TP04

OBJECTIF

Découverte des algorithmes dit de "décomposition"

INTRODUCTION

Ecrire un algorithme c'est la solution à un problème posé. Cette solution n'est pas facile à trouver car pour chaque problème une solution différente, voire plusieurs solutions. Et comme il n'y a jamais le même problème il n'y a quasiment jamais les mêmes solutions. Cela peut être particulièrement déroutant quand on débute.

Une bonne façon d'appréhender l'algorithmique avant de se décourager est de poser les problèmes sur papier et de ne pas se lancer trop rapidement dans la phase de codage. Voici quelques conseils :

1. Bien lire le sujet en le reformulant sur papier si besoin. L'objectif est ici de bien comprendre le sujet. Noter les points importants.
2. Ecrire le dictionnaire de données faisant apparaitre les entrées et sorties su programme. Si le sujet le précise, noter dans ce dictionnaire les valeurs possibles de ses données, des exemples de résultats.
3. Sans écrire encore l'algorithme essayer de représenter visuellement vos données (sous forme de boite par exemple pour représenter une variable) ou tout autres moyens graphiques. Avec ce visuel prenez des valeurs exemples pour vos données et tenter de résoudre le problème à la main. La solution ressemble très souvent à ce que vous avez tendance à faire automatiquement à la main.
   Essayer plusieurs valeurs exemple pour vérifier que votre raisonnement marche à chaque fois. Essayer ensuite des valeurs exemples qui sont particulières (0, nombre négatif, autres…)
4. Ecrire ensuite l'algorithme en vous servant du travail précédent. Améliorer ce dernier si possible.

APPRENDRE A LIRE UN ENNONCE

Un algorithme doit fonctionner quelques soit les valeurs en entrées. C'est pour cela que les énoncés seront exprimés comme dans les exemples suivants :

Exemple 1 : Encadrer la racine carrée d'un nombre réel positif ***X*** par deux entiers successifs.
Exemple 2 : Alice à ***N*** bonbons. Elle en donne ***P*** à Bob. Calculer combien il reste de bonbons à Alice.
Exemple 3 : Trouver le plus court chemin entre deux adresses ***A*** et ***B***.

Reprenons l'exemple 1. On parle d'un nombre réel positif **X**. **X** peut prendre une infinité de **valeurs autorisés** respectant les contraintes de l'énoncé :

- X > 0
- X est un réel

Quelques exemples de valeurs possibles pour X :

12.6	15	18.965	…

Cette valeur ne doit pas être codé en « dur » dans le programme. Sinon on est en train de réaliser un programme spécifique pour une et une seule valeur de X. Le programme que nous allons écrire doit marcher quel que soit la valeur de X. On le comprend facilement ici X sera une entrée saisie au clavier. Ainsi le programme fonctionnera pour toutes valeurs de X.

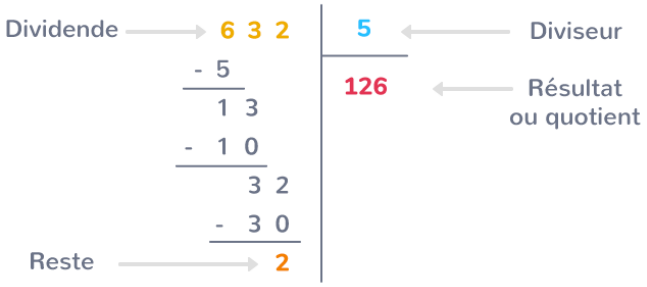
Ensuite, à la lecture de l'énoncé il faut identifier les **valeurs particulières** de X. Par exemple si X est un carrée parfait c'est une valeur particulière qui nécessite un **traitement particulier**. Ex X = 144 = 12^2 ou X =25 = 5^2 etc …

Enfin, il faut identifier les valeurs non autorisées pour X. Ici l'énoncé est assez claire. Toute valeur < ou = à 0 pour X sera interdite

IDENTIFIER LES TYPES DE PROBLEMES

Chaque problème est différent, donc chaque solution algorithmique sera différente. Cependant il existe tout de même des "familles" d'algorithme. Pour dégrossir le travail il faut savoir identifier ses problèmes types. Nous allons aborder dans ce TP les problèmes dit de **décomposition de nombres entier.**

Ce type de problème consiste à décomposer un nombre entier en sous-nombres. Cela s'avère très utile dans de nombreux cas. La technique permettant de décomposer un nombre en sous nombre est d'utiliser la division d'entier ou euclidienne, celle que l'on apprend en primaire !



La division répond à l'équation A=B x Q + R

Avec $0 \leq R < B$

- **A** le dividende (dans l'exemple 632),
- **B** le diviseur (5),
- **Q** le quotient (126),
- **R** le reste (2)

Le langage Java propose 2 opérateurs :

- / : permet d'obtenir le quotient Q d'une division de A par B, ou A et B sont 2 entiers
- % : permet d'obtenir le reste R de la division de A par B, ou A et B sont 2 entiers

EXERCICE 1 : GERER LE TEMPS

Java propose de nombreuses fonctions pour gérer le temps.

Coder le programme suivant :

```
long timeMillis =System.currentTimeMillis();
System.out.println("time = " + timeMillis);
```

En vous aidant des différentes documentations préciser à quoi correspond l'affichage obtenu.

https://docs.oracle.com/javase/8/docs/api/java/lang/System.html#currentTimeMillis--

https://koor.fr/Java/API/java/lang/System/currentTimeMillis.wp

La cérémonie d'ouverture des jeux olympiques 2024 qui aura lui en France débutera le vendredi 26 juillet 2024 à 20 : 24 précisément, heure locale, ce qui correspond en millisecondes à la valeur 1722025440000 ms

Voir : https://www.paris2024.org/fr/ceremonie/

Ecrire un programme permettant d'afficher le nombre de jours, heures, minutes et secondes nous séparant de cette cérémonie.

Exemple d'affichage possible :

```
Les jeux olympiques de Paris débute dans 284 jours 13 heures  38 minutes 49 secondes
```

EXERCICE 2 : CODE POSTALE

Ecrire un algorithme permettant de connaitre le numéro de département en fonction d'un code postale, ainsi que le bureau de distribution. Par exemple pour le code postale 26520, le département est 26, le bureau de distribution postale est le 520.

https://www.code-postal.com/

EXERCICE 3 : RENDRE LA MONAIE

Un commerçant veut une application lui permettant de rendre la monnaie de façon simple. Dans cet exercice nous manipulerons uniquement des euros, pas de centimes. Le commerçant ne peut rendre que des billets de 20€, 10€, 5€, des pièces de 2€ et 1€.

Exemple : un client donne un billet de 50€ pour payer un article qui en vaut 18€. L'application affichera :

```
Vous devez rendre 1 billet de 20€, 1 billet de 10€, une pièce de 2€.
```

EXERCICE 4 : LES CHANGEMENTS DE BASES

Ecrire un algorithme permettant de convertir un nombre binaire (sur 4 bits uniquement) en sa valeur décimal.

Par exemple pour le nombre 1011 le programme affichera :

```
1101 en binaire vaut 13 en décimal.
```

Attention à faire les vérifications qui s'imposent !