Séquence 0 – Les bases de l’ algorithmique

Objectifs

1. Bases de l’ algorithmique
2. Les différents structures algorithmiques
3. Notion de coût
4. Diversité et unité des langages de programmation

1 Algorithme : Une suite finie d’ instructions

« Un **algorithme** est une suite finie (...) d'instructions et d’opérations permettant de résoudre une classe de problèmes » Source Wikipedia

A faire vous même 1.

À l’eau les mains ! Difficulté ★★★

Voici un algorithme.

Ouvrir le robinet
Se remonter les manches
Mouiller ses mains
Mettre du savon dans ses mains
Frotter longtemps
Rincer ses mains
Sécher ses mains avec une serviette
Fermer le robinet

1. Expliquer brièvement ce que permet de faire exactement cet algorithme.

2. Écrire un algorithme qui fasse la même chose, mais qui soit plus respectueux de la planète en économisant au maximum l’eau utilisée.

2 Les differentes structures algorithmiques

2.1 Les boucles

Quand il est nécessaire de répéter un même bloc d’ instructions plusieurs fois, alors il est possible d’ utiliser une boucle.

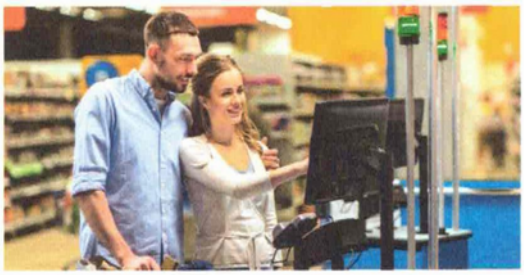
A faire vous même 2.

Les caisses automatiques

Dans les supermarchés actuels, on trouve de plus en plus de caisses automatiques qui remplacent progressivement les hôte(sse)s de caisse. Pour rendre la monnaie aux clients, ces automates en libre-service suivent et produisent des algorithmes.

1. À l'aide de boucles et des deux instructions : **Donner une pièce de 1€** et **Donner une pièce de 20 cents**, à n'utiliser qu'une seule fois chacune, écrire un algorithme qui permette de rendre 3,80 € au client.
2. Le distributeur peut donner des pièces de 2 €, 1 €, 50 cents, 20 cents, 10 cents, 5 cents, 2 cents ou encore 1 cent.

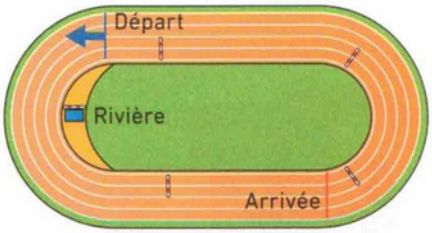
Écrire un algorithme contenant des boucles qui permette de rendre 9,48 € en donnant le moins de pièces possible au client.



A faire vous même 3.

Le 3 000 mètres steeple

Le 3 000 mètres steeple est une épreuve d'athlétisme assez spectaculaire : les coureurs doivent courir 3 000 m, soit 7 tours et demi de stade, sur lesquels sont disposées 5 barrières espacées de 80 m et dont l'une d'elle est suivie d'une « rivière ». Les premiers 225 m s'effectuent sans croiser de barrières : les juges les rajoutent après le passage des sportifs.



À l'aide de boucles et en utilisant les instructions ci-dessous, compléter l'algorithme ci-contre qui doit décrire de la façon la plus courte possible le parcours effectué par les coureurs.

Courir ... mètres
Sauter une barrière
Sauter la rivière

Prendre le départ de la course

Franchir la ligne d'arrivée

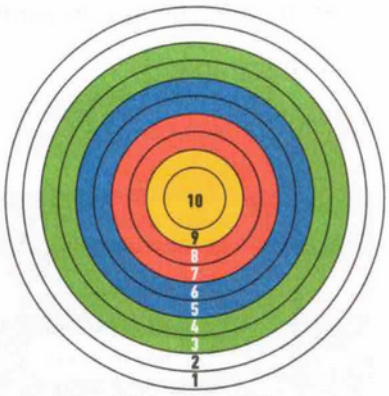
2.2 Les variables

Une variable est une étiquette/une boite/une case dans laquelle on va pouvoir stocker un contenu (un nombre entier, un nombre décimal, du texte, …).

A faire vous même 4.

Le tir à l'arc

Un participant à une épreuve de tir à l'arc tire 5 flèches sur une cible avec des zones valant de 1 à 10 points. Le nombre total de points du joueur (*score total*) augmente donc à chaque flèche en ajoutant la valeur du nouveau tir (*valeur fleche*).
À l'aide d'une boucle et de deux variables *score total* et *valeur fleche*, compléter l'algorithme décrivant le déroulement d'une épreuve de tir à l'arc et permettant de calculer au fur et à mesure le score du tireur, puis d'annoncer son score final.



Mettre *score total* à .....

Répéter ..... fois

.....

.....

.....

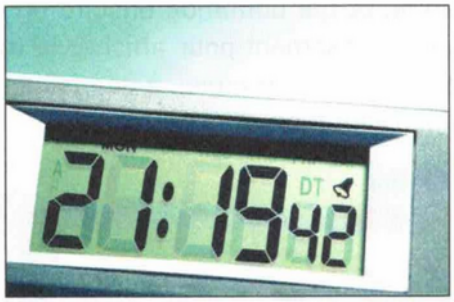
Annoncer .....

A faire vous même 5.

L'horloge digitale

Pour faire fonctionner une horloge digitale, il y a trois variables : *heures*, *minutes* et *secondes*.
Voici ci-contre un morceau de l'algorithme qui permet de gérer l'affichage de l'heure qu'il est (de 0 à 24 heures).
Compléter cet algorithme pour qu'il permette d'afficher les heures, minutes et secondes sur une journée complète (soit de 24 heures).

Répéter 60 fois
  Afficher *secondes*
  Ajouter 1 à *secondes*
Mettre *secondes* à 0
Ajouter 1 à *minutes*



2.3 Les instructions conditionnelles

Dans un algorithme, suivant une condition, il est possible d' exécuter un bloc d' instruction ou un autre.

A faire vous même 6.

What time is it?

Les Anglais utilisent les abréviations AM (*ante meridiem :* avant le midi) pour désigner l'heure du matin et PM (*post meridiem :* après le midi) pour celle de l'après-midi, en écrivant toujours un nombre d'heures entre 1 et 12 (inclus). Ainsi, 16 h 30 à Paris s'écrit 4:30 PM à Londres.
On souhaite construire un algorithme qui demande l'heure en écriture française (en utilisant deux variables : *heures* et *minutes*) et affiche l'heure au format anglais.
Voici un morceau de l'algorithme qui permet de faire cette conversion :

Si *heures* = 0 **Alors** Dire « 12 : » et *minutes* et « AM »
Si *heures* > 12 **Alors** Ajouter –12 à *heures*
Dire *heures* et « : » et *minutes* et « PM »
Sinon

Compléter cet algorithme en utilisant éventuellement le morceau précédent.

Demander le nombre d'heures et attendre

A faire vous même 7.

La séance de tirs au but

Lors de certains matchs de football, en cas d'égalité à la fin du temps réglementaire, on procède à une séance de tirs au but (5 par équipe) pour désigner le vainqueur. Si les deux équipes sont encore à égalité à la fin de ces tirs au but, on continue alors en « mort-subite », c'est-à-dire jusqu'à ce qu'une équipe marque et l'autre non. À l'aide d'instructions conditionnelles et de deux variables ***buts-A*** et ***buts-B***, compléter l'algorithme décrivant le déroulement de cette « mort-subite ».

Répéter jusqu'à ***buts-A*** différent ***buts-B***

2.4 Les blocs d' instructions

Quand on a besoin d' utiliser plusieurs fois un ensemble d' instructions, on créé un bloc.

A faire vous même 8.

Le 4 × 100 mètres nage libre

Une épreuve de natation olympique pour laquelle l'équipe de France masculine est parmi les meilleures du monde est le 4 × 100 mètres nage libre. Cette épreuve de relais est disputée dans une piscine de 50 m de longueur par des équipes de 4 nageurs, qui nagent chacun leur tour un aller-retour. Décrire cette course par un programme court qui utilise le bloc défini ci-contre.

Définir **Nager son 100 mètres**
Plonger dans la piscine
Nager 50 m en ligne droite
Toucher le bord de la piscine
Faire demi-tour
Nager 50 m en ligne droite
Toucher le bord de la piscine

Premier nageur sur le plot de départ

.....

.....

Nager son 100 mètres

Nageur suivant sur le plot de départ

.....

Arrêter le chronomètre

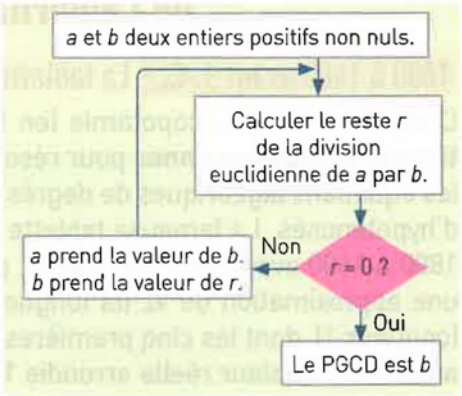
3 Un algorithme de recherche du PGCD

L'algorithme de calcul du PGCD (Plus Grand Commun Diviseur) de deux entiers positifs non nuls a été proposé par le mathématicien grec Euclide. L'organigramme est présenté ci-contre.

Le PGCD est très utile pour simplifier une fraction.

Par exemple, PGCD(48, 36) = 12 :

$$\frac{36}{48} = \frac{3 \times 3 \times 4}{2 \times 2 \times 3 \times 4} = \frac{3 \times \cancel{12}}{2 \times 2 \times \cancel{12}} = \frac{3}{4}$$



1. En vous aidant de l'organigramme, calculer le PGCD de 782 et 221 par la méthode d'Euclide. Compléter le tableau ci-contre en faisant figurer toutes les valeurs intermédiaires prises par les trois variables *a*, *b* et *r*. Il est possible d'utiliser l'opérateur Python % dans la console Python (`a % b` renvoie le reste de la division euclidienne de *a* par *b*).

a	*b*	*r*
782	221	119
.....	.....	.....
.....	.....	.....
.....	.....	.....

2. En observant la structure de l'organigramme ci-dessus, justifier le caractère itératif de cet algorithme. Qu'apportent ces itérations ?

3. En vous aidant du document ci-contre, traduire cet organigramme en un algorithme écrit en pseudo-code.

Comment écrire un algorithme en pseudo-code ?

Le **pseudo-code** est une façon de décrire un algorithme en langage presque naturel, sans référence à un langage de programmation en particulier. Voici quelques conventions retenues dans le cahier :

- Préciser les spécifications en dehors de l'algorithme. Par exemple :
En entrée : deux nombres entiers positifs *a* et *b* non nuls.
En sortie : renvoie le PGCD de *a* et de *b*.
- Indenter le bloc d'instructions d'une boucle ou d'une fonction.
- Utiliser le symbole ← pour l'affectation.
- Les tableaux commencent à 0. On accède à l'élément d'indice *i* avec `tableau[i]`.
- Placer les arguments d'une fonction entre parenthèses.
- Utiliser des noms de variable ou de fonction explicites.
- Ne mettre que des instructions génériques, indépendantes de tout langage de programmation.

4 Performance et coût d' un algorithme

Certains algorithmes de cryptographie utilisent la multiplication de très grands nombres entiers. Ils nécessitent donc l'utilisation d'algorithmes de calcul performants. On s'intéresse ici à deux algorithmes de calcul de x^n, x étant un nombre réel et n étant un entier positif.

L'algorithme pour résoudre un problème mathématique n'est pas unique. Pour déterminer quel algorithme choisir parmi plusieurs possibles, une des méthodes consiste à évaluer le coût des algorithmes en termes de temps de calcul, en estimant le nombre d'opérations élémentaires effectuées ou le nombre de tours de boucle. Ce coût est appelé **complexité temporelle**.

On appelle **opération élémentaire** toute opération d'affectation de variable, de calcul (addition, soustraction, division, multiplication) ou encore de comparaison entre deux valeurs. On suppose que les instructions sont exécutées l'une après l'autre, sans opérations simultanées.

1. Dérouler chacun des algorithmes en complétant les deux tableaux de variables pour $x = 2$ et $n = 5$.

◀ Algorithme 1

```
Algorithme : puissance 1 (x, n)
p ← 1
pour i allant de 1 à n
        p ← p * x
renvoyer p
```

Variable	i	p
Initialisation	Non définie	1
1er tour	1	2
2e tour	2	4
3e tour	3	8
4e tour	4	16
5e tour	5	32

Algorithme 2 ▶

```
Algorithme : puissance 2 (x, n)
p ← 1
tant que n > 0
        si n est impair alors
            p ← p * x
        x ← x * x
        n ← n // 2
renvoyer p
```

Variable	p	x	n
Initialisation	1	2	5
1er tour	2	4	2
2e tour	2	16	1
3e tour	32	256	0

2. Quel est le point commun essentiel entre ces deux algorithmes ?

3. Pour simplifier l'étude de la complexité, on ne s'intéresse qu'au nombre de tours de boucle. Compléter le tableau suivant en indiquant le nombre de tours de boucle nécessaire au calcul.

Nombre de tours de boucle	Algorithme 1	Algorithme 2
Calcul de 2^5	5 tours	3 tours
Calcul de $2^{5\,000}$	5 000 tours	13 tours
Calcul de 2^n	n tours	environ $\log_2(n+1)$ tours

Pour remplir la dernière colonne, il suffit de s'intéresser aux affectations successives de la variable n.

4. Aller sur lycee.editions-bordas.fr/cahier-NSI1re, puis, dans « Séquence 9 », cliquer sur l'interface « Complexité ». Afficher uniquement les courbes des deux fonctions $f(n) = n$ et $g(n) = \log_2(n)$ (précision : $\log_2(n) = \frac{\log(n)}{\log(2)}$). Quelle est celle qui domine l'autre, et pour quelles valeurs de n ?

La fonction $f(n) = n$ domine sur $\log_2(n)$ pour toutes les valeurs de n.

5. Que peut-on supposer pour les valeurs des temps d'exécution des algorithmes 1 et 2 de la question 1 ? Conclure en précisant quel algorithme choisir pour calculer x^n, si n est grand.

On peut supposer que le temps d'exécution de l'algorithme 2 sera beaucoup plus faible que celui de l'algorithme 1. L'algorithme 2 doit être choisi pour calculer x^n car sa complexité est favorable.