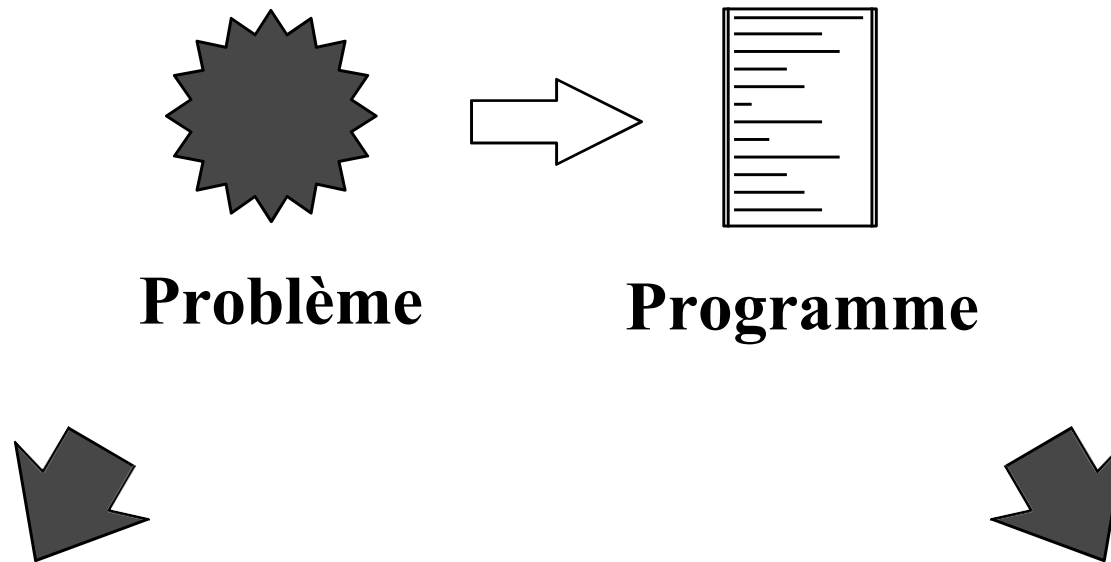

Algorithmique et structures de données

CHAPITRE 1

Introduction/Rappels

La programmation



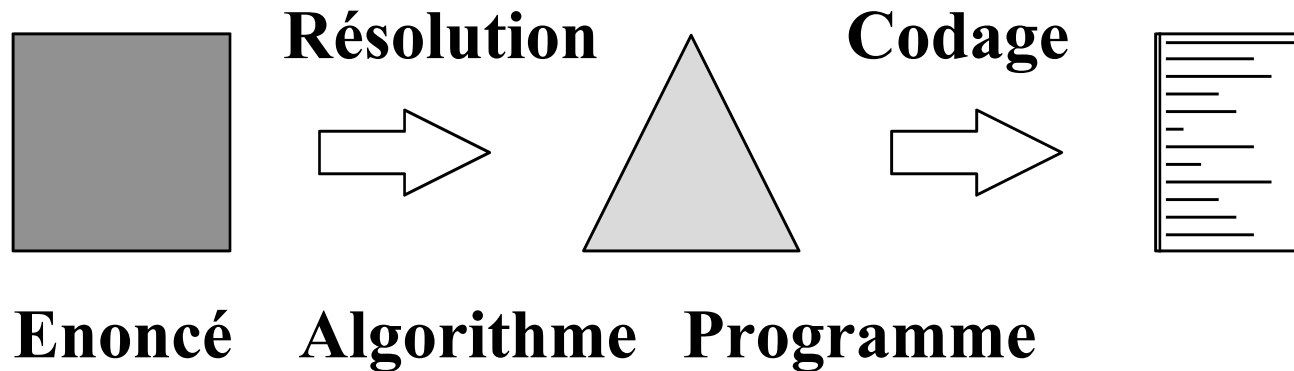
Problème

Programme

- Question à résoudre par une solution informatique
- Instance d'un problème = entrée nécessaire pour calculer une solution du problème

- Ensemble de données
- Ensemble de résultats
= solution informatique au problème
- Description d' un ensemble d'actions
- Exécution dans un certain ordre

Notion d'algorithme



= Description d'un processus de résolution d'un problème bien défini

= Succession d'actions qui, agissant sur un ensemble de ressources (entrée), fourniront la solution (sortie) au problème

- Comment faire pour l'obtenir ?

Pseudo code

Faire la différence entre les contraintes
propres à un langage et les difficultés
inhérentes à un problème donné

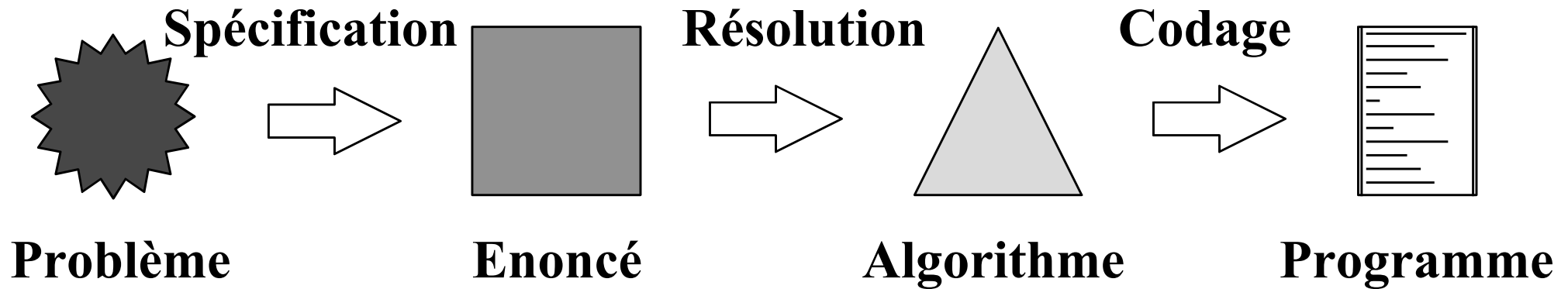
```
lire (n)
pour i ← 0 à n
    si (i mod 2) # 0
        alors afficher(i)
```

Plus abstrait, plus lisible, plus concis...

```
#include<iostream>
using namespace std;
int main ()
{
    int n, i;
    cin>> n;
    for (i=0; i<=n; i++) {
        if (i%2) {
            cout <<i<<endl;
        }
    }
}
```

Met en avant l'essence de l'algorithme

Conception d'un programme



**Ne pas se laisser aveugler par l'objectif final :
le codage !**

Formalisme...

- Un algorithme doit être lisible et compréhensible par plusieurs personnes.
- Il doit donc suivre des règles
- Il est composé d'une entête et d'un corps :
 - l'entête, qui spécifie :
 - le nom de l'algorithme (**Nom :**) son
 - utilité (**Rôle :**)
 - les données “en entrée”, c'est-à-dire les éléments qui sont indispensables à son bon fonctionnement (**Entrée :**)
 - les données “en sortie”, c'est-à-dire les éléments calculés, produits, par l'algorithme (**Sortie :**)
 - les données locales à l'algorithmique qui lui sont indispensables (**Declaration :**)

Formalisme...

- le corps, qui est composé :
 - du mot clef **début**
 - d'une suite d'instructions **indentées**
 - du mot clef **fin**

Formalisme...

■ Exemple de code :

Nom : addDeuxEntiers

Rôle : Additionner deux entiers a et b et mettre le résultat dans c

Entrée : a,b : entier

Sortie : c : entier

Déclaration : -

début

$c \leftarrow a + b$

fin

Les conditionnelles...

- Jusqu'à présent les instructions d'un algorithme étaient **toutes** interprétées **séquentiellement**

- - Nom** : euroVersDollar2
 - Rôle** : Convertisseur des sommes en euros vers le dollar
 - Entrée** : valeurEnEuro : Réel
 - Sortie** : valeurEnDollar : Réel
 - Déclaration** : tauxConversion : Réel
 - début**
 - tauxConversion $\leftarrow$ 1.45
 - valeurEnDollar $\leftarrow$ valeurEnEuro * tauxConversion
 - fin**

- Mais il se peut que l'on veuille conditionner l'exécution d'un algorithme
 - Par exemple: la résolution d'une équation du second degré est conditionnée par le signe de Δ

L'instruction si alors sinon...

- L'instruction `si alors sinon` permet de conditionner l'exécution d'une suite d'instructions à la valeur d'une expression booléenne

- Sa syntaxe est :

si *expression booléenne* **alors**

suite d'instructions exécutées si l'expression est vraie

sinon

suite d'instructions exécutées si l'expression est fausse

finsi

- Le deuxième partie de l'instruction est optionnelle, on peut avoir la syntaxe suivante :

si *expression booléenne* **alors**

suite d'instructions exécutées si l'expression est vraie

fin si

Exemple...

Nom : abs

Rôle : Calcule la valeur absolue d'un entier

Entrée : unEntier : **Entier**

Sortie : laValeurAbsolue : **Entier**

Déclaration : -

début

si unEntier ≥ 0 **alors**

 laValeurAbsolue $\leftarrow$ unEntier

sinon

 laValeurAbsolue $\leftarrow$ -unEntier

fin si

fin

Exemple...

Nom : max

Rôle : Calcule le maximum de deux entiers

Entrée : lEntier1, lEntier2 : **Entier**

Sortie : leMaximum : **Entier**

Déclaration : -

début

si lEntier1 < lEntier2 **alors**

 leMaximum $\leftarrow$ lEntier2

sinon

 leMaximum $\leftarrow$ lEntier1

fin si

fin

Exemple...

Nom : max

Rôle : Calcule le maximum de deux entiers

Entrée : lEntier1, lEntier2 : **Entier**

Sortie : leMaximum : **Entier**

Déclaration : -

début

leMaximum $\leftarrow$ lEntier1

si lEntier2 > lEntier1 **alors**

leMaximum $\leftarrow$ lEntier2

fin si

fin

Les itérations...

- Lorsque l'on veut répéter plusieurs fois un même traitement, plutôt que de copier n fois la ou les instructions, on peut demander à l'ordinateur d'exécuter n fois un morceau de code
- Il existe deux grandes catégories d'itérations :
 - Les itérations **déterministes** : le nombre de boucles est défini à l'entrée de la boucle
 - les itérations **indéterministes** : l'exécution de la prochaine boucle est conditionnée par une expression booléenne

Les itérations déterministes...

- Il existe une seule instruction permettant de faire des boucles déterministes, c'est l'instruction `pour`

- Sa syntaxe est :

pour *identifiant d'une variable de type scalaire* ← *valeur de début* **à** *valeur de fin*
faire

instructions à exécuter à chaque boucle

fin pour

- dans ce cas la variable utilisée prend successivement les valeurs comprises entre *valeur de début* et *valeur de fin*

Exemple...

Nom : somme

Rôle : Calculer la somme des n premiers entiers positifs, $s=0+1+2+\dots+n$

Entrée : n : Naturel

Sortie : s : Naturel

Déclaration : i : Naturel

début

$s \leftarrow 0$

pour $i \leftarrow 0$ **à** n **faire**

$s \leftarrow s+i$

fin pour

fin

Les itérations indéterministes...

- La boucle s'exécutera jusqu'à ce que l'expression booléenne devienne fausse.

- L'instruction **tant que** :

tant que *expression booléenne* **faire**

instructions

fin tant que

- Ce qui signifie: "tant que l'expression booléenne est vraie, on exécute les instructions".

- Si vous ne voulez pas que votre algorithme “tourne” indéfiniment, l'expression booléenne doit faire intervenir des variables dont le contenu doit être modifié par, au moins, une des instructions du corps de la boucle



Un exemple...

Nom : invFact

Rôle : Détermine le plus grand entier e telque $e! \leq n$

Entrée : n : **Naturel** ≥ 1

Sortie : e : **Naturel**

Déclaration : fact : **Naturel**

début

fact $\leftarrow$ 1

$e \leftarrow$ 1

tant que fact $\leq$ n **faire**

$e \leftarrow e+1$

 fact $\leftarrow$ fact * e

fin tant que

$e \leftarrow e-1$

fin

Explication avec $n=10$...

Nom : invFact

Rôle : Détermine le plus grand entier e telque $e! \leq n$

Entrée : n : **Naturel** ≥ 1

Sortie : e : **Naturel**

Déclaration : fact : **Naturel**

début

fact $\leftarrow$ 1

$e \leftarrow 1$

tant que fact $\leq n$ **faire**

$e \leftarrow e+1$

 fact $\leftarrow$ fact * e

fin tant que

$e \leftarrow e-1$

fin

	n	e	fact	fact $\leq n$
fact $\leftarrow$ 1	10	?	1	vrai
$e \leftarrow 1$	10	1	1	vrai
$e \leftarrow e+1$ fact $\leftarrow$ fact * e	10	2	2	vrai
$e \leftarrow e+1$ fact $\leftarrow$ fact * e	10	3	6	vrai
$e \leftarrow e+1$ fact $\leftarrow$ fact * e	10	4	24	faux
$e \leftarrow e-1$	10	3	24	faux

Partie 3:

Pseudo-code schématique

Pseudo-code schématique

- Nous décrirons les tâches à partir des opérations élémentaires suivantes:

□ Lire	□ Additionner
□ Écrire	□ Soustraire
□ TANT QUE condition FAIRE	□ Multiplier
□ POUR ensemble_de_données FAIRE	□ Diviser
□ SI condition ALORS --- SINON	
□ Affecter =	

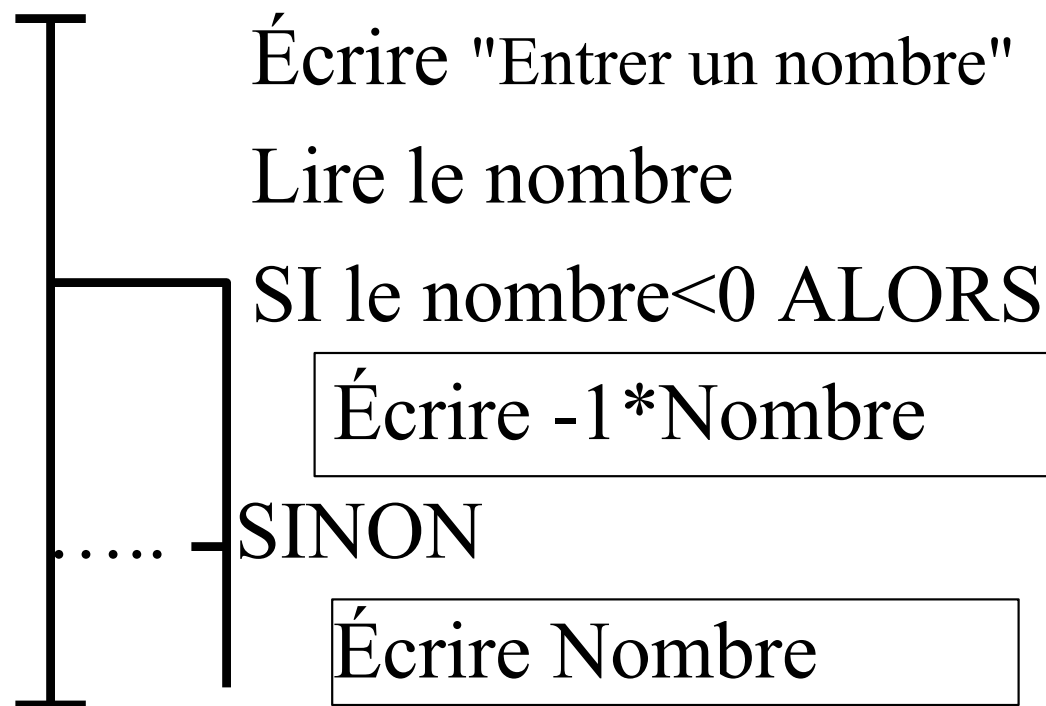
Pseudo-code schématique: Exemple 1

- Lire un nombre et écrire la valeur absolue de ce nombre
 - Quelles sont les entrées? **Un nombre.**
 - Quelles sont les sorties? **Écriture d'un nombre.**
 - Comment trouvons-nous une valeur absolue? **Il faut changer le signe du nombre s'il est négatif.**
 - Est-ce qu'il y a des répétitions ? **Non.**
 - Est-ce que des décisions ou vérifications sont requises? **Oui. Combien? 1. Quelles sont les alternatives? Inversion du signe si le nombre est négatif, sinon on fait rien.** Elles influencent quelles opérations ? **Écriture du nombre.**

Pseudo-code schématique

Exemple 1 - suite

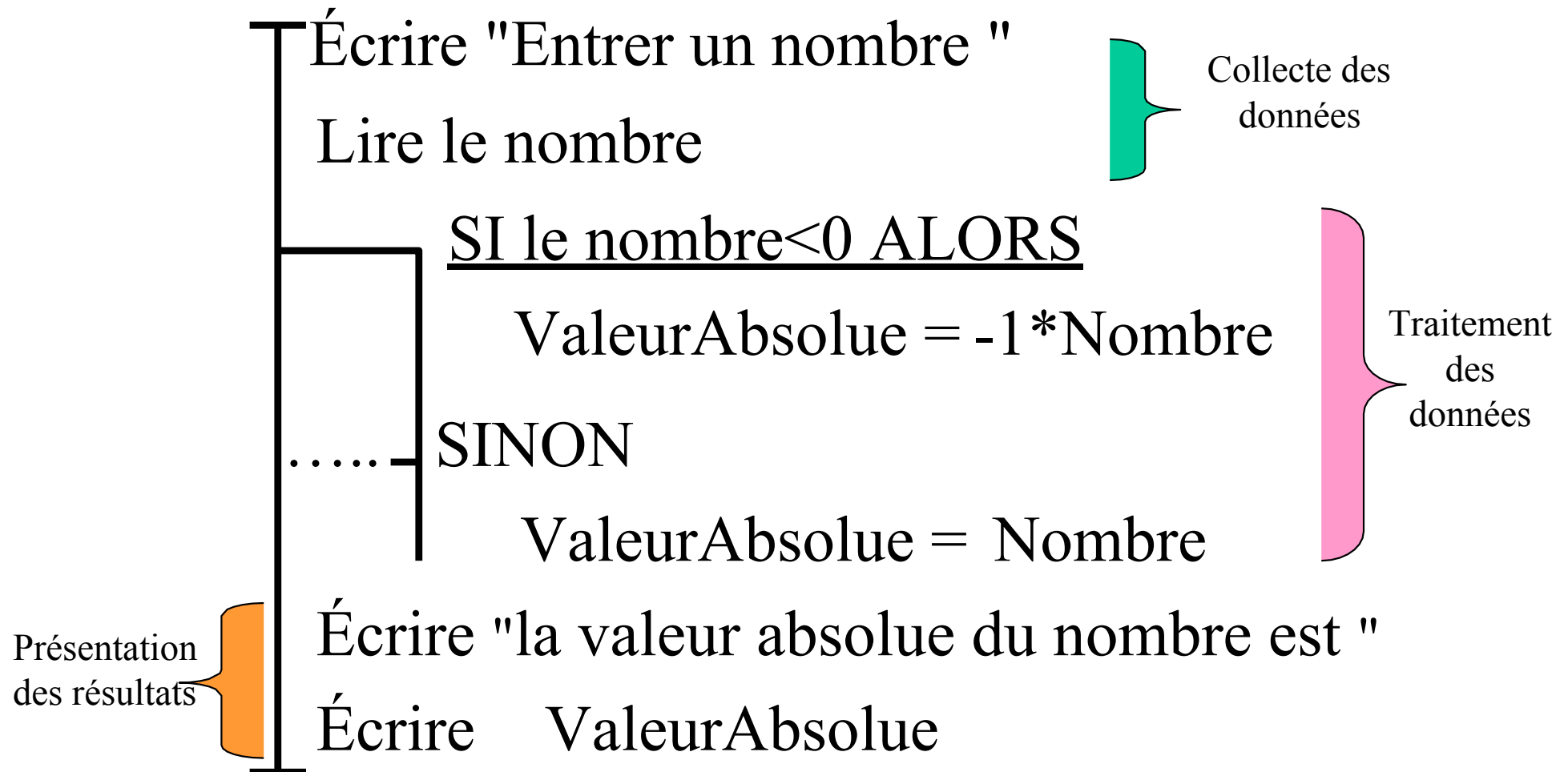
- Lire un nombre et écrire la valeur absolue de ce nombre (suite)



Décalage pour
démontrer la
dépendance du
test nombre < 0

Pseudo-code schématique

- Autre façon



Pseudo-code schématique : Exemple 2

- Deviner un nombre choisi au hasard. Indiquer si le nombre proposé est inférieur ou supérieur au nombre à deviner.
 - Quelles sont les entrées? **Un ou des nombres (dépend si devine le nombre facilement ou non).**
 - Quelles sont les sorties? **Bravo ou trop petit ou trop grand.**
 - Comment trouvons-nous un nombre choisi au hasard? **On propose des nombres tant qu'on n'a pas trouvé le bon.**
 - Est-ce qu'il y a des répétitions ? **Oui.** Combien ? **Une.** Et elles répètent quelles opérations? **Demande un nombre à l'utilisateur et vérifie ce nombre.**

Pseudo-code schématique

- Deviner un nombre choisi au hasard. Indiquer si le nombre proposé est inférieur ou supérieur au nombre à deviner.
 - Est-ce que des décisions ou vérifications sont requises? **Oui.**
 - Combien? **Deux.**
 - Quelles sont les alternatives? **Si le nombre est bien deviné, terminer le programme et écrire à l'écran bravo, ou si le nombre n'est pas deviné, vérifier le sens de l'erreur et écrire trop petit, trop grand.**
 - Elles influencent quelles opérations ? **La répétition, car recommence tant que le nombre n'est pas deviné, et aussi ce qui est écrit à l'écran (trop petit, trop grand).**

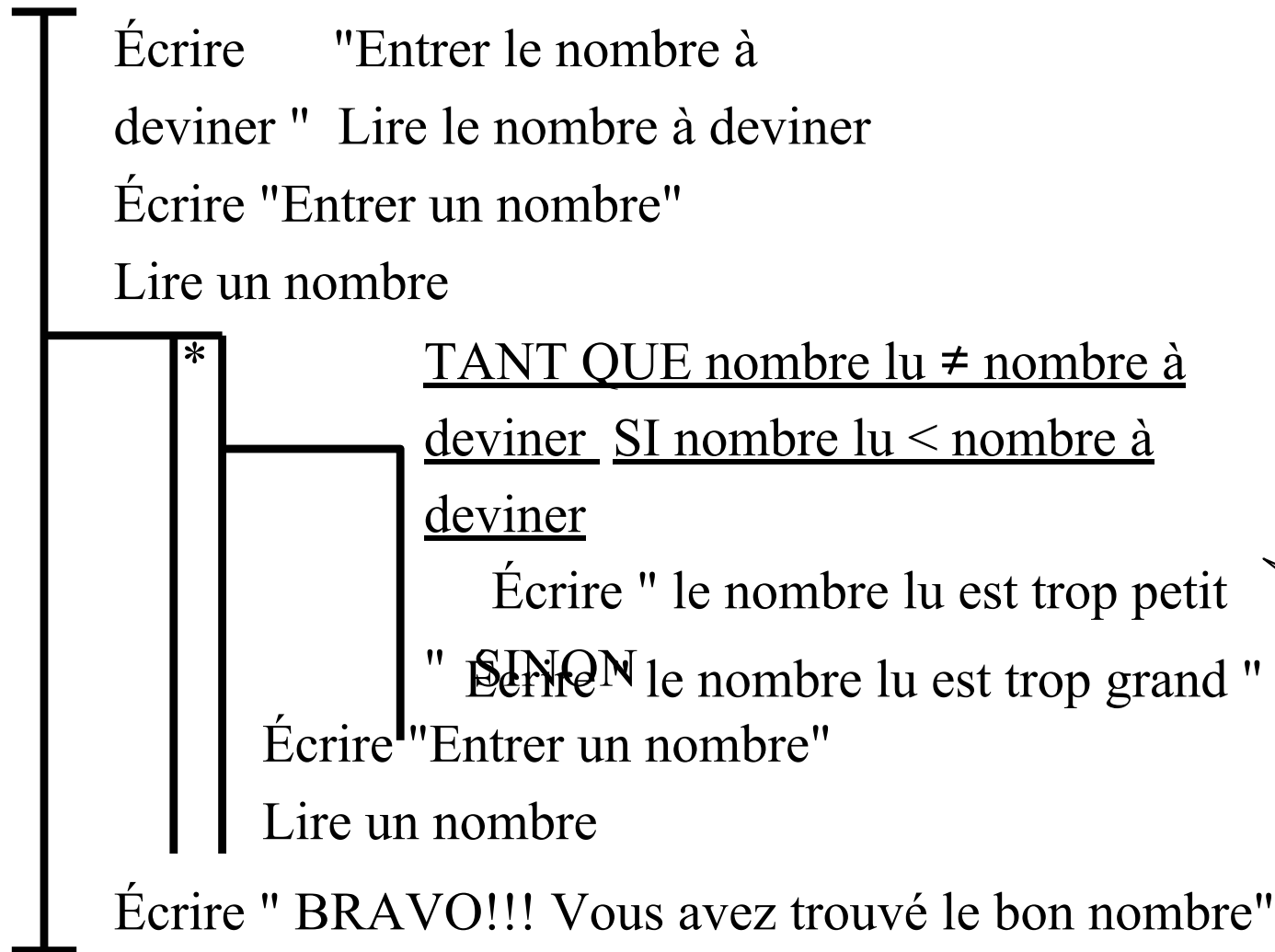
Pseudo-code schématique

- Deviner un nombre choisi au hasard (Version 1)

```
┌ Écrire " Entrer le nombre à deviner"  
  Lire le nombre à deviner  
  Écrire " Entrer un nombre "  
  Lire un nombre  
  TANT QUE nombre ≠ nombre à deviner  
    ┌ SI nombre < nombre à deviner  
      Écrire "Trop petit"  
    SINON  
      Écrire "Trop grand"  
      Afficher "Entrer un nombre"  
      Lire un nombre  
    └ Écrire "BRAVO!!!"
```

Pseudo-code schématique

- Ajout du schéma répétitif



Décalage pour démontrer la dépendance du test du TANT QUE

Pseudo-code schématisque

- Deviner un nombre choisi au hasard (Version 2)

Écrire "Entrer le nombre à deviner "

Lire le nombre à deviner

Devinée = Faux

*

TANT QUE Devinée est Faux

FAIRE Écrire "Entrer un nombre "

Lire un nombre

SI nombre lu = nombre à deviner ALORS

Devinée = Vrai

..... SINON

SI nombre < nombre à deviner ALORS

Écrire " le nombre est trop petit "

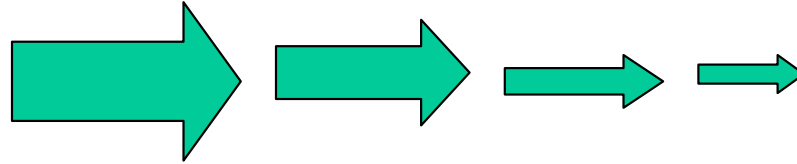
..... SINON

Écrire " le nombre est trop grand "

Écrire " BRAVO, vous avez trouvé le nombre à deviner "

Pensée informatique

Raffinement graduel



- ⇒ Lors de l'élaboration d'un algorithme il est parfois utile d'envisager un processus de raffinement graduel.
- ⇒ Il s'agit dans un premier temps d'énumérer les opérations « évidentes » qui réalisent l'application désirée.
- ⇒ Ces opérations évidentes ne sont pas nécessairement des opérations élémentaires.
- ⇒ Par la suite il suffit de détailler les opérations non élémentaires à l'aide d'opérations élémentaires.

Pseudo-code schématique

- Lire trois nombres entiers quelconques et vérifier si l'un de ces nombres est le produit des 2 autres. Par exemple, si 2, 10 et 5 sont lus alors le deuxième nombre est le résultat de la multiplication des deux autres.
 - Quelles sont les entrées? **Trois nombres.**
 - Quelles sont les sorties? **Oui ou non.**
 - Comment vérifie-t-on qu'un nombre est le produit de deux autres? **On vérifie si en multipliant deux des nombres, on obtient le troisième.**
 - Est-ce qu'il y a des répétitions ? **Oui.** Combien ? **Une.** Et elles répètent quelles opérations? **La vérification du produit pour chaque combinaison.**

Pseudo-code schématique

- Lire trois nombres entiers quelconques et vérifier si l'un de ces nombres est le produit des 2 autres. Par exemple, si 2, 10 et 5 sont lus alors le deuxième nombre est le résultat de la multiplication des deux autres.
 - Est-ce que des décisions ou vérifications sont requises?
Oui. Combien? Une par combinaison de nombres.
Quelles sont les alternatives? **Pour chaque combinaison de nombres (3), il faut vérifier si un nombre égal le produit des deux autres. Elles influencent quelles opérations ? La réponse finale.**

Pseudo-code schématique

- Lire trois nombres entiers quelconques et vérifier si l'un de ces nombres est le produit des 2 autres. Par exemple, si 2, 10 et 5 sont lus alors le deuxième nombre est le résultat de la multiplication des deux autres.

┌ Écrire "Entrer trois nombres"
├ Lire les trois nombres
└ 1. Vérifier si l'un des nombres est le produit de deux autres
Écrire le résultat

Opération à raffiner

Suite et fin

