisateur souhaite continuer
TANTQUE ((t_ut < 50) ET (rep='o'))
  // pour l'instant, le plus grand est dans la case 0
maxi ← tabloval[0]
           // cherchons case par case (de l'indice 1 à t_ut-1)
POUR cpt DE 1 A t_ut-1 FAIRE
  // si l'on trouve plus grand :
  SI (tabloval[cpt] > maxi) ALORS
          // la valeur est mémorisée dans maxi
      maxi ← tabloval[cpt]
  FINSI
FAIT
```

# Les fonctions

➔ Ensemble d'instructions utilisé à différents endroits d'un programme

➔La structure d'une fonction ressemble à celle d'un programme si ce n'est qu'elle peut prendre un certain nombre de paramètres en entrée et qu'elle (doit) peut renvoyer une valeur.

```
type_retour NOM DE LA FONCTION(type_1 var_1, type_2 var_2, ...)
 1:     {Déclaration des variables locales}
 2: ...
 3: Début
 4:     ...
 5:     ...
 6:         {Liste des instructions}
 7:     ...
 8:     ...
 9:     Renvoyer     {une valeur de type type_retour}
10: Fin
```

# Les pointeurs

- Un pointeur est une « variable » qui contient une adresse d'une variable
- Permettent d'accéder à des adresses de variables
- permet à une fonction de passer/retourner plusieurs valeurs en passant par exemple l'adresse de début d'un tableau
- Var x: réel

⇒La notation &x signifie : « adresse de x ».

⇒La déclaration d'un pointeur nommé ptr_reel vers un réel est la suivante : **VAR** *ptr_reel : réel (se lisant : « le contenu de ptr_reel est de type réel »)

⇒La déclaration d'un pointeur nommé ptr_int vers un entier est la suivante : **VAR** *ptr_int : entier