Algorithmique et Programmation

Plan

- Tâches de l'ordinateur
- Notion de codage
- Fonctionnement de l'ordinateur
- Dialoguer avec l'ordinateur
- C'est quoi la programmation?
- Algorithme
- Notion de variable
- Instruction d'affectation
- Instruction d'écriture
- Instruction de lecture
- La condition
- La répétition

Tâches de l'ordinateur

- Diverses application:
  - Edition de feuilles de paye
  - Gestion de stock
  - Jeux
  - Traitement de texte
  - Montage vidéo
  - …

Tâches de l'ordinateur

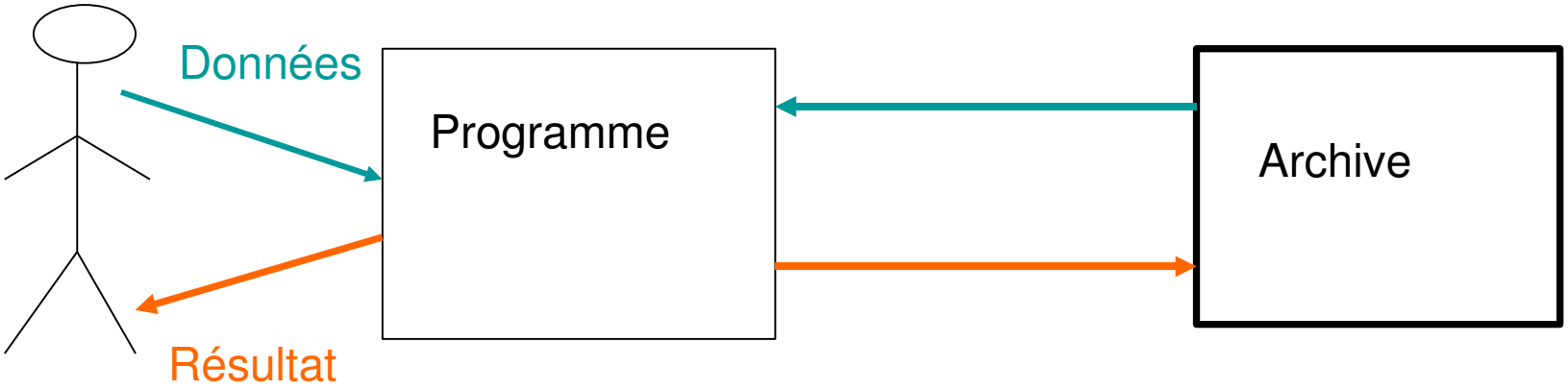
- Programme = source de diversité
  - A chaque tâche correspond un programme
- L'ordinateur est capable de mettre en mémoire un programme puis l'exécuter
- Un programme est constitué d'une suite d'instructions.
- Une instruction spécifie
  - Les opérations à exécuter
  - Les arguments
- Puissance = vitesse d'exécution
- Souplesse = programme

Données du programme et résultats

- Exemple: on dispose d'un programme qui calcule la moyenne des notes.
  - Celui-ci a besoin qu'on lui fournisse les notes (données)
  - Pour qu'il nous retourne la moyenne (résultat)
- Autre exemple: établissement d'un bulletin de paye:
  - Données: NSS, nombre d'heures, grade, …
  - Résultat: salaire net, salaire brut, retenues, …

Communication ou archivage

- D'où viennent les données ? Où vont les résultats?



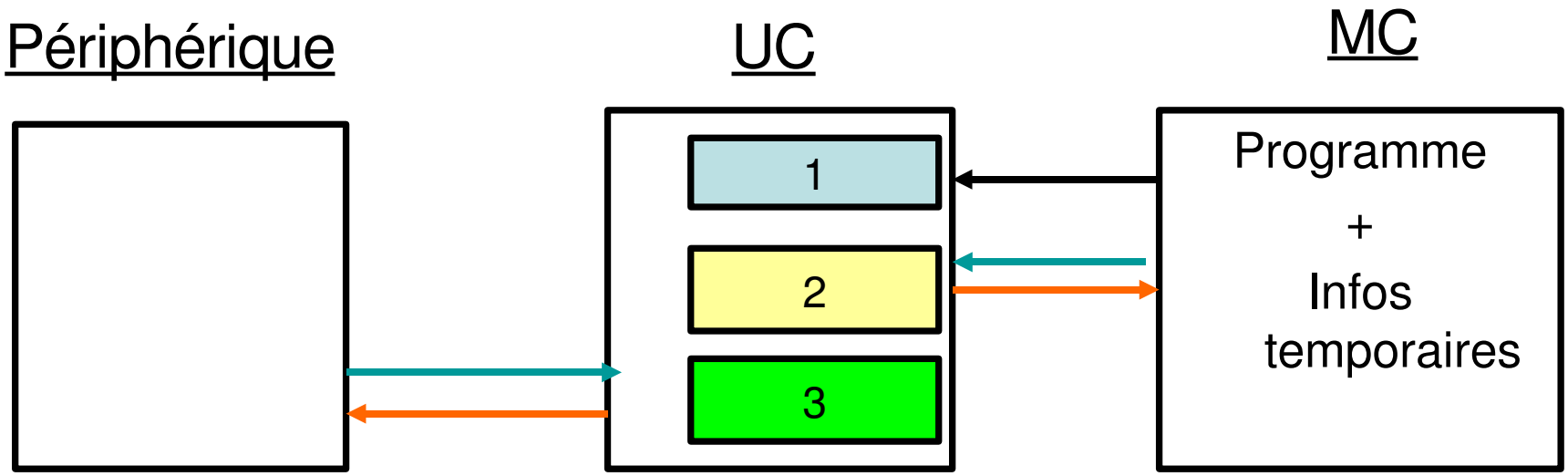
Notion de codage

- Toutes les informations traitées par l'ordinateur sont en binaire
  - Quand on tape sur une touche du clavier, l'ordinateur la transforme en binaire
  - Quand l'ordinateur affiche sur l'écran un résultat, il fait l'opération inverse
- Nous aussi on utilise le codage
  - 13, treize, XIII
- Nous avons interprété XIII par le nombre 13. Comment on a pu dire que ce ne sont pas les lettres X et I ?
- Pour interpréter les données, l'ordinateur a en plus besoin de connaître leurs types

Fonctionnement de l'ordinateur

- Il traite l'informations grâce à un programme qu'il mémorise. Il communique et archive des informations
- Mémoire centrale: Programme+infos temporaires
- Unité centrale: chargée de prélever une à une les instructions du programme
  - Deux types d'instructions
    - Opérations internes (addition, soustraction, …)
    - Opérations de communication (affichage, archivage, …)
- Périphériques: d'entrée, de sortie, d'entrée/sortie

Fonctionnement de l'ordinateur



1. Prélèvement d'une instruction
2. Exécution de l'instruction avec possibilité d'échange avec la MC
3. Exécution d'une instruction d'échange avec un périphérique

Organisation de la MC

- C'est une grille où chaque *case* peut prendre la valeur 0 ou 1 (bit)
- On ne manipule pas des cases mais des ensembles de case qu'on appelle *mots*
- Généralement un mot correspond à un *octet* (8 bits)
- Chaque mot a une *adresse*.

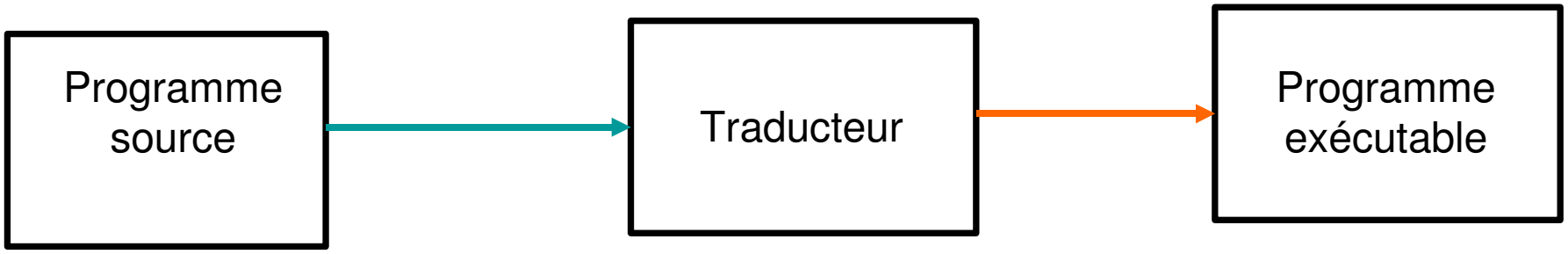
Unité centrale

- Sait exécuter des opérations très simples:
  - Addition, soustraction, comparaison, …
- Chaque instruction du programme doit préciser
  - la nature de l'opération (son code binaire)
  - la ou les adresses sur lesquelles porte l'opération
- Les instructions sont exécutées l'une à la suite de l'autre
  - Sauf si on rencontre une opération de branchement

Programmation

- L'ordinateur ne comprend que le binaire, est-ce pour autant qu'on doive écrire des programmes en binaire ?
- Il existe des langages de programmation dits « évolués » (proches du langage courant, généralement de l'anglais)
- Pour chaque langage, il existe un programme « qui le traduit » en binaire

Traduction des programmes



Il existe essentiellement deux modes de traduction

- Compilation: la traduction se fait une fois pour toute
- Interprétation: a chaque fois qu'on veut exécuter le programme, l'interprète traduit une instruction à la fois. Une fois que celle-ci est exécutée, il passe à l'instruction suivante.

Programmation

- A priori, écriture de programmes dans un langage de programmation (C, Java, Pascal, Visual Basic, Fortran, Python, Perl, …)

- Or il y a plusieurs langages, est-ce que ça veut dire qu'il existe plusieurs sortes de programmation?

- En réalité, la plupart des langages utilisent les mêmes concepts

  ➔ dans le cours, on utilisera une notation particulière: notation algorithmique

Programmation

- 2 étapes:
  1. Analyse du problème et recherche du moyen d'aboutir au résultat à partir des données dont on dispose ➔ écriture d'un algorithme
  2. Traduction de l'algorithme dans un langage de programmation

Algorithme

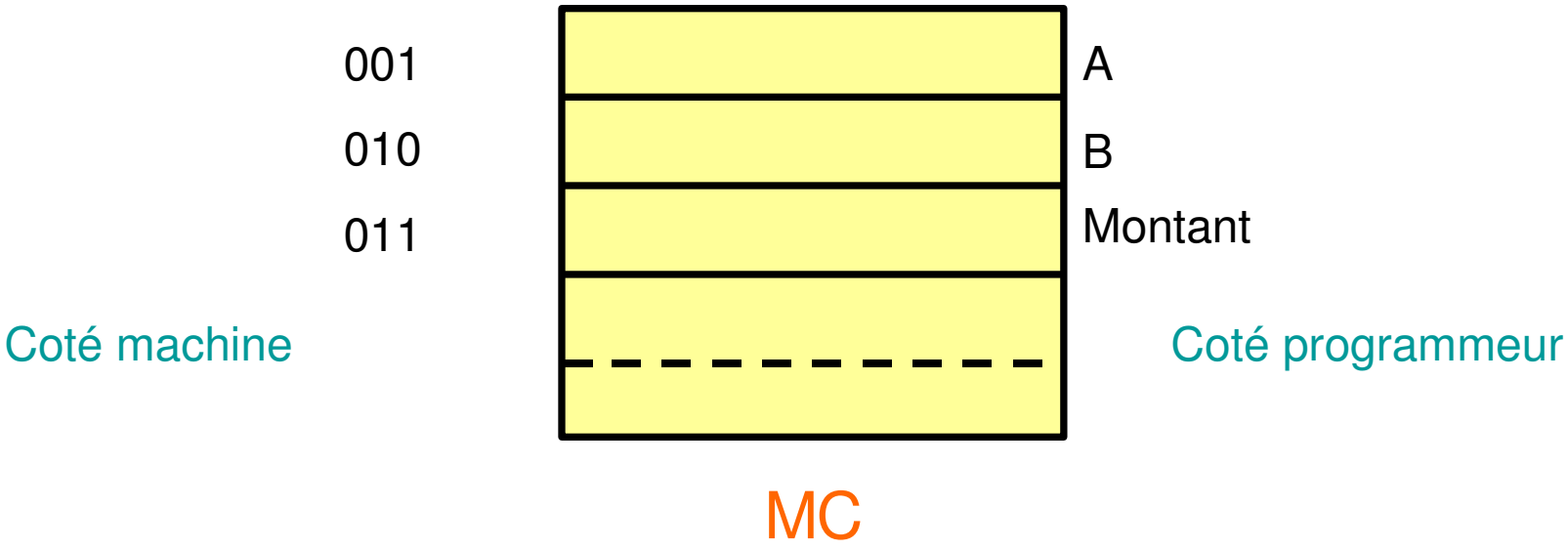
- Une description des différentes étapes permettant de résoudre un problème quelconque
- Exemple: résolution d'une équation du 2nd degré

$$ax^2 + bx + c = 0$$

1. Connaître les valeurs de *a, b* et *c*
2. Calculer le discriminant $\Delta = b^2 - 4ac$
3. Si $\Delta < 0$ alors pas de solution
    a. Sinon Si $\Delta = 0$ alors solution double = -b/2a
        i. Sinon (donc $\Delta > 0$) alors deux solutions

Notion de variable

- Les variables servent à « nommer » des emplacements ou adresses de la mémoire
- Permettent de manipuler des valeurs sans connaître leurs emplacements exactes



Type d'une variable

- Le type d'une variable permet
  - De savoir quel est l'espace mémoire occupé par une variable
  - Quelles sont les opérations autorisées sur la variable
- Déclaration d'une variable dans un algorithme
  - Variable nom_de_variable: type
  - Exemple:
    - Variable Note: Réel
    - Variable coefficient: entier

Rappel

- Programme = suite d'instructions écrites dans un langage de programmation
- Algorithme = suite d'instructions écrites dans un langage proche du langage naturel
- Variable : le nom d'un emplacement mémoire
  - Un variable est définie par
    - Son nom et
    - Son type

Instruction d'affectation

- Rôle: mettre une valeur dans un emplacement mémoire désigné par son nom
- Syntaxe:

1. nom_variable ← valeur

   Ex: Note ← 15

   On lit, Note reçoit 15 ou bien 15 est affecté à Note

2. nom_variable1 ← nom_variable2

   Ex: Note1 ← Note2

   La variable Note1 reçoit « la valeur » de Note2

3. nom_varible ← expression

   Ex: Moyenne ← (Note1*2 +Note2)/3

   Moyenne reçoit « le résultat » de l'évaluation de l'expression

Instruction d'affectation

- Si la variable Note est égale = 10, à quoi sera-t-elle égale après l'exécution de : Note ← Note + 5
- A quoi seront égales les variables A et B après l'exécution de la suite d'instructions suivante

3. A ←5
4. B ← A+4
5. A ← A+1
6. B ← A - 4

Trace d'un algorithme

Instruction	valeur de A	Valeur de B
0:	?	?
1: A ← 5	5	?
2: B ← A+4	5	9
3: A ← A+1	6	9
4: B ← A-4	6	2

A la fin, A=6 et B=2

Instruction d'écriture

- Rôle: permet de restituer une valeur. Généralement, ça consiste à afficher sur l'écran
- Syntaxe:

3. Ecrire (valeur)

   Ex: Ecrire (4)

4. Ecrire (variable)

   Ex: Ecrire(Note)

5. Ecrire (expression)

   Ex: Ecrire ('La moyenne=', (Note1+Note2)/2)

- Remarque: *Ecrire(Note) n'est pas la même chose que Ecrire('Note')*

Instruction de lecture

- Rôle: Permet d'introduire une donnée au programme. Généralement, on tape la valeur sur le clavier

- Syntaxe: Lire(variable)
  Ex: Lire(Note)

- Effet:
  - à la rencontre de cette instruction, l'ordinateur arrête l'exécution du programme et attend qu'on tape une valeur.
  - On termine la saisie en appuyant sur la touche Entrée.
  - La valeur qu'on tape est affectée à la variable lue

- Remarque: Lire(valeur) et Lire(expression) n'ont pas de sens

Rappel

- Variable = nom d'un emplacement mémoire
  - Nom + type
- Algorithme = programme écrit dans un langage proche du français
  - Suite d'instructions
    - Affectation
    - Lecture
    - Écriture

Algorithme

- Syntaxe:

Algorithme *nom_algo*

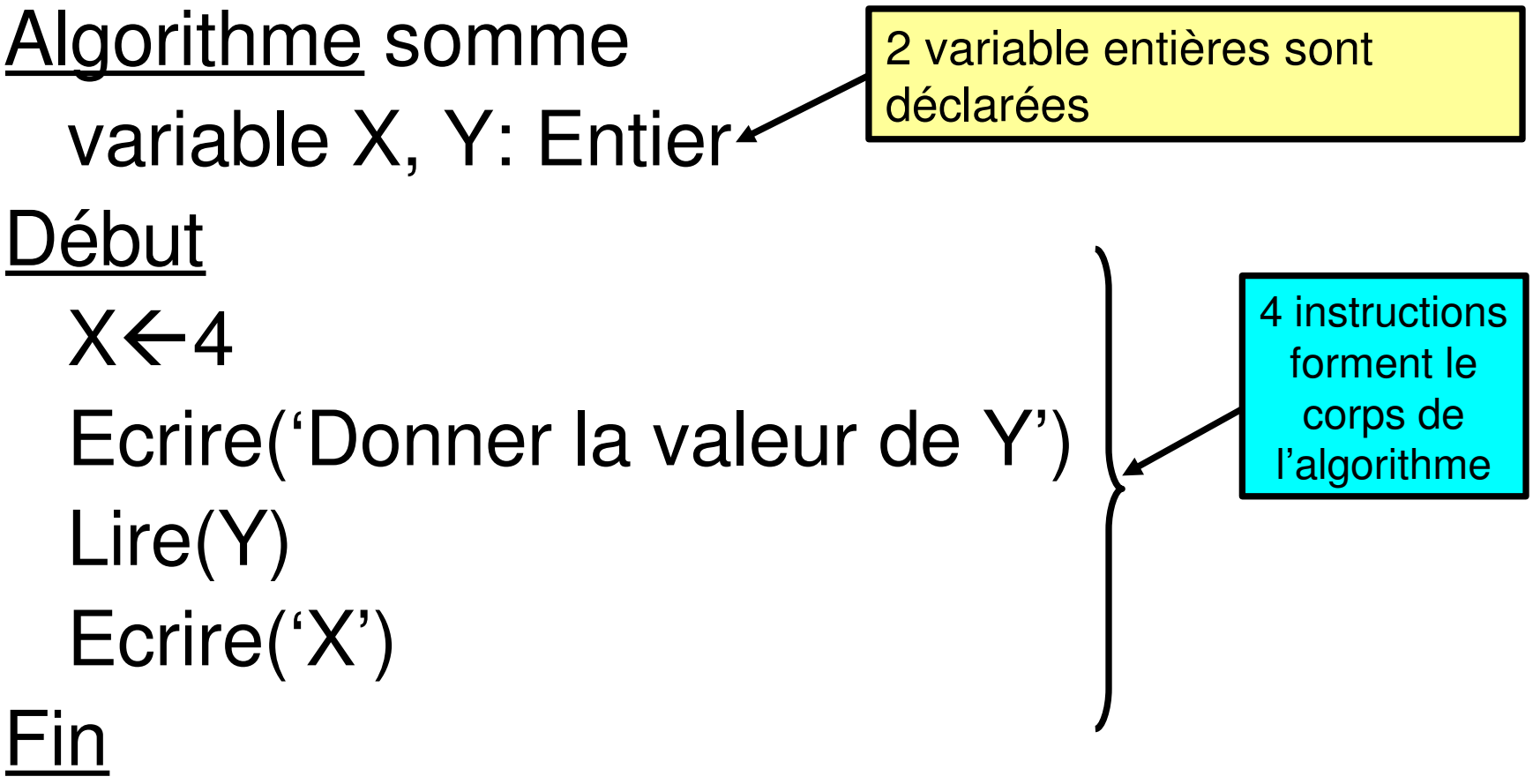
Déclaration des variables

Début

la suite des instructions

Fin

Algorithme: Exemple



Instruction de choix simple

- Rôle: Permet d'exécuter des instructions quand une condition est vérifiée
- Syntaxe:

  Si *condition* Alors

  { Instructions }

  FinSi

Instruction de choix simple

- Ex: on veut afficher un message quand X est positive

  Si X > 0 Alors

  Ecrire(‘X est positive’)

  FinSi

Instruction de choix simple

- La condition peut être composée en utilisant des ‘ET’ et des ‘OU’

Exemple:

Si (((Y=3) OU (Z<4)) ET (X>0)) Alors

{Instructions}

FinSi

Instruction de choix avec alternative

- Rôle: permet de spécifier ce qu'il faut faire dans le cas où la condition n'est pas vérifiée
- Syntaxe:

Si condition Alors
{ Instructions }
Sinon
{ Instructions' }
FinSi

Instruction de choix avec alternative

- Exemple :

Si X > 0 Alors

    Ecrire(‘X est positive’)

Sinon

    Ecrire(‘X n’est pas positive’)

FinSi

Instruction de choix

- On peut imbriquer les conditions
- Exemple

```
Si X > 0 alors
        Ecrire(‘ X supérieure à 0’)

Sinon (ça veut dire que X ≤ 0)
        Si X=0 alors
                Ecrire(‘X égale à 0’)

        Sinon (ça veut dire que X<0)
                Ecrire(‘X inférieure à 0’)
        FinSi
FinSi
```

Exo

- Écrire un algorithme qui permet de résoudre les équations du second degré de la forme

$$ax^2 + bx + c = 0$$

- Idée: Il faut prévoir 3 variables a, b et c qui doivent être renseignées par l'utilisateur

- Il faut prévoir une variable D dans laquelle on met le discriminant

- En fonction des valeurs de D afficher les messages adéquats.

Exo ($3x^2 + 5x + 25$)

```
Algorithme equation
Variable a, b, c, D : Réel
Début
    a ← 3
    b ← 5
    c ← 25
    D ← b² – 4ac
Si D <0 Alors
    Ecrire(« pas de solution »)
Sinon
        Si D=0 Alors
            Ecrire(-b/2a)
        Sinon
            Ecrire(-b-rac(D)/2a)
            Ecrire(-b+rac(D)/2a)
        Finsi
Finsi
Fin
```

Exo : suite

- L'algorithme précédent ne peut être utilisé que pour une seule équation.
  - Comment faire pour qu'il soit capable de résoudre n'importe quelle équation ?
  - Solution: remplacer les affectations par des opérations de lecture

Instruction de répétition

- Rôle: permet de répéter l'exécution d'un ensemble d'instructions tant qu'une condition est vérifiée
- Syntaxe:

```
Tant que condition
        {Instructions}
FinTantQue
```

Exemple:

```
i← 1
Tant que i>0
        Ecrire (i)
        i← i+1
FinTantQue
```

Exo

- Ecrire un algorithme qui permet d'afficher la somme des n premiers entiers non nuls.
- La valeur de n est rentrée par l'utilisateur

Algorithme

```
Algorithme somme
Variable n, i, S : Entier
Début
    Lire(n)
    i ← 1
    S ← 0
    Tant que i <= n
        S ← S + i
        i ← i+1
    FinTantQue
    Ecrire(S)
Fin
```