

ALGORITHMIQUE

↔ Conditions ↔

I. Structure conditionnelle simple

🗨 Définition

On utilise une telle **structure** dès que l'on doit effectuer une action `si` une condition est réalisée.

📄 Algorithme

En `pseudo-code`, on écrit :

```
Si <condition> Alors
  <action>      # réalisée si condition Vraie
FinSi
```

📄 Pseudo-Code

👉 Remarque

La `<condition>` est une phrase logique ou une `variable booléenne`. Elle doit être formulée précisément pour être sans ambiguïté soit **Vraie**, soit **Fausse**.

La machine teste si la `<condition>` est vraie ou fausse. Dans le cas où la condition n'est pas vraie, l'`<action>` est ignorée et on passe directement à l'instruction suivant le `FinSi`.

📄 Algorithme 1

On souhaite afficher le plus grand nombre pair inférieur ou égal à un entier saisi au clavier.

```
Algorithme : Nombre pair
Variables : Nb, NbPair (entier)

Début
  Afficher("Entrer un entier positif : ") et Saisir(Nb)
  NbPair ← Nb
  Si ((Nb mod 2) = 1) Alors # mod 2 donne le reste par 2
    Afficher("Le nombre saisi est impair")
    NbPair ← Nb - 1
  FinSi
  Afficher("Le plus grand nombre pair inférieur ou égal à",Nb,"est",NbPair)
Fin
```

📄 Pseudo-Code

🐍 Python

En `python`, on utilise le mot clé `if` et les `:` afin de créer le bloc `Si...Alors...FinSi`.


```
1 if <condition> :
2   <instruction>
3 #endif
```

🐍 Code Python

Le `#endif` est un commentaire, il n'est pas traité, mais peut permettre de « mieux » lire le code.

En revanche, **NE PAS OUBLIER LES DEUX POINTS ET L'INDENTATION!!!!**

Python

En , l'exemple 1 donne :

```
1 Nb = int(input("Saisir un entier positif"))
2 NbPair = Nb
3 if ((Nb % 2) == 1) :    # %2 donne le reste par 2
4     print ("Le nombre saisi est impair")
5     NbPair = Nb - 1
6 print(f"Le plus gd nb pair inférieur ou égal à {Nb} est {NbPair}")
```

 Code Python

Algorithme 2

On veut afficher le plus grand de deux nombres réels saisis.

Algorithme : Maximum de deux nombres
Variables : A, B, Max (réels)

Début

Afficher("Entrer deux réels") et Saisir(A,B)

Max ← A # on affecte la valeur de A au Max

Si (B>A) Alors

Max ← B


FinSi

Afficher("Le maximum est",Max)


Fin

 Pseudo-Code

Python

L'exemple précédent, en , donne :

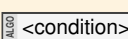
```
1 .
2 .
3 .
4 .
5 .
6 .
7 .
```

 Code Python

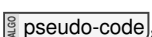
II. Structure alternative

II.1. Cas de deux conditions

Idée

Le principe est le même, mais si la  est fausse, d'autres instructions sont réalisées.

Algorithme

En , on écrit :

```
Si <condition> Alors
    <action 1>
Sinon
    <action 2>
FinSi
```

 Pseudo-Code

Propriété

Si la **<condition>** est vérifiée, l'instruction ou le groupe d'instructions précédées de **Alors** est exécuté. Dans le cas contraire, ce sont les instructions précédées de **Sinon** qui sont exécutées. Il s'agit ici d'une **alternative**, c'est à dire que seul l'un des deux groupes d'instructions est exécuté.

Python

En logopython, on utilise **if** et **else** afin de créer le bloc **Si...Alors...Sinon...FinSi** :

```
1 if <condition> :
2     <instruction 1>
3 else :
4     <instruction 2>
5 #endif
```

Code Python

Remarque

Attention : Dans une structure conditionnelle, le choix d'une **<condition>** ou de la **<condition contraire>** peut avoir de l'influence non seulement sur la lisibilité, mais parfois sur la rapidité d'exécution d'un programme.

Algorithme 3

On écrit un autre algorithme d'affichage du plus grand entier pair inférieur ou égal à un entier saisi :

```
Algorithme : Nombre pair v2
Variables : Nb, NbPair (entier)

Début
    Afficher("Entrer un entier positif") et Saisir(Nb)
    Si ((Nb mod 2) = 0) Alors
        NbPair ← Nb
    Sinon
        NbPair ← Nb - 1
    FinSi
    Afficher("Le plus grand nombre pair inférieur ou égal à",Nb,"est",NbPair)
Fin
```

Pseudo-Code

Python

L'exemple précédent, en **python**, donne :

```
1 .
2 .
3 .
4 .
5 .
6 .
```

Code Python

Algorithme 4


On écrit une autre version de l'algorithme qui affiche le plus grand de deux réels saisis :

Algorithmme : Maximum v2
Variables : A,B,Max (réels)

Début
Afficher("Entrer deux nombres réels") et Saisir(A,B)
Si (A>B) Alors
 Max ← A
Sinon
 Max ← B
FinSi
Afficher("Le plus grand de deux nombres saisis est",Max)
Fin

Pseudo-Code

Python

L'exemple précédent, en , donne :

```
1 .
2 .
3 .
4 .
5 .
6 .
7 .
```

Code Python

II.2. Cas d'au moins trois conditions

i Idée

Si l'action à exécuter dépend de plus de 2 conditions, on peut procéder de plusieurs façons.

Algorithme - Succession de Si...Alors

Si <condition 1> Alors
 <action 1>
FinSi
Si <condition 2> Alors
 <action 2>
...
FinSi

Pseudo-Code

Python

En , cela peut donner :

```
1 if <condition 1> :
2     <instruction 1>
3 if <condition 2> :
4     <instruction 2>
5 ...
```

Code Python

Algorithme - Structures imbriquées


Pseudo-Code

```

Si <condition 1> Alors
  <action 1>
Sinon
  Si <condition 2> Alors
    <action 2>
  Sinon
    Si <condition 3> Alors
      <action 3>
    Sinon
      ...
  FinSi
FinSi
FinSi

```

Python

En , cela peut donner :

Code Python

```

1  if <condition 1> :
2      <instruction 1>
3  else :
4      if <condition 2> :
5          <instruction 2>
6      else :
7          if <condition 3> :
8              <instruction 3>
9          else :
10             ...

```

Remarque

On notera que la présence des **indentations** est indispensable à la compréhension des structures imbriquées ! Cependant, cette dernière structure est lourde. On pourra donc également utiliser une succession de **Si/SinonSi**. Un avantage de l'utilisation de **Si/SinonSi** est l'alignement vertical des conditions :

- le code est plus lisible et il y a moins de possibilité de se tromper à cause des multiples indentations !

Algorithme - SinonSi

Pseudo-Code

```


Si <condition 1> Alors
  <action 1>
SinonSi <condition 2> Alors
  <action 2>
...
Sinon #facultatif
  ...
FinSi

```

Illustration

Structure *SinonSi*Structure *Si Sinon*


Python

En , cela peut donner, avec la structure `if...elif...else...` :

```

1 if <condition 1> :
2     <instruction 1>
3 elif <condition 2> :
4     <instruction 2>
5 ...
6 else :      # facultatif (dans tous les autres cas)
7     ...

```

 Code Python

III. Exemple

III.1. Cadre

Cadre

Écrire un algorithme qui saisit l'âge (variable `Age` entier positif) de l'utilisateur, et affiche, selon les cas, s'il est capable de voter ou de conduire (variables `Voter` de type `booléen` et `Conduire` de type `chaîne`).

Rappels et instructions

L'écriture d'un `algorithme` obéit à des règles précises et doit comporter :

- un `nom` (répondant aux mêmes contraintes que les noms de variables) ;
- son `rôle` (un commentaire décrivant ce qu'il fait et éventuellement sa façon de procéder) ;
- la déclaration des `variables` utilisées (noms et types) ;
- un `début` et une `fin` encadrant la suite d'`instructions` constituant l'algorithme ;
- des `commentaires` afin d'être facilement lisible et compréhensible.

On signalera un commentaire par le symbole `#`. Tout ce qui suit ce symbole sur la même ligne sera ignoré.

III.2. Première version

Algorithme - v1

```


Algorithme : Voter ou conduire (v1)
Variables : Age (entier), Voter (booléen), Conduire (chaîne)

Début
    Afficher("Quel est votre âge ?") et Saisir(Age)
    Si (Age ≥ 18) Alors
        Voter ← Vrai
        Conduire ← "Conduire seul"
    FinSi
    Si (Age ≥ 15) et (Age < 18) Alors
        Voter ← Faux
        Conduire ← "Possible accompagné"
    FinSi
    Si (Age < 15) Alors
        Voter ← Faux
        Conduire ← "Impossible"
    FinSi
    Afficher("Voter :", Voter, "Conduire :", Conduire)
Fin


```

 Pseudo-Code

Python

En , on peut proposer :

```
1 #VoterConduire v1
2 .
3 .
4 .
5 .
6 .
7 .
8 .
9 .
10 .
11 .
12 .
```

 Code Python

III.3. Deuxième version


Algorithme - v2

Algorithme : Voter ou conduire (v2)
Variables : Age (entier), Voter (booléen), Conduire (chaîne)

Début
Afficher("Quel est votre âge ?") et Saisir(Age)
Si (Age \geq 18) Alors #il est majeur
 Voter \leftarrow Vrai
 Conduire \leftarrow "Conduire seul"
Sinon #il est mineur donc ne peut pas voter
 Voter \leftarrow Faux
 Si (Age \geq 15) Alors
 Conduire \leftarrow "Possible accompagné"
 Sinon
 Conduire \leftarrow "Impossible"
 FinSi
FinSi
Afficher("Voter :",Voter,"Conduire :",Conduire)
Fin

 Pseudo-Code

Python

En , on peut proposer :

```
1 #VoterConduire v2
2 .
3 .
4 .
5 .
6 .
7 .
8 .
9 .
10 .
11 .
12 .
```

 Code Python

III.4. Troisième version

Algorithme - v3


Pseudo-Code

```

Algorithme : Voter ou conduire (v3)
Variables : Age (entier), Voter (booléen), Conduire (chaîne)

Début
  Afficher("Quel est votre âge ?") et Saisir(Age)
  Si (Age ≥ 18) Alors
    Voter ← Vrai
    Conduire ← "Conduire seul"
  SinonSi (Age ≥ 15) Alors
    Voter ← Faux
    Conduire ← "Possible accompagné"
  Sinon
    Voter ← Faux
    Conduire ← "Impossible"
  FinSi
  Afficher("Voter :",Voter,"Conduire :",Conduire)
Fin
  
```

Python

En , on peut proposer :

Code Python

```

1 #VoterConduire v3
2 .
3 .
4 .
5 .
6 .
7 .
8 .
9 .
10 .
11 .
12 .
  
```

Humour

