UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Informatique
Modélisation UML

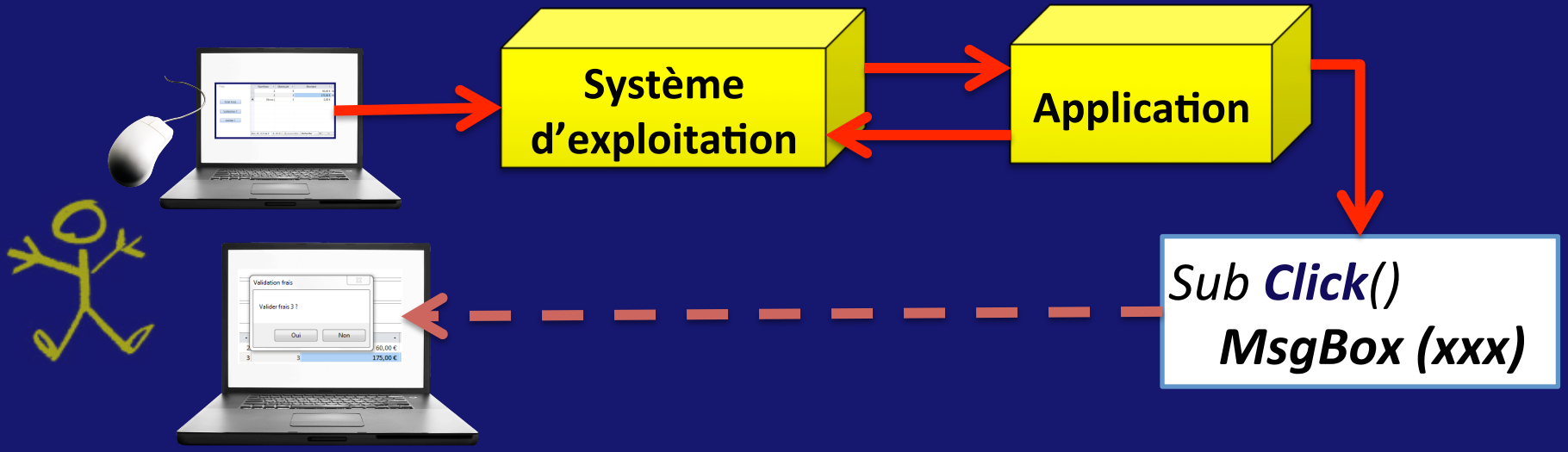
Objectifs de la séance :

Introduction au fonctionnement des applications informatiques

Architecture des applications

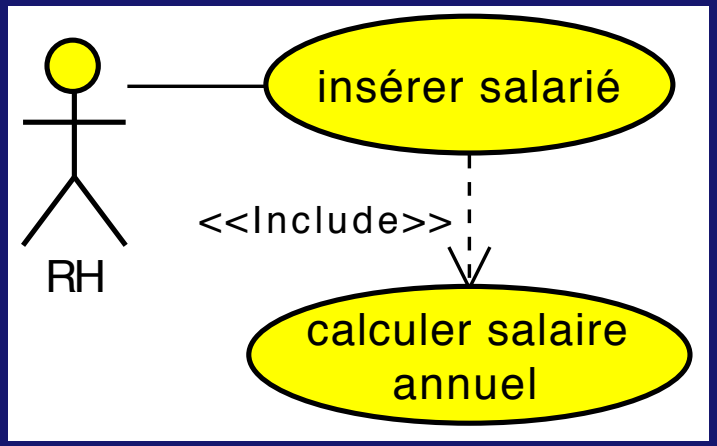
Fonctionnement d'une application

- Comment fonctionne une application informatique ?
  - un ordinateur ne sait rien faire tout seul, il faut lui dire exactement quoi et comment faire
  - un ordinateur suit des ordres (des **instructions**), qui sont codés dans les **applications** informatiques



Fonctionnement d'une application

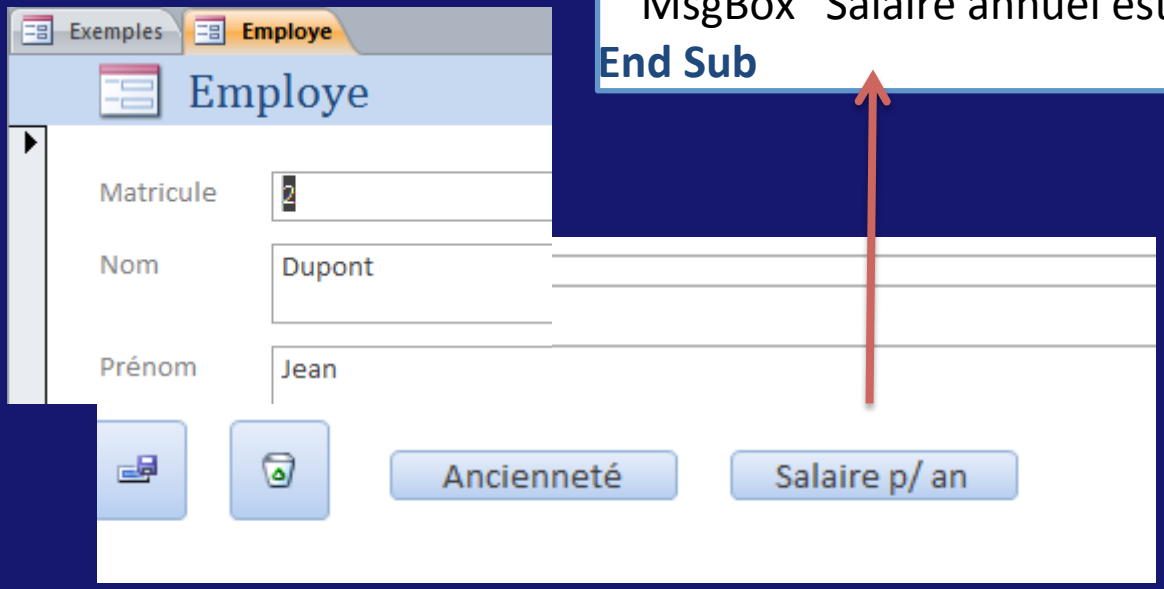
- Les applications informatique mettent en œuvre
  - les fonctionnalités identifiées dans les cas d'utilisation
  - les processus représentés par les diagrammes d'activités
- Exemple :
  - lorsque le RH insère un nouvel employé, il doit renseigner ses informations et calculer le salaire annuel

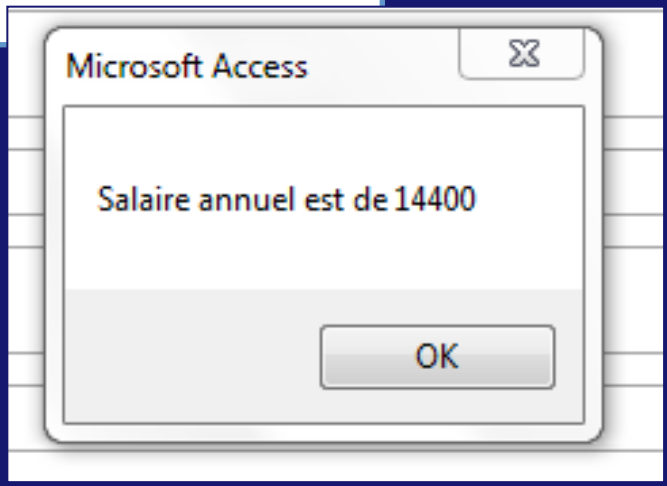


Fonctionnement d'une application

Passage ?!

```
Private Sub Annuel_Click()
    Salaire = Forms("Employe").Controls("Salaire")
    Annuel = Salaire * 12
    MsgBox "Salaire annuel est de " & Annuel
End Sub
```





Fonctionnement d'une application

- Pour comprendre comment fonctionne les applications, il faut comprendre
  - comment elles s'organisent ?
    - Architecture
  - comment réaliser / organiser les instructions ?
    - Algorithme

Architecture logicielle

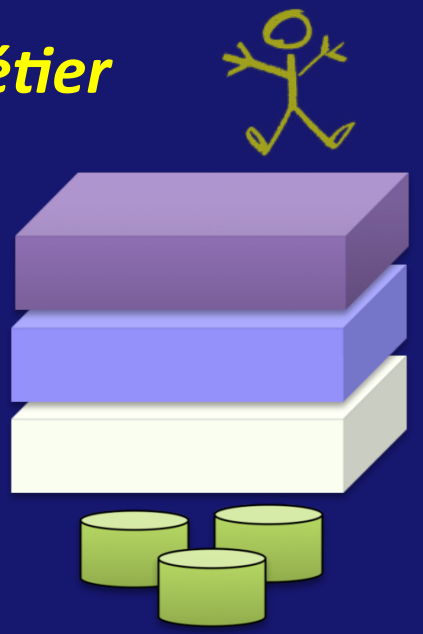
- Architecture logicielle
  - L'organisation interne d'une application informatique est déterminée par son architecture logicielle
  - L'architecture logicielle correspond à la description de la structure générale d'une application
    - différents éléments composant l'application et leur interrelations
    - représentation abstraite d'un système, de son organisation générale et de sa composition en sous-systèmes
  - Définie lors de la phase de conception

Architecture logicielle

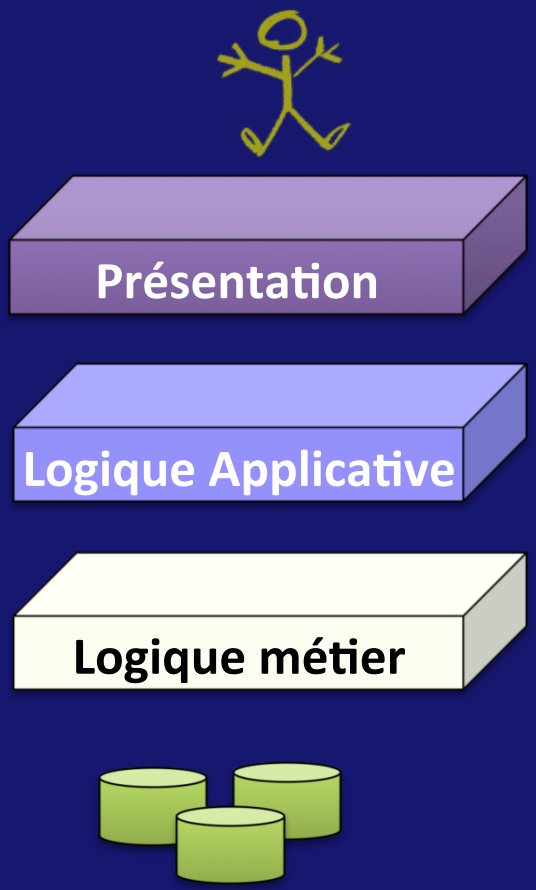
- Architecture logiciel décrit comment on organise un logiciel
  - permettre une identification claire de ses composants et de leur responsabilités
- Une architecture bien réfléchie permet
  - construction d'applications plus faciles à maintenir et à faire évoluer
  - réduction des coûts de maintenance
  - augmentation de la qualité de l'application
- Différents styles architecturaux sont possibles
  - **Architecture multicouches**

Architecture logicielle

- **Architecture multicouche (*n-tier*)**
  - Découplage en plusieurs couches, chacune communiquant uniquement avec leurs voisines directes
- Découplage traditionnel en **3 couches**, correspondant à une certaine catégorisation des composants
  - ***Présentation***, logique ***applicative***, logique ***métier***
  - L'objectif est de **minimiser l'impact** d'un **changement** à l'intérieur d'une couche sur les autres couches
    - L'***interface utilisateur*** doit pouvoir ***évoluer*** tandis que le cœur de l'application reste inchangé
    - On doit pouvoir utiliser les ***bases de données*** pour ***plusieurs applications***



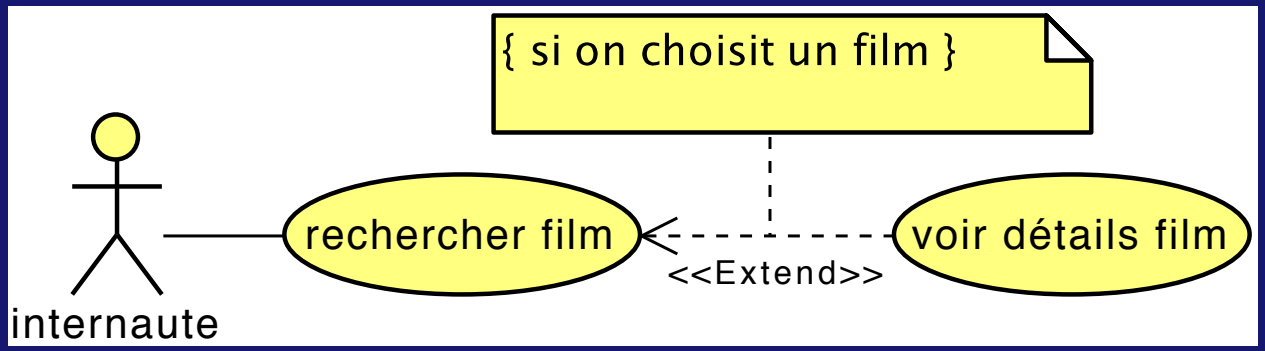
Architecture logicielle



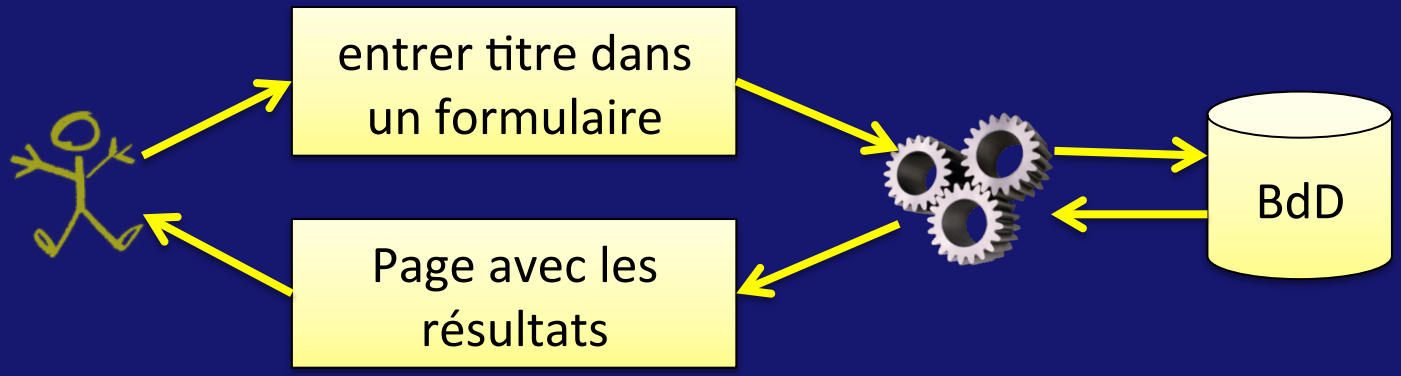
- **Présentation**
  - éléments gérant l'interaction avec l'utilisateur
  - interface utilisateur
  - « dialogues » (« *boundary* »)
- **Logique applicative**
  - éléments gérant les règles applicatives
  - cœur d'une application
  - « contrôle » (« *control* »)
- **Logique métier**
  - éléments gérant l'accès aux données
  - concepts métiers et leurs relations
  - « entités » (« *entities* »).

Architecture logicielle

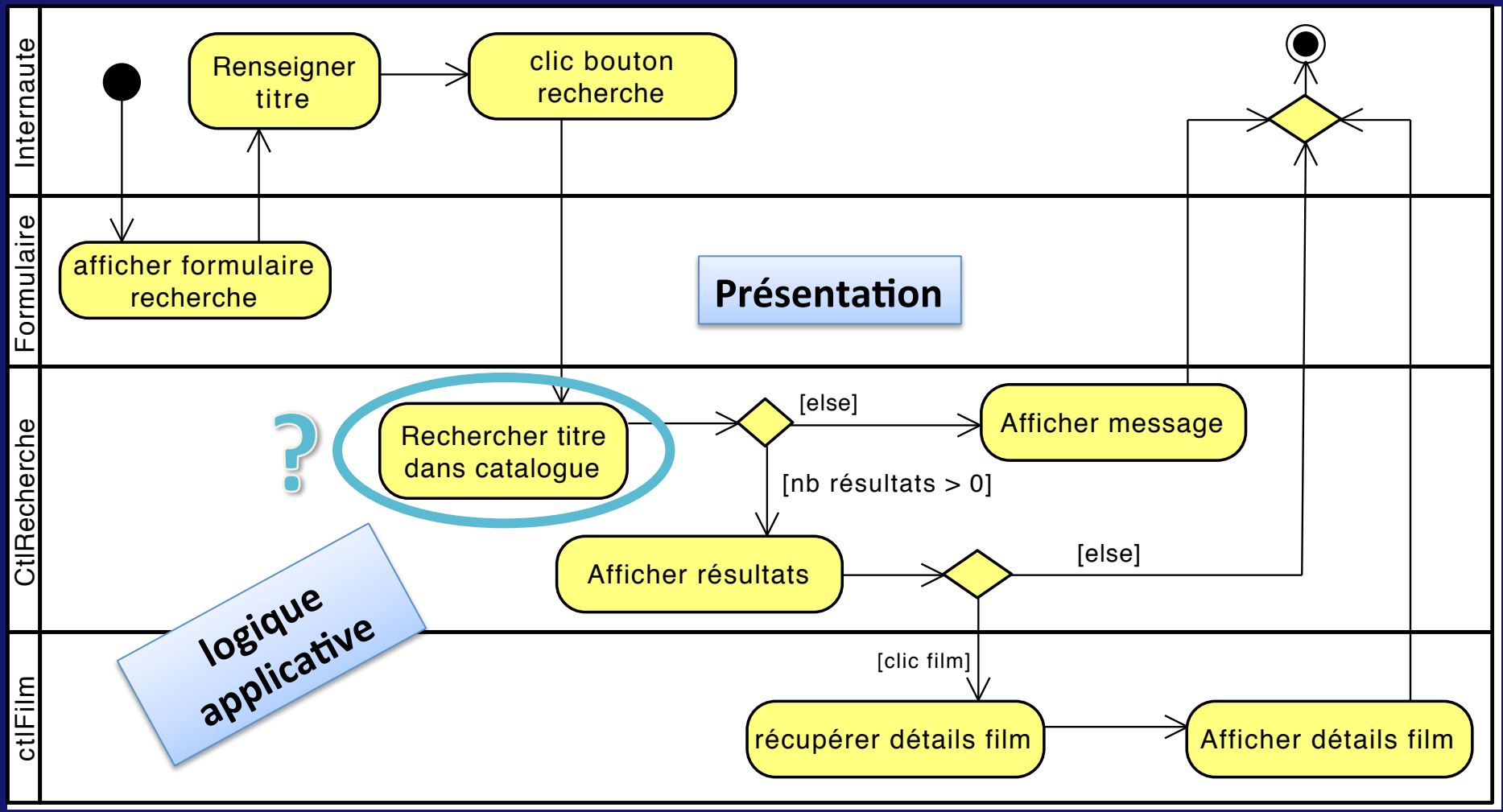
- Exemple :
  - recherche d'un film dans le site d'une « dvd-tèque »



Découpage en plusieurs parties (**couches**)



Architecture logicielle



Algorithme

- La mise en place d'une application (ou d'une activité) demande une séquence d'instructions précises que l'ordinateur devra suivre
  - C'est l'algorithme
- Un algorithme…
  - est une séquence finie de pas à effectuer dans un certain ordre afin de parvenir à un résultat
  - décrit, de façon non ambiguë, l'ordonnancement des actions à effectuer pour traiter une fonctionnalité
- Un algorithme est donc un processus

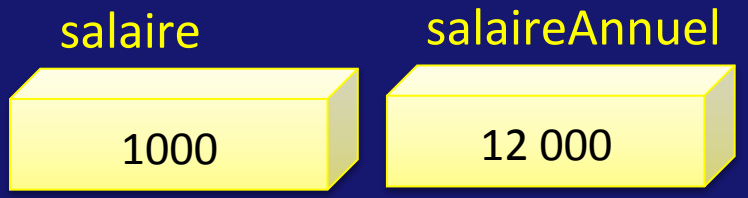
Algorithme

- Différents éléments d'un algorithme
  - les données (les variables)
  - les expressions (les actions simples)
  - les activités (les procédures ou fonctions)
  - les structures de contrôle conditionnelles (les décisions)
  - les structures de contrôle itératives (les boucles)
- On implémente un algorithme à l'aide d'un langage de programmation
  - l'algorithme est la « recette » d'une application
  - un même algorithme peut être implémenté avec différentes langages de programmation

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Algorithme

- Les données (Variables)
  - les algorithmes doivent souvent manipuler des données
  - ces données doivent être gardées quelque part pour être manipulées par
    → variables
- Variables
  - une variable est un conteneur
  - on y garde une valeur qu'on pourra utiliser plus tard

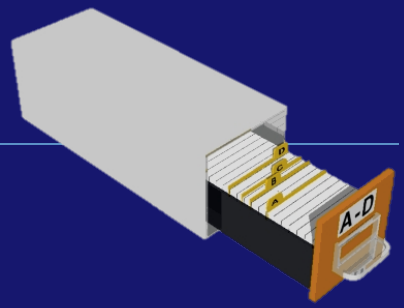


affectation
on attribue une valeur à une variable

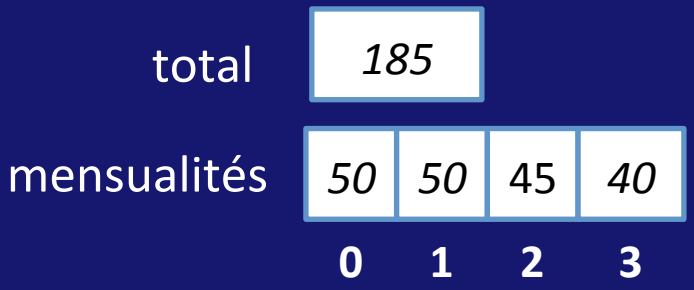
```
salaire = 1000
salaireAnnuel = salaire * 12
```

expression
on utilise la valeur stockée dans une variable

Algorithme



- Tableaux
  - les tableaux sont des variables capables de garder plusieurs valeurs
  - chaque valeur est gardée dans une position précise
    - première position est souvent 0



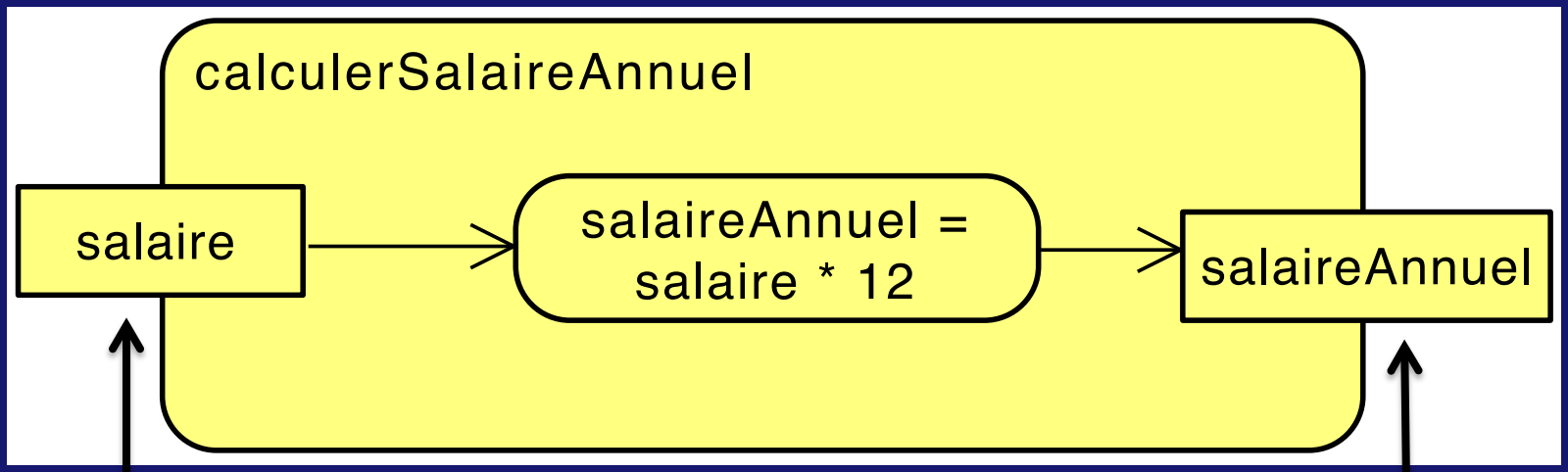
mensualités **[0]** = 50
mensualités **[2]** = 45

la variable ***mensualités*** contient **plusieurs valeurs**

chaque valeur est **accessible** par sa **position** (0, 1, 2 …)

Algorithme

- Un algorithme va souvent avoir une (ou plusieurs) entrée(s) et une (ou plusieurs) sortie(s)

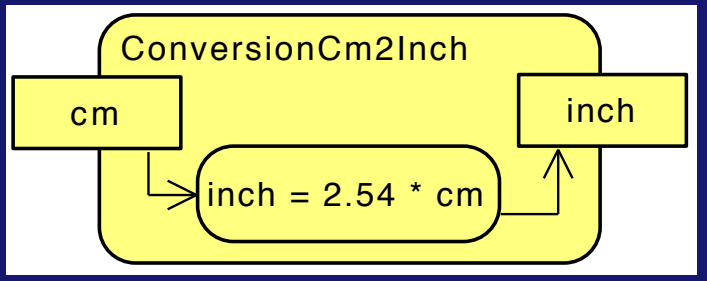


- on peut assimiler un algorithme à une activité

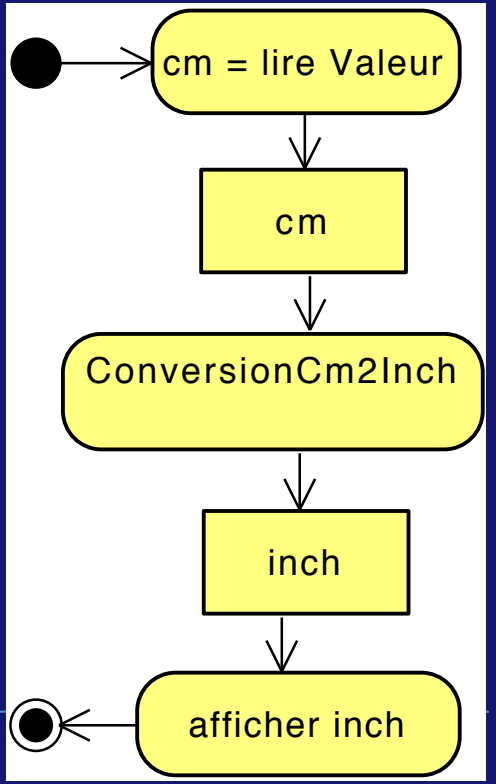
UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Algorithme

- Comme les processus, les algorithmes peuvent s'organiser en sous-processus
  - diviser en petites parties pour réduire la complexité

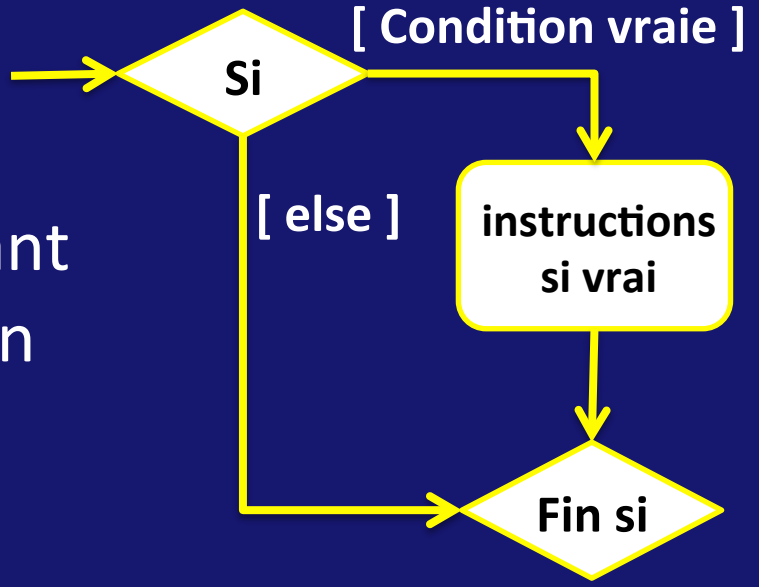


- Procédure ou fonctions
  - Une procédure (ou fonction) correspond à une activité, à un ensemble d'actions (instructions) invocables à partir d'une autre activité
  - Il peut y avoir des paramètres d'entrées (valeurs dont elle a besoin) et produire d'autres valeurs (sa sortie)
    - une procédure sans aucune sortie est parfois appelée une « subroutine » (« sub »)



Algorithme

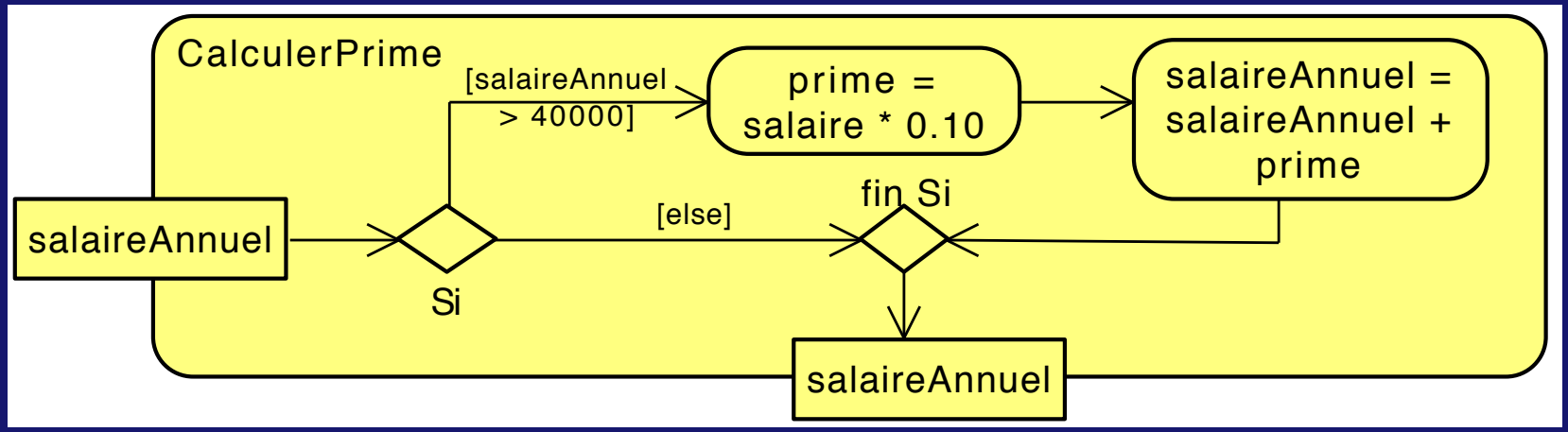
- Structures conditionnelles (décisions)
  - structures de contrôle permettant de choisir quel flot prendre selon une *condition* (test)
  - le flot d'instructions à suivre est conditionné par le résultat d'un test
    - On n'emprunt un certain flot que si la condition est vraie
  - structure de type « Si *condition* alors … »



Si *condition* **Alors**
'si condition vraie …
Fin Si

Algorithme

- Structures conditionnelles (décisions)
  - Exemple : on attribue une prime de 10% aux employés dont le salaire annuel est supérieur à 40K€

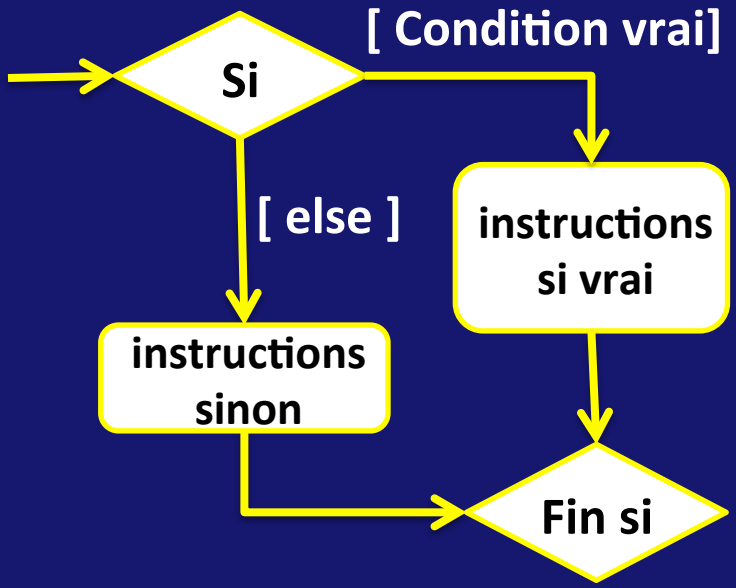


Si *salaireAnnuel > 40 000* **Alors**
prime = salaireAnnuel * 0.10
salaireAnnuel = salaireAnnuel + prime
Fin Si

salaireAnnuel	*45 000*
prime	*4 500*
salaireAnnuel	***49 500***

Algorithme

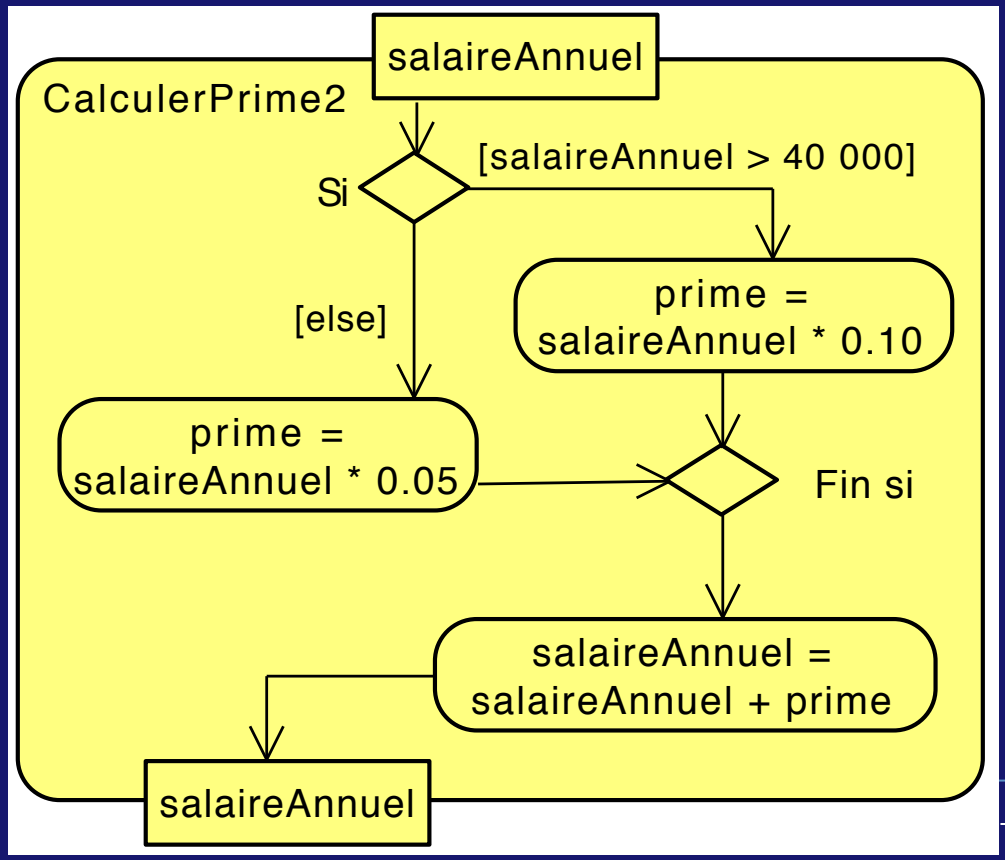
- Structures conditionnelles (décisions)
  - on peut organiser plusieurs flots possibles



Si *condition* **Alors**
'si condition vraie …
Sinon
'si condition fausse
Fin Si

Algorithme

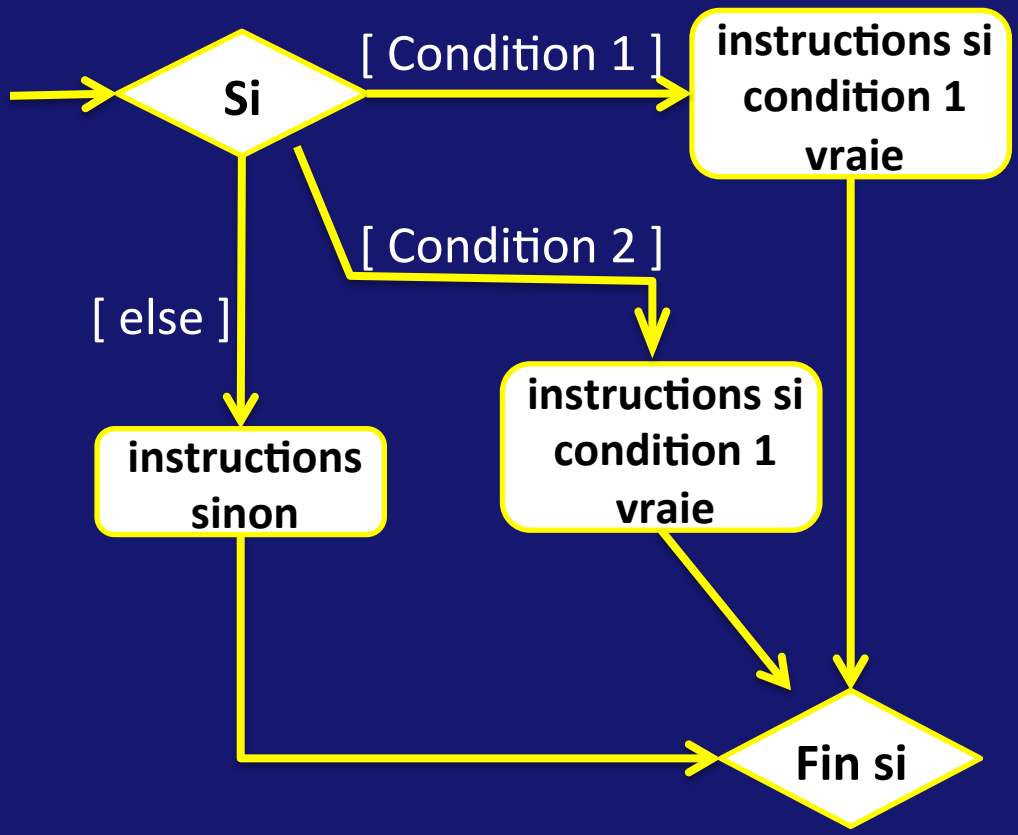
- Structures conditionnelles (décisions)
  - Exemple : on attribue une *prime de 10% aux* employés dont le salaire annuel est *supérieur à 40K€* et une *prime de 5%* pour les autres.



Si *salaireAnnuel > 40 000* **Alors**
prime = salaireAnnuel * 0.10
Sinon
prime = salaireAnnuel * 0.05
Fin Si
salaireAnnuel = salaireAnnuel + prime

Algorithme

- Structures conditionnelles (décisions)
  - On peut même cumuler plusieurs conditions



Si *condition1* **Alors**
'si condition1 vraie …
Sinon Si *condition2* **Alors**
'si condition2 vraie …
Sinon
'si ttes conditions fausses
Fin Si

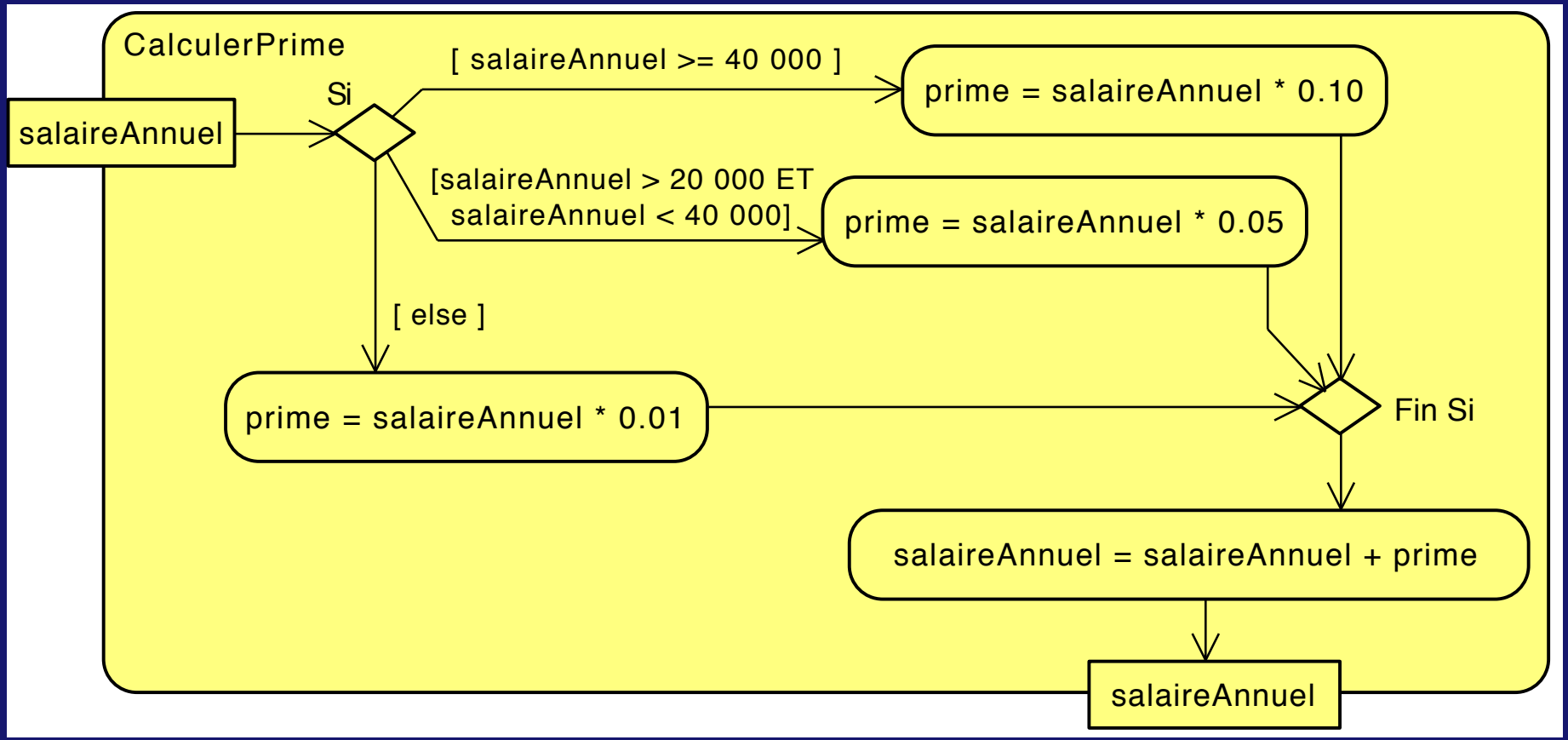
Algorithme

- Exercice
  - Ecrire, à l'aide des diagrammes d'activités, un algorithme permettant d'attribuer à un employé une *prime de 10%* si son le *salaire annuel est supérieur* à *40K€*, une *prime de 5%* si son *salaire annuel est entre 20K€ et 40K€*, et de *1% pour tous les autres*.

- Comment s'y prendre ??
  - bien identifier les entrées et les sorties
  - bien identifier les actions à réaliser
  - identifier les décisions et leurs conditions
  - organiser le(s) flot(s)

Algorithme

- Exercice



Algorithme

- Exercice

  Entrée : salaireAnnuel

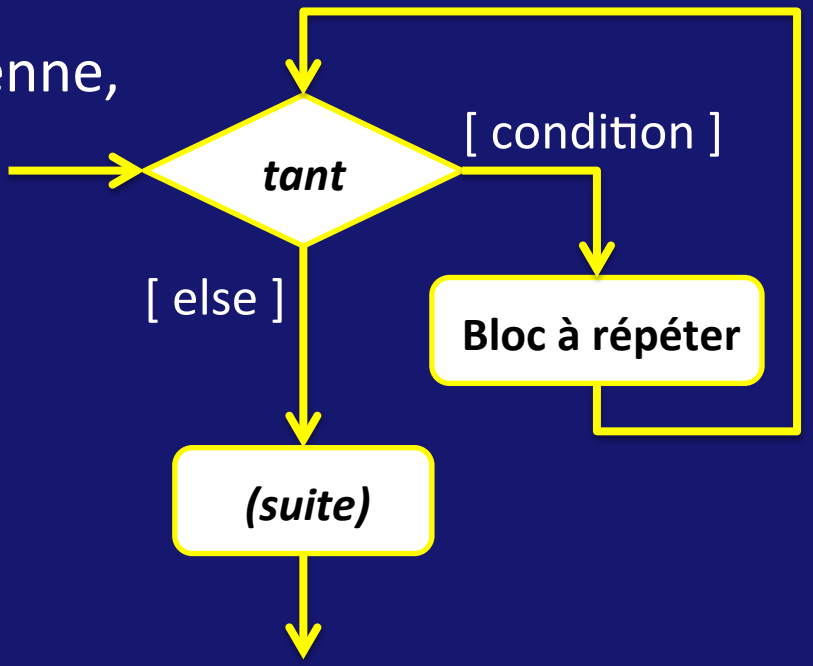
  Sortie : salaireAnnuel *(mis à jour)*

  Si *salaireAnnuel > 40 000* Alors
  prime = salaireAnnuel * 0.10
  Sinon si *salaireAnnuel > 20 000* Alors
  prime = salaireAnnuel * 0.05
  Sinon
  prime = salaireAnnuel * 0.01
  Fin Si

  salaireAnnuel = salaireAnnuel + prime

Algorithme

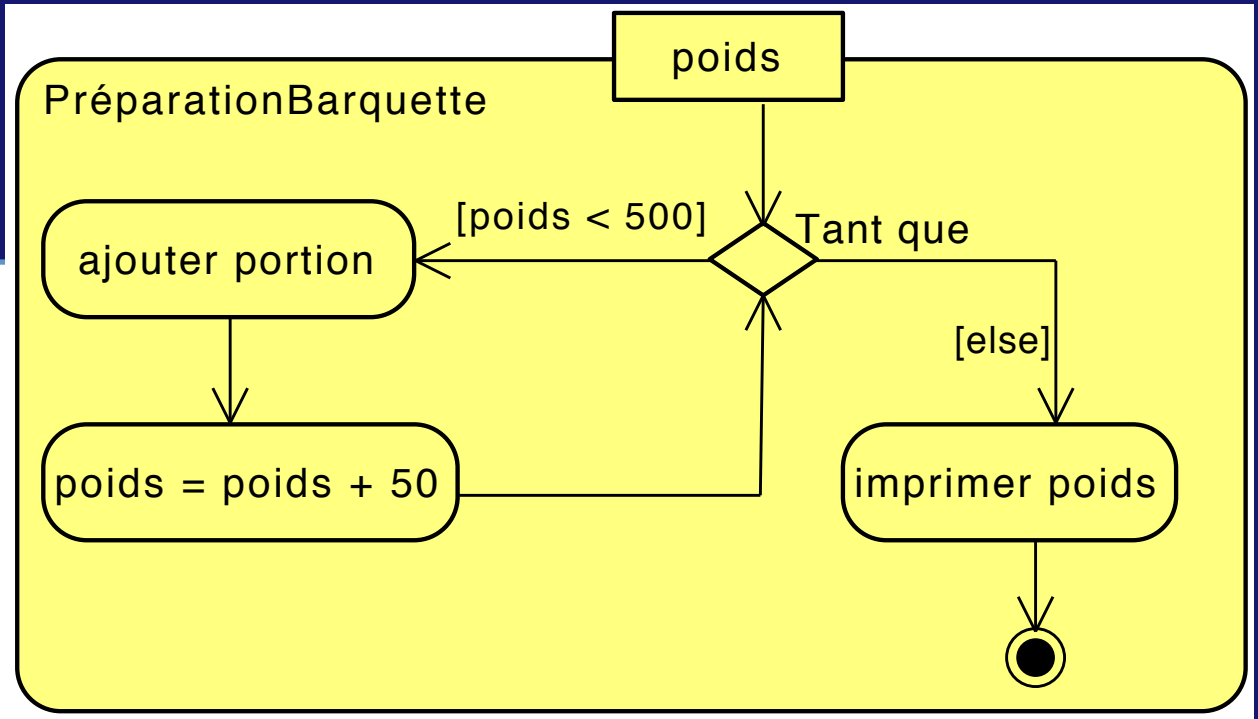
- Structures itératives (boucles)
  - parfois il faut répéter un même bloc d'instructions
  - on peut répéter un bloc un nombre fixe de fois ou tant qu'une condition soit vraie
    - Exemple : calculer une moyenne, un exponentiel…



Tant que *condition*
Faire
' instructions à répéter …
Fin faire
suite …

Algorithme

- Structures itératives (boucles)
  - Exemple : on prépare une barquette de 500g de viande hachée. Tant que la barquette n'a pas atteint ce poids, on ajoute une portion de 50g.



Tant que *poids < 500*
Faire
ajouter portion
poids = poids + 50
Fin faire
imprimer poids

Algorithme

- Structures itératives (boucles)
  - Exercice : pour calculer x^y, il faut faire *y* fois *x * x*
    - *y = 0* → *résultat = 1*
    - *y = 1* → *résultat = 1 * x*
    - *y = 2* → *résultat = (1 * x) * x*
    - *y = 3* → *résultat = ((1 * x) * x) * x*

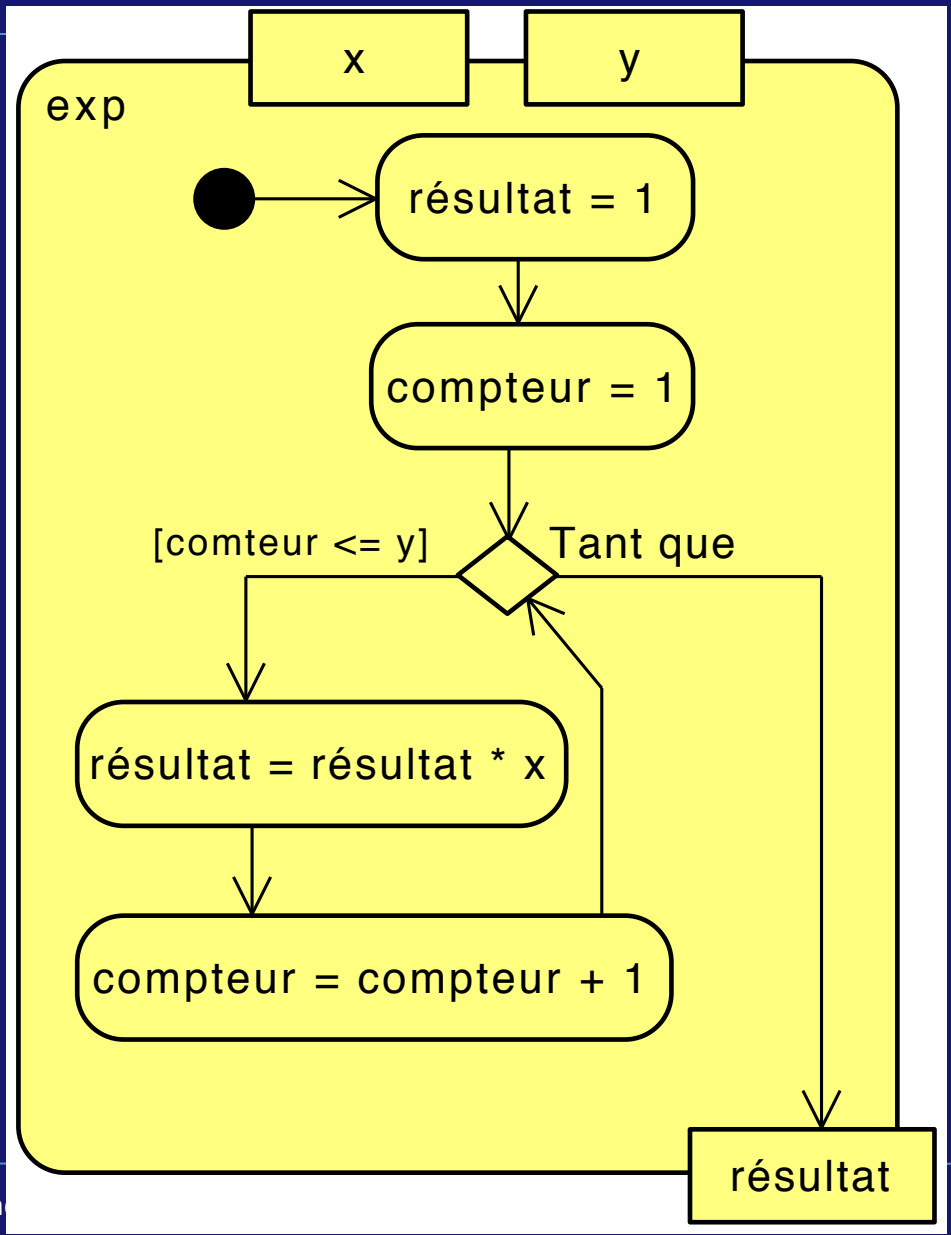
    . . .
  - on doit alors
    - garder la valeur précédente (une variable)
    - compter combien de fois on fait x * x et s'arrêter à y
  - Entrée ? *x et y*
  - Sortie ? *résultat*

à chaque tour, on prend la *valeur précédente* et on la multiplie par *x*

Algorithme

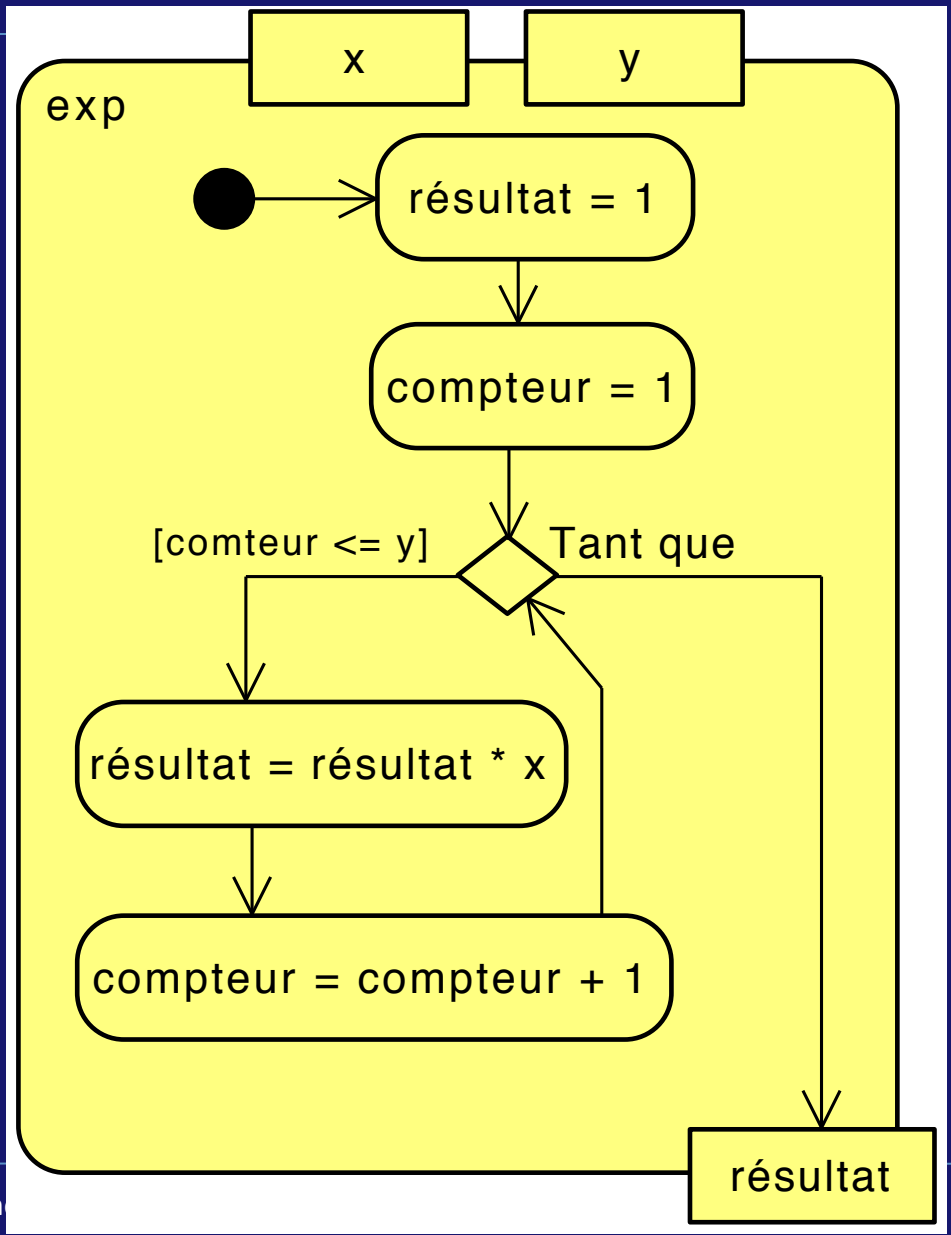
- Structures itératives (boucles)

Entrée : *x, y*
Sortie : *résultat*
résultat = 1
compteur = 1
Tant que *compteur <= y*
Faire
*résultat = résultat * x*
compteur = compteur + 1
Fin faire



Algorithme

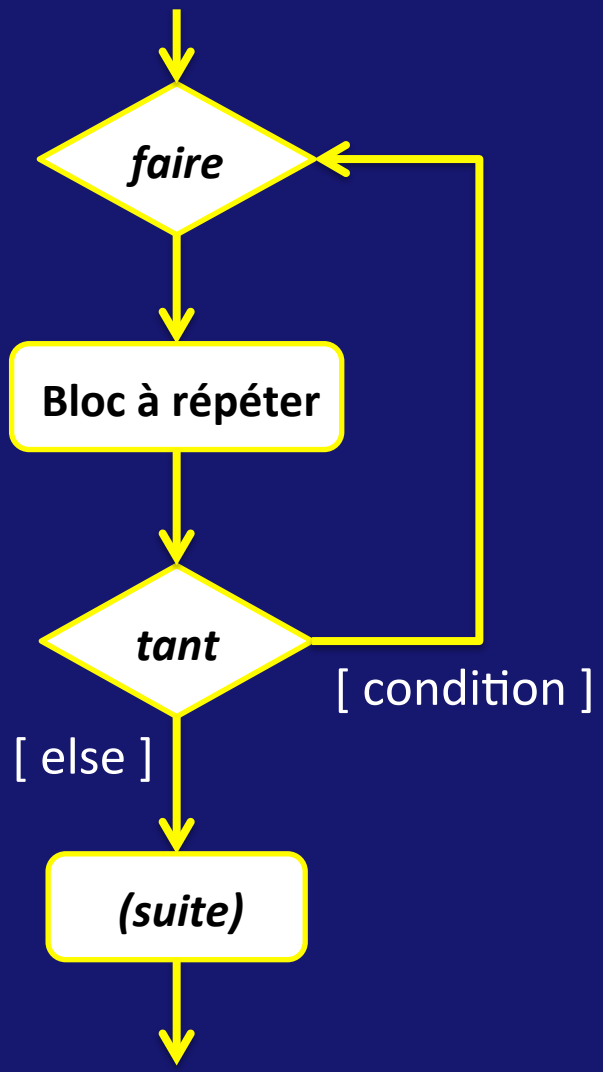
- Structures itératives (boucles)
  - Dans les cas précédents, on vérifie la condition *avant* d'entrer dans la boucle
    - on peut donc ne jamais entrer dans le bloc d'instructions si la condition ne se réalise jamais
    - exemple : si *y = 0* (par ex. 2^0)



Algorithme

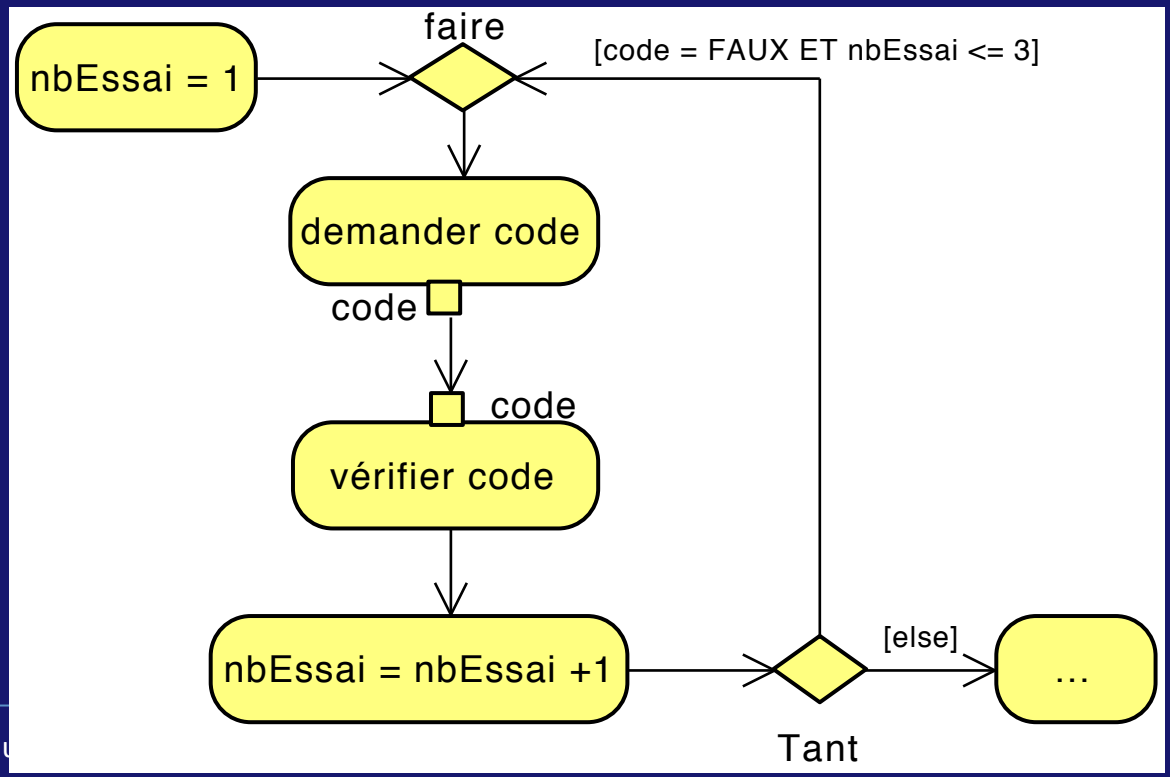
- Structures itératives (boucles)
  - On peut également vérifier la condition *après* avoir exécuté le bloc d'instructions
    - on réalise le bloc au moins une fois

Faire
' instructions à répéter …
Tant que *condition*
suite …



Algorithme

- Structures itératives (boucles)
  - On demande le code au client tant que le code soit faux et le nombre d'essais <= 3
  - Le client entre le code (demander code) au moins une fois



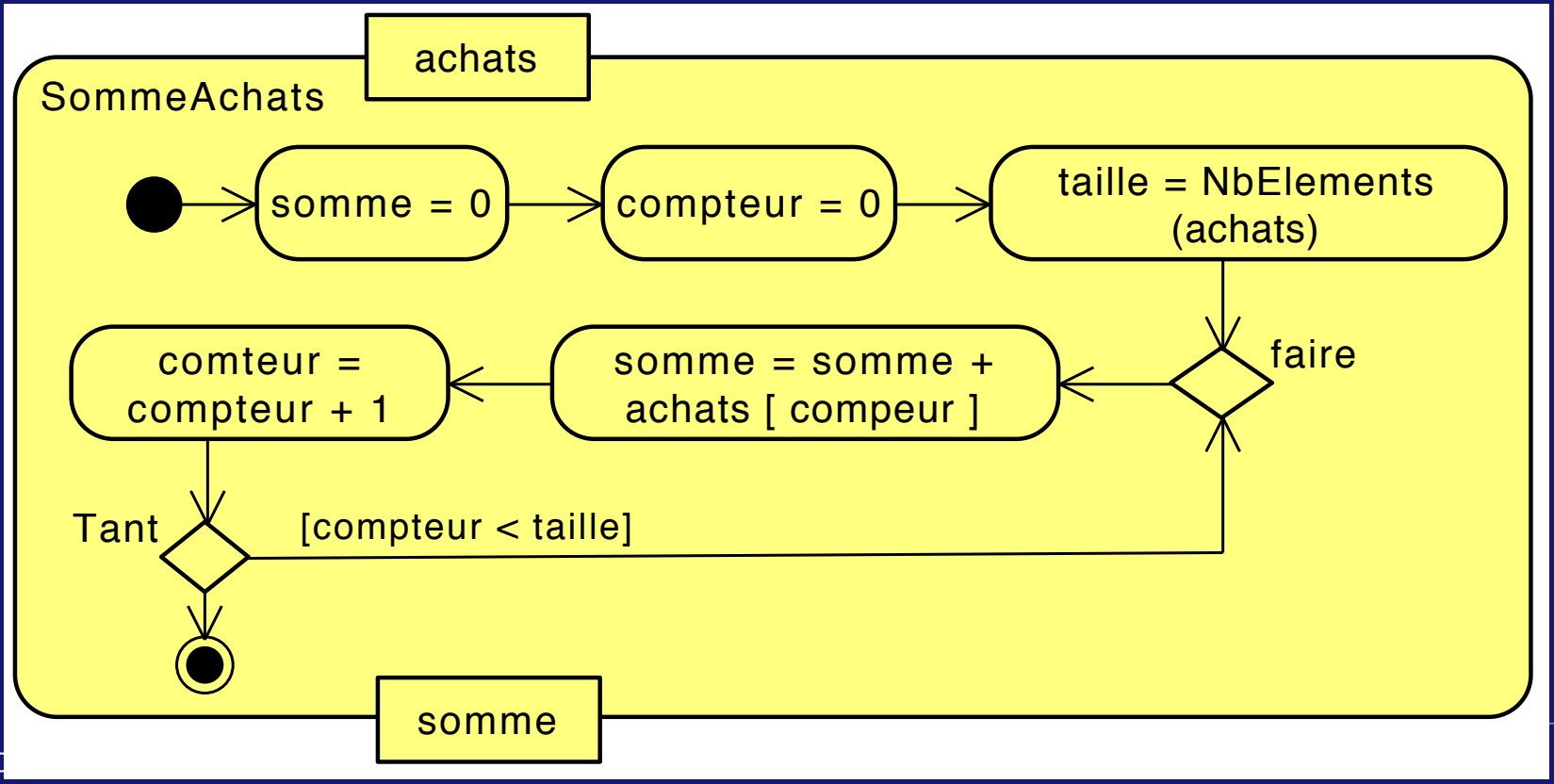
Algorithme

- Structures itératives (boucles)
  - on peut également répéter un bloc d'instructions pour chaque élément d'un tableau
    - connaître la taille du tableau
  - Exemple : faire la somme d'une liste d'achats

Entrée : *achats [ ]*
Sortie : *somme*
somme = 0
compteur = 0
taille = NbElements (achats)
Faire
somme = somme + achat [compteur]
compteur = compteur + 1
Tant que *compteur < taille*

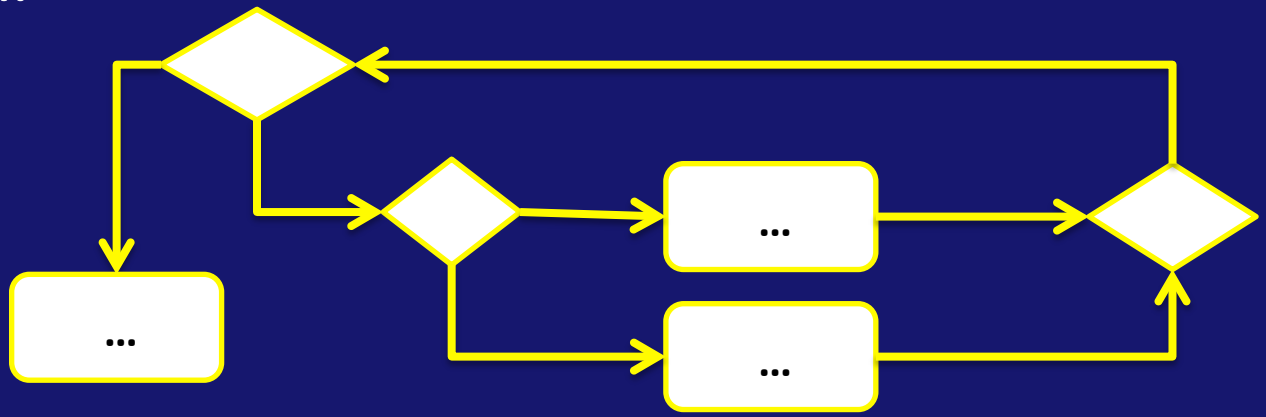
Algorithme

- Structures itératives (boucles)
  - on peut également répéter un bloc d'instructions pour chaque élément d'un tableau



Algorithme

- On peut imbriquer plusieurs blocs d'instructions
  - un test à l'intérieur d'un autre test,
  - une boucle à l'intérieur d'une autre boucle,
  - une boucle à l'intérieur d'un test,
  - un test à l'intérieur d'une boucle
  - …



Algorithme

- Exemple :
  - Si la température est supérieur à 37, on a de la fièvre. Au-delà de 40, notre état est d'hyperthermie.
  - Si la température est inférieur à 35, on est en hypothermie.
  - En dehors de ces cas, on est dans un état normal.

Faire un algorithme qui, à partir d'une température donnée, identifie notre état.

Algorithme

- Exemple :

  Faire un algorithme qui, à partir d'une température donnée, identifie notre état.

  – Entrée ? *température*

  – Sortie ? *état*

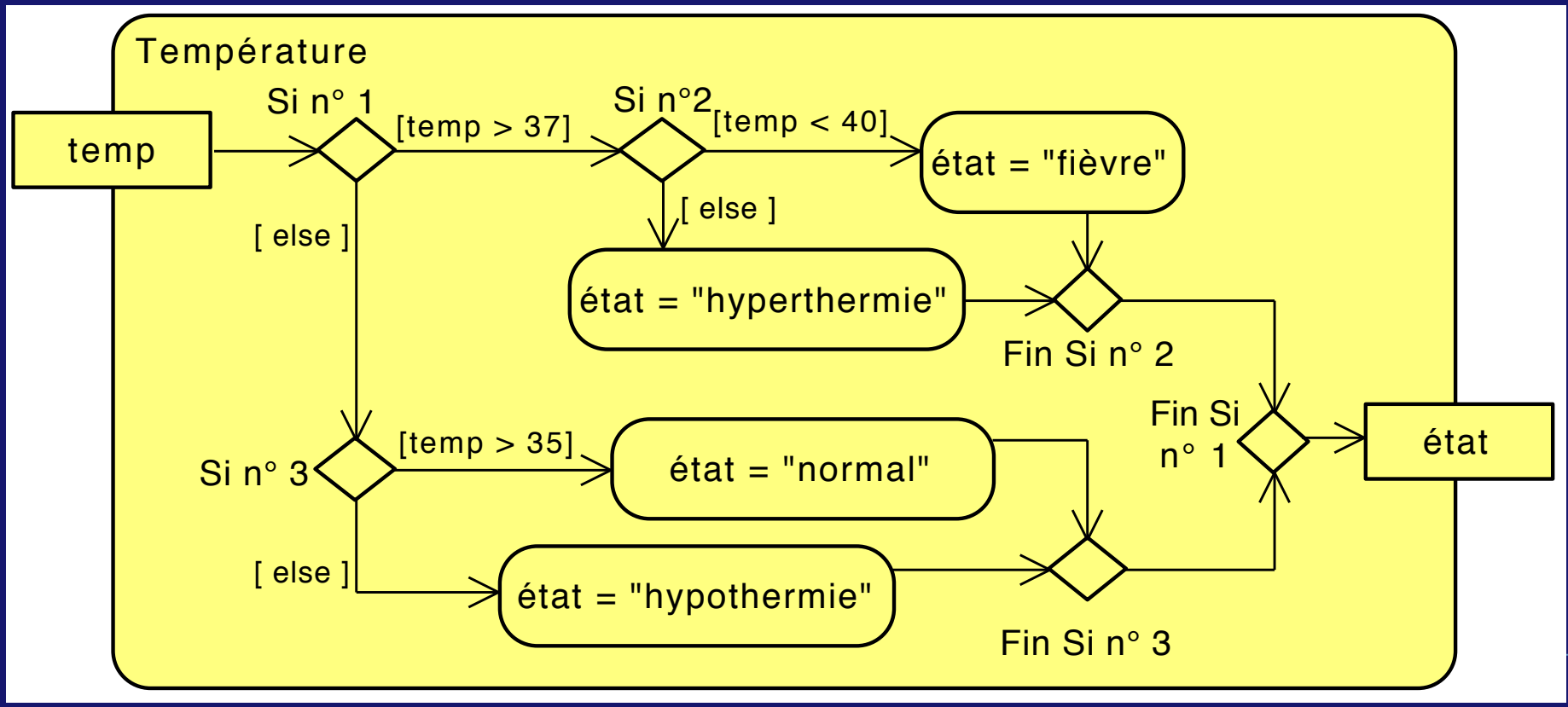
  – Décisions ?

    - *37 ≤ température < 40 → état = fièvre*
    - *température ≥ 40 → état = hyperthermie*
    - *35 ≤ température < 37 → état = normal*
    - *température ≤ 35 → état = hypothermie*

Algorithme

- Exemple :

Faire un algorithme qui, à partir d'une température donnée, identifie notre état.



Algorithme

- Exemple :

  Faire un algorithme qui, à partir d'une température donnée, identifie notre état.

```
Entrée : temp
Sortie : état

Si temp > 37   Alors
      Si temp < 40 Alors
            état = " fièvre "
      Sinon
            état = " hyperthermie "
      Fin Si
Sinon
      Si temp > 35 Alors
            état = " normal "
      Sinon
            état = " hypothermie "
      Fin Si
Fin Si
```

Algorithme

- Exemple :

Afficher la table de multiplication

- *pour chaque ligne i , nous avons 9 colonnes (j de 1 à 9)*

Les lignes (***i***)
varient de 1 à 9

Les colonnes (***j***)
varient de 1 à 9

	1	2	3	4	. . .
1	1	2	3	4	. . .
2	2	4	6	8	. . .
3	3	6	9	12	. . .
4	4	8	12	16	. . .
. . .	. . .	. . .	. . .	. . .	. . .

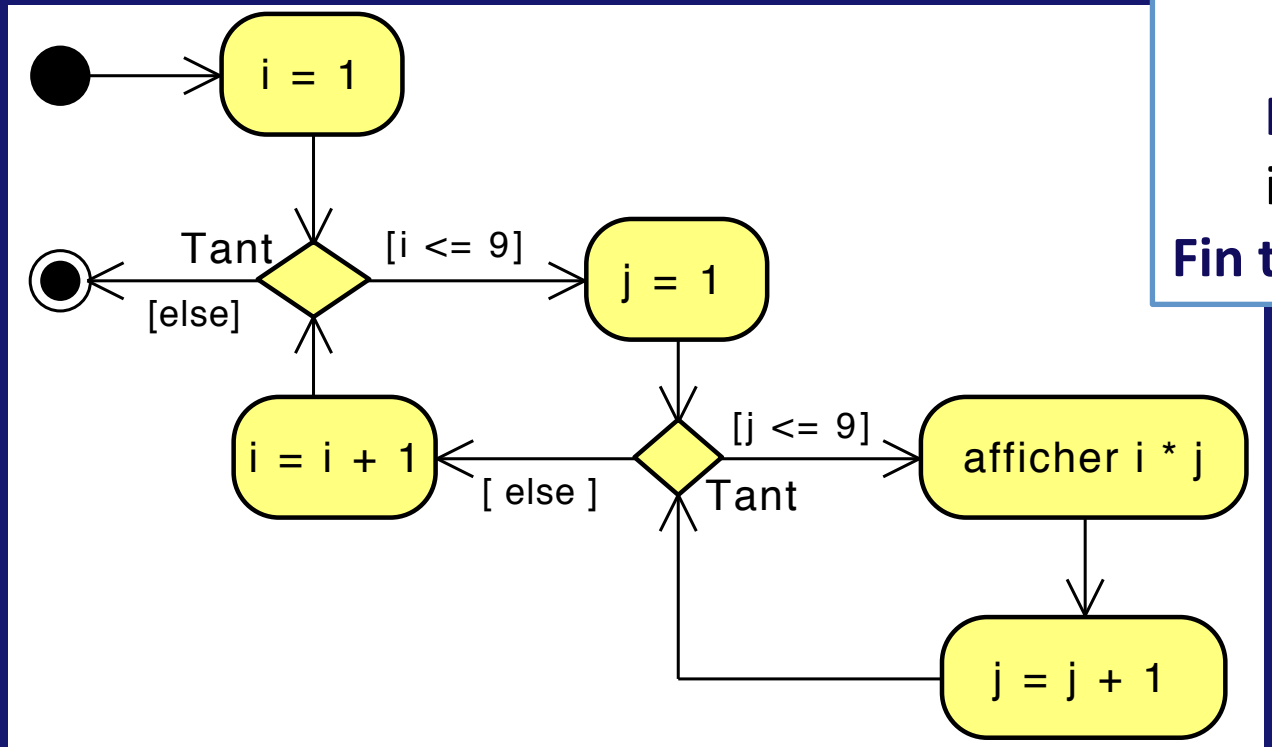
chaque position (***i , j***) contient la valeur de *i * j*

Algorithme

- Exemple :

  Afficher la table de multiplication

```
i = 1
Tant que i <= 9   Faire
      j = 1
      Tant que  j <= 9  Faire
            afficher i * j
            j = j  + 1
      Fin tant
      i = i + 1
Fin tant
```



*pour **chaque valeur** de **i**, on parcourt **tous les valeurs de j***