DUT SRC – IUT de Marne-la-Vallée
19/11/2012
INF120 - Algorithmique

Cours 5
Méthodologie pour l'algorithmique

Philippe Gambette

Sources

- *Le livre de Java premier langage*, d'A. Tasso
- Cours INF120 de J.-G. Luque
- Cours de J. Henriet : http://julienhenriet.olympe-network.com/Algo.html

Plan du cours 5 – Méthodologie pour l'algorithmique

- Résumé de l'épisode précédent
- Compétences acquises en INF120
- Comprendre un algorithme
- Concevoir un algorithme
- Vérifier un algorithme

Plan du cours 5 – Méthodologie pour l'algorithmique

- Résumé de l'épisode précédent
- Compétences acquises en INF120
- Comprendre un algorithme
- Concevoir un algorithme
- Vérifier un algorithme

Résumé de l'épisode précédent

Entrées/sorties :

- périphériques matériels pour interagir avec l'ordinateur

- récupération de données en entrée et envoi de données en sortie
  (exemples : entrées clavier et affichage en ligne de commande)

Fonctions (= algorithmes)

- définir une fonction =
  → la déclarer : définir son nom, ses variables d'entrée et leur type, son type de sortie
  → écrire l'ensemble de ses instructions (déclarations de variables, affectations, tests, boucles, appels d'algorithmes)

- appeler une fonction =
  → donner les bons paramètres en entrée
  → récupérer le résultat en sortie

Plan du cours 5 – Méthodologie pour l'algorithmique

- Résumé de l'épisode précédent
- Compétences acquises en INF120
- Comprendre un algorithme
- Concevoir un algorithme
- Vérifier un algorithme

Compétences acquises en INF120

- Connaître les éléments de base d'un algorithme
- Savoir lire et comprendre un algorithme
- Savoir concevoir un algorithme pour résoudre un problème
- Savoir programmer un algorithme

Compétences acquises en INF120

Connaître les éléments de base d'un algorithme

Un algorithme résout un problème, en fournissant un **résultat en sortie** à partir de **données en entrée**.

Pour cela, il utilise plusieurs types d'**instructions** :
- des **affectations** dans des **variables** (mémoires)
- des **boucles**
- des **appels** à d'autres algorithmes
- des **tests**
- des “**lectures**” d'entrées et “**renvois**” de sorties

Chaque **variable** a un **nom**. On doit :
- la **déclarer** en définissant son **type** (ensemble de valeurs possibles)
- puis l'**initialiser** (lui donner une **valeur**)

avant de l'utiliser.

Compétences acquises en INF120

en pseudo-code

Connaître les éléments de base d'un algorithme

Un algorithme résout un problème, en fournissant un **résultat en sortie** à partir de **données en entrée**.

Entrées : entiers *i* et *j*
Type de sortie : entier

une instruction par ligne !

Pour cela, il utilise plusieurs types d'**instructions** :

- des **affectations** dans des **variables** (mémoires) produit ← entier1+produit (variable / valeur)
- des **boucles** Tant que … faire : … Fin TantQue
- des **appels** à d'autres algorithmes Addition(3,5) — entrées de l'algorithme entre parenthèses, respectant l'ordre de déclaration des entrées
- des **tests** Si ... alors : ... sinon : … FinSi
- des "**lectures**" d'entrées et "**renvois**" de sorties

renvoyer *i+j*

Chaque **variable** a un **nom**. On doit :

- la **déclarer** en définissant son **type** (ensemble de valeurs possibles)
- puis l'**initialiser** (lui donner une **valeur** par une affectation)

avant de l'utiliser.

Variables : entiers *a* et *b* (type)

a ← 2
b ← *a*+2

Compétences acquises en INF120

en Java

Connaître les éléments de base d'un algorithme

Un algorithme résout un problème, en fournissant un **résultat en sortie** à partir de **données en entrée**.

public static int Addition(int i, int j){...}
type de sortie
entrées précédées de leur type

Pour cela, il utilise plusieurs types d'**instructions** :

instructions finies par ";"

- des **affectations** dans des **variables** (mémoires)
- des **boucles**
- des **appels** à d'autres algorithmes
- des **tests**
- des "**lectures**" d'entrées et "**renvois**" de sorties

produit = entier1+produit
variable
valeur

while(…){...}

Addition(3,5)
entrées de l'algorithme entre parenthèses, respectant l'ordre de déclaration des entrées

if(...){...}else{...}

return i+j;

Chaque **variable** a un **nom**. On doit :

- la **déclarer** en définissant son **type** (ensemble de valeurs possibles)
- puis l'**initialiser** (lui donner une **valeur** par une affectation)

avant de l'utiliser.

int a, b;
type

a = 2
b = a+2

Plan du cours 5 – Méthodologie pour l'algorithmique

- Résumé de l'épisode précédent
- Compétences acquises en INF120
- Comprendre un algorithme
- Concevoir un algorithme
- Vérifier un algorithme

Méthodologie pour l'algorithmique : comprendre

Savoir lire et comprendre un algorithme

Premiers éléments à identifier :
- qu'est-ce que l'algorithme **prend en entrée** ? **Combien** de variables, de quel **type** ?
- qu'est-ce que l'algorithme **renvoie en sortie** ? **Rien** ? Ou bien **une** variable ? De quel **type** ?

Ensuite :
- quels sont les autres **algorithmes appelés** par l'algorithme ?

Enfin :
- faire la **trace** de l'algorithme, c'est-à-dire l'essayer sur un **exemple**
(… ou plusieurs pour passer au moins une fois par toutes les instructions de l'algorithme)
et voir ce que valent **toutes les variables à chaque étape** (et noter ces valeurs dans un tableau contenant une ligne par variable et une colonne par étape),
- noter en particulier le **résultat obtenu en sortie** pour une **entrée testée.**

Comprendre un algorithme

```
public static void DessineToit(int x, int y, Graphics g){
    int i=0;
    Color rouge=couleurRGB(200,0,0);
    while (i<51){
        dessineLigne(g,x+2*i,y-i,x+2*i,y,rouge);
        dessineLigne(g,x+2*i+1,y-i,x+2*i+1,y,rouge);
        dessineLigne(g,x+200-2*i,y-i,x+200-2*i,y,rouge);
        dessineLigne(g,x+200-2*i+1,y-i,x+200-2*i+1,y,rouge);
        i=i+1;
    }
}
```

Premiers éléments à identifier :
- qu'est-ce que l'algorithme **prend en entrée** ? **Combien** de variables, de quel type ?
- qu'est-ce que l'algorithme **renvoie en sortie** ? **Rien** ? Ou bien **une** variable ? De quel **type** ?

Ensuite :
- quels sont les autres **algorithmes appelés** par l'algorithme ?

Enfin :
- faire la **trace** de l'algorithme, c'est-à-dire l'essayer sur un **exemple** et voir ce que valent **toutes les variables à chaque étape**,
- noter en particulier le **résultat obtenu en sortie** pour une **entrée testée.**

Comprendre un algorithme

```
public static void DessineToit(int x, int y, Graphics g){
    int i=0;
    Color rouge=couleurRGB(200,0,0);
    while (i<51){
        dessineLigne(g,x+2*i,y-i,x+2*i,y,rouge);
        dessineLigne(g,x+2*i+1,y-i,x+2*i+1,y,rouge);
        dessineLigne(g,x+200-2*i,y-i,x+200-2*i,y,rouge);
        dessineLigne(g,x+200-2*i+1,y-i,x+200-2*i+1,y,rouge);
        i=i+1;
    }
}
```

- qu'est-ce que l'algorithme **prend en entrée** ? **Combien** de variables, de quel **type** ?
- qu'est-ce que l'algorithme **renvoie en sortie** ? **Rien** ? Ou bien **une** variable ? De quel **type** ?

Comprendre un algorithme

```
public static void DessineToit(int x, int y, Graphics g){
    int i=0;
    Color rouge=couleurRGB(200,0,0);
    while (i<51){
        dessineLigne(g,x+2*i,y-i,x+2*i,y,rouge);
        dessineLigne(g,x+2*i+1,y-i,x+2*i+1,y,rouge);
        dessineLigne(g,x+200-2*i,y-i,x+200-2*i,y,rouge);
        dessineLigne(g,x+200-2*i+1,y-i,x+200-2*i+1,y,rouge);
        i=i+1;
    }
}
```

- quels sont les autres **algorithmes appelés** par l'algorithme ?

Comprendre un algorithme

```
public static void DessineToit(int x, int y, Graphics g){
    int i=0;
    Color rouge=couleurRGB(200,0,0);
    while (i<51){
        dessineLigne(g,x+2*i,y-i,x+2*i,y,rouge);
        dessineLigne(g,x+2*i+1,y-i,x+2*i+1,y,rouge);
        dessineLigne(g,x+200-2*i,y-i,x+200-2*i,y,rouge);
        dessineLigne(g,x+200-2*i+1,y-i,x+200-2*i+1,y,rouge);
        i=i+1;
    }
}
```

- quels sont les autres **algorithmes appelés** par l'algorithme ?

“dessineLigne(Graphics g, int abscisse1, int ordonnee1, int abscisse2, int ordonnee2, Color couleur), qui dessine sur l'objet g une ligne commençant au point de coordonnées (abscisse1,ordonnee1) et se terminant au point de coordonnées (abscisse2,ordonnee2).”

Comprendre un algorithme

```
public static void DessineToit(int x, int y, Graphics g){
    int i=0;
    Color rouge=couleurRGB(200,0,0);
    while (i<51){
        dessineLigne(g,x+2*i,y-i,x+2*i,y,rouge);
        dessineLigne(g,x+2*i+1,y-i,x+2*i+1,y,rouge);
        dessineLigne(g,x+200-2*i,y-i,x+200-2*i,y,rouge);
        dessineLigne(g,x+200-2*i+1,y-i,x+200-2*i+1,y,rouge);
        i=i+1;
    }
}
```

- faire la **trace** de l'algorithme, c'est-à-dire l'essayer sur un **exemple** et voir ce que valent **toutes les variables à chaque étape**

Quel exemple ?

Comprendre un algorithme

```
public static void DessineToit(int x, int y, Graphics g){
    int i=0;
    Color rouge=couleurRGB(200,0,0);
    while (i<51){
        dessineLigne(g,x+2*i,y-i,x+2*i,y,rouge);
        dessineLigne(g,x+2*i+1,y-i,x+2*i+1,y,rouge);
        dessineLigne(g,x+200-2*i,y-i,x+200-2*i,y,rouge);
        dessineLigne(g,x+200-2*i+1,y-i,x+200-2*i+1,y,rouge);
        i=i+1;
    }
}
```

- faire la **trace** de l'algorithme, c'est-à-dire l'essayer sur un **exemple** et voir ce que valent **toutes les variables à chaque étape**

Quel exemple ?

`DessineToit(300,200)`

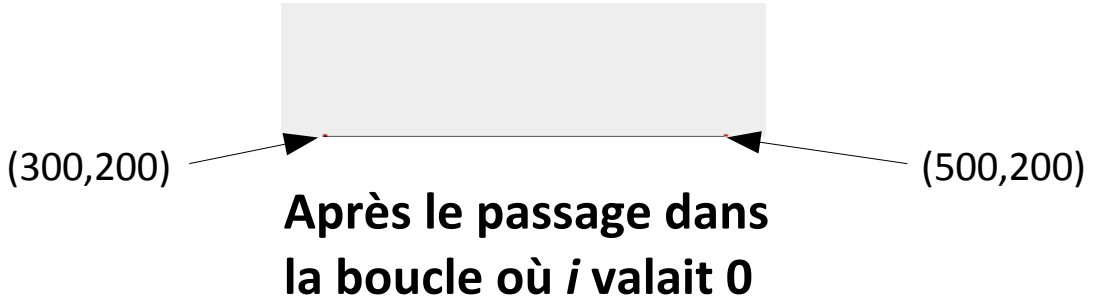
Comprendre un algorithme

```
public static void DessineToit(int x, int y, Graphics g){
    int i=0;
    Color rouge=couleurRGB(200,0,0);
    while (i<51){
        dessineLigne(g,x+2*i,y-i,x+2*i,y,rouge);
        dessineLigne(g,x+2*i+1,y-i,x+2*i+1,y,rouge);
        dessineLigne(g,x+200-2*i,y-i,x+200-2*i,y,rouge);
        dessineLigne(g,x+200-2*i+1,y-i,x+200-2*i+1,y,rouge);
        i=i+1;
    }
}
```

- faire la **trace** de l'algorithme, c'est-à-dire l'essayer sur un **exemple** et voir ce que valent **toutes les variables à chaque étape**

Quel exemple ?

`DessineToit(300,200)`



Comprendre un algorithme

```
public static void DessineToit(int x, int y, Graphics g){
    int i=0;
    Color rouge=couleurRGB(200,0,0);
    while (i<51){
        dessineLigne(g,x+2*i,y-i,x+2*i,y,rouge);
        dessineLigne(g,x+2*i+1,y-i,x+2*i+1,y,rouge);
        dessineLigne(g,x+200-2*i,y-i,x+200-2*i,y,rouge);
        dessineLigne(g,x+200-2*i+1,y-i,x+200-2*i+1,y,rouge);
        i=i+1;
    }
}
```

- faire la **trace** de l'algorithme, c'est-à-dire l'essayer sur un **exemple** et voir ce que valent **toutes les variables à chaque étape**

Quel exemple ?

`DessineToit(300,200)`

Après le passage dans la boucle où *i* valait 10

Comprendre un algorithme

```
public static void DessineToit(int x, int y, Graphics g){
    int i=0;
    Color rouge=couleurRGB(200,0,0);
    while (i<51){
        dessineLigne(g,x+2*i,y-i,x+2*i,y,rouge);
        dessineLigne(g,x+2*i+1,y-i,x+2*i+1,y,rouge);
        dessineLigne(g,x+200-2*i,y-i,x+200-2*i,y,rouge);
        dessineLigne(g,x+200-2*i+1,y-i,x+200-2*i+1,y,rouge);
        i=i+1;
    }
}
```

- faire la **trace** de l'algorithme, c'est-à-dire l'essayer sur un **exemple** et voir ce que valent **toutes les variables à chaque étape**

Quel exemple ?

`DessineToit(300,200)`

Après le passage dans la boucle où *i* valait 50

Le toit avec une pente à 26,565°

```
public static void DessineToit(int x, int y, Graphics g){
    int i=0;
    Color rouge=couleurRGB(200,0,0);
    while (i<51){
        dessineLigne(g,x+2*i,y-i,x+2*i,y,rouge);
        dessineLigne(g,x+2*i+1,y-i,x+2*i+1,y,rouge);
        dessineLigne(g,x+200-2*i,y-i,x+200-2*i,y,rouge);
        dessineLigne(g,x+200-2*i+1,y-i,x+200-2*i+1,y,rouge);
        i=i+1;
    }
}
```

Traduction en pseudo-code :

Algorithme **DessineToit**
Variables d'entrée : entiers *x* et *y*
Variables : entier *i*, couleur *rouge*
Début
    i ← 0
    rouge ← **couleurRGB**(200,0,0)
    Tant que *i*<51 faire :
        dessineLigne(x+2*i*,y-*i*,x+2*i*,y,rouge)
        dessineLigne(x+1+2*i*,y-*i*,x+1+2*i*,y,rouge)
        dessineLigne(x+200-2*i*,y-*i*,x+200-2*i*,y,rouge)
        dessineLigne(x+201-2*i*,y-*i*,x+201-2*i*,y,rouge)
        i ← *i*+1
    Fin TantQue
Fin

Plan du cours 5 – Méthodologie pour l'algorithmique

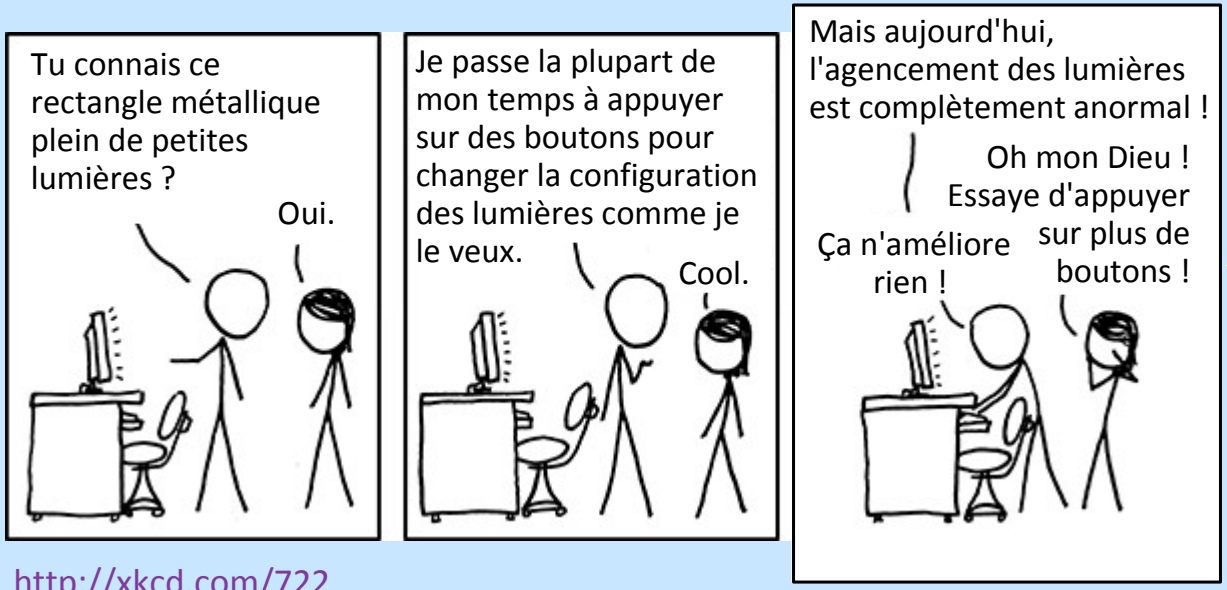
- Résumé de l'épisode précédent
- Compétences acquises en INF120
- Comprendre un algorithme
- Concevoir un algorithme
- Vérifier un algorithme

Méthodologie pour l'algorithmique - concevoir

Savoir concevoir un algorithme pour résoudre un problème

La "minute xkcd"

Problèmes informatiques



C'est comme ceci que j'explique mes problèmes d'ordinateur à mon chat. D'habitude, il a l'air plus content que moi.

http://xkcd.com/722
http://xkcd.free.fr?id=722

Méthodologie pour l'algorithmique - concevoir

Savoir concevoir un algorithme pour résoudre un problème

Premiers éléments à identifier :
- quels sont les **outils à disposition ?** (pour ces outils : données en entrée, type de données en entrée, résultat en sortie, type de résultat en sortie, résultat attendu sur un exemple...)
- quel est le **comportement attendu** pour mon algorithme ? (données en entrée, type de données en entrée, résultat en sortie, type de résultat en sortie, résultat attendu sur un exemple...)

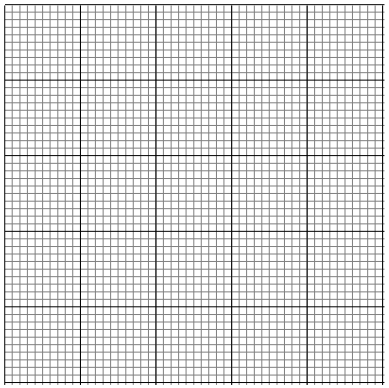
Ensuite, résoudre le problème en utilisant ces outils :
- comment résoudre le problème **étape par étape** ? (essayer sur l'exemple testé)
- est-ce que les **outils à disposition** sont **utilisables** pour réaliser chaque étape ?

Enfin :
- comment **structurer** l'utilisation des outils à disposition ? (**combinaison** des différents outils à l'intérieur de structure de **boucles**, de **tests**, utilisation d'un **organigramme**...)
- comment **décomposer** le problème ? (et **reformuler** chaque sous-problème pour le résoudre avec les outils à disposition, écrire un algorithme par sous-problème)

Méthodologie pour l'algorithmique - concevoir

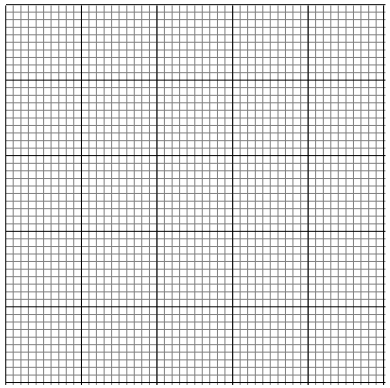
Concevoir un algorithme qui dessine du papier millimétré

Outils disponibles : DessineLigne et CouleurRGB



Méthodologie pour l'algorithmique - concevoir

Concevoir un algorithme qui dessine du papier millimétré



Outils disponibles : DessineLigne et CouleurRGB

Entrées :

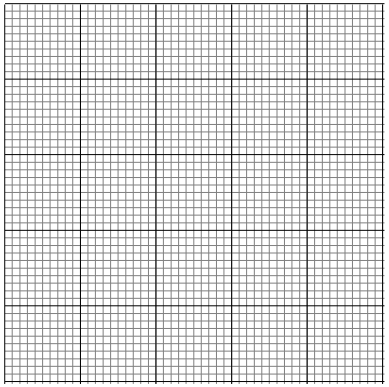
Sortie :

Comportement attendu illustré dans cet exemple (5 grandes cases)

Décomposition du problème :

Méthodologie pour l'algorithmique - concevoir

Concevoir un algorithme qui dessine du papier millimétré



Outils disponibles : DessineLigne et CouleurRGB

Entrées : couleur des grandes cases, couleur des petites cases, largeur et hauteur des petites cases (en pixels), nombre de petites cases, nombre de grandes cases.

Sortie : aucune sortie, affichage sur la fenêtre à l'écran

Comportement attendu illustré dans cet exemple (5 grandes cases)

Décomposition du problème : dessiner d'abord la petite grille grise plus la grande grille noire, tous deux avec le même algorithme **DessineGrille**.

Entrées de DessineGrille : couleur *couleurGrille*, entiers *taille* et *nbCases*

Idée de l'algorithme DessineGrille : 2 boucles, une pour dessiner les lignes verticales, une pour dessiner les lignes horizontales.

Plan du cours 5 – Méthodologie pour l'algorithmique

- Résumé de l'épisode précédent
- Compétences acquises en INF120
- Comprendre un algorithme
- Concevoir un algorithme
- Vérifier un algorithme

Méthodologie pour l'algorithmique - vérifier

Se reporter à la fiche distribuée :

Fiche d'auto-correction des algorithmes en pseudo-code

Voici un ensemble de questions que vous devez vous poser après avoir écrit un algorithme en pseudo-code (en Java, c'est pareil, il vous suffit de "traduire" les questions ci-dessous en Java).

Vous devriez toujours répondre oui, sinon cela signifie qu'il y a un oubli ou une erreur à corriger.

Déclaration de l'algorithme

Ai-je déclaré le nom de l'algorithme ?

Ai-je déclaré les entrées de l'algorithme ? Ai-je utilisé des noms de variables corrects (commencent par une lettre, pas de caractères spéciaux ni d'espaces) ? Ai-je précisé leur type (s'il s'agit de tableaux, ai-je précisé quel type d'éléments sont stockés dans le tableau ?) ? Correspondent-elles aux entrées indiquées par l'énoncé ?

Ai-je déclaré le type de sortie de l'algorithme ?

Ai-je déclaré les variables de l'algorithme ? Ai-je utilisé des noms de variables corrects (commencent par une lettre, pas de caractères spéciaux ni d'espaces) ? Ai-je précisé leur type ? Ai-je pensé à ne pas redéclarer les variables d'entrée, mais à déclarer l'éventuelle variable de sortie ? Sont-elles utilisées dans l'algorithme, c'est-à-dire entre **Début** et **Fin** ?

Est-ce qu'aucun nom d'algorithme n'apparaît parmi les variables d'entrée et les variables déclarées ?

Instructions de l'algorithme

Pour chaque mot qui apparaît à l'intérieur du corps de l'algorithme, c'est-à-dire entre **Début** et **Fin** :
- S'il est entouré de guillemets, il fait partie d'une chaîne de caractères, ça peut donc être n'importe quel mot.
- Sinon :
 * Si c'est un nom de type en pseudo-code (entier, flottant, booléen, chaîne de caractères, couleur, tableau de ...), ça n'a rien à faire là (les noms de types sont présents uniquement dans les déclarations).
 * Si c'est un élément de langage du pseudo-code (**Pour**, **Tant que**, **Si**, **faire**, **alors**, **Fin pour**, **Fin Tant que**, **renvoyer**, **Case**, **Nouveau Tableau**, **Affiche**) ai-je bien respecté la syntaxe d'utilisation de cet élément du langage (vérifier dans le glossaire) ?
 * Si c'est le nom d'un algorithme (= d'une fonction), il s'agit d'un appel d'algorithme : l'algorithme a-t-il été déclaré auparavant, ou est-il connu d'après l'énoncé ? Ai-je mis entre parenthèses des entrées correctes (bon type, bon nombre d'entrées), séparées par des virgules ? Si l'algorithme renvoie un résultat en sortie, est-ce que je le récupère par une affectation dans une variable (variable du même type que le type de sortie de l'algorithme ?), ou comme entrée d'un autre algorithme (entrée de l'autre algorithme de même type que le type de sortie de l'algorithme ?) ?
 * Si c'est le nom d'une variable, a-t-elle été déclarée parmi les variables ou les variables d'entrée ? De plus :
  → Si elle est à gauche d'une flèche d'affectation, est-ce que son type est le même que ce qui est à droite de la flèche d'affectation ?
  → Sinon, a-t-elle été initialisée précédemment ? S'il s'agit d'une case d'un tableau, est-ce qu'elle existe, et est-elle remplie ? Est-ce que la valeur de cette variable est utilisée d'une manière correcte, soit comme entrée d'un algorithme, soit dans un calcul effectué avec des opérations de base (par exemple l'addition ou la multiplication d'entiers, la concaténation de chaînes de caractères, etc.) ?