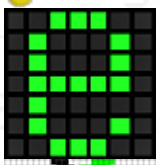
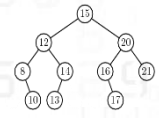
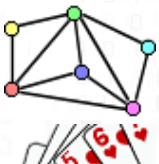
DÉFIS-PROGRAMMATION

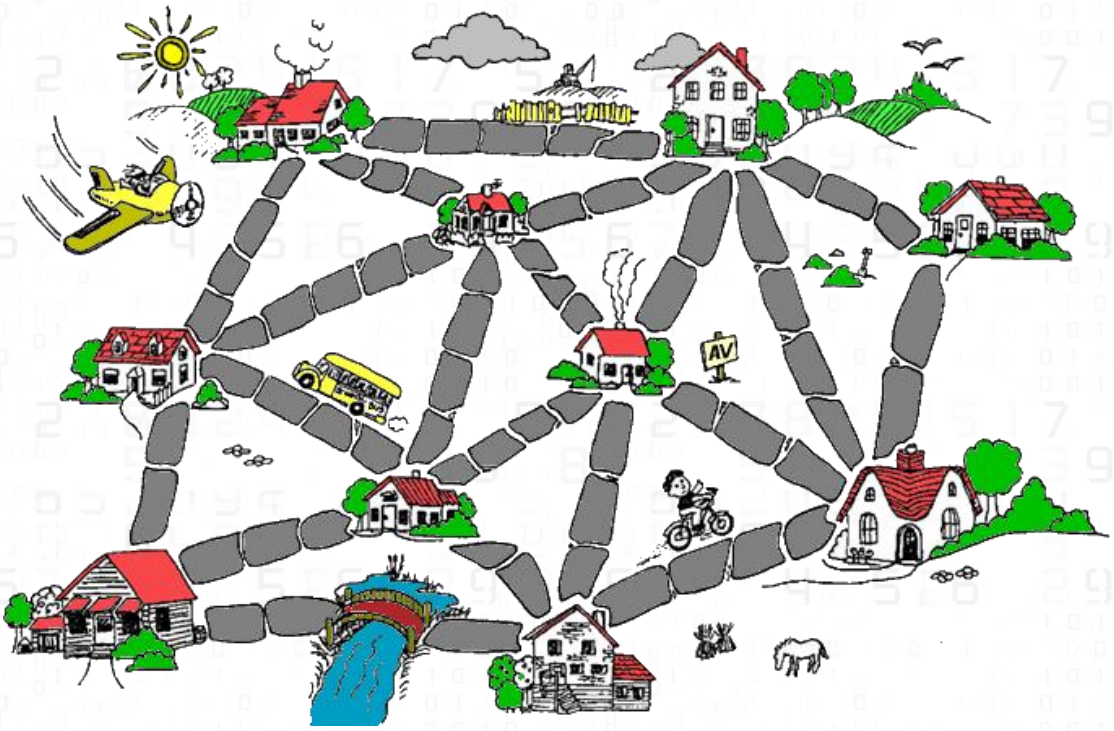
Circonscription du Chapus







SOMMAIRE

Présentation

A l'école primaire, les élèves découvrent l'informatique à travers l'utilisation de logiciels existants qui permettent d'effectuer des tâches précises : recherche sur Internet, envoi de courriers électroniques, rédaction de textes, etc. Cet apprentissage conduit peu à peu à la maîtrise des compétences répertoriées dans le B2i, reprises dans la compétence 4 du socle commun. Ces compétences sont réparties en 5 domaines :

1. Maîtriser l'environnement informatique.
2. Adopter une attitude responsable.
3. Produire et traiter des documents.
4. S'informer, se documenter.
5. Communiquer, échanger.

Les programmes de 2016 ont ajouté :

- L'apprentissage du code informatique.
- Les algorithmes.

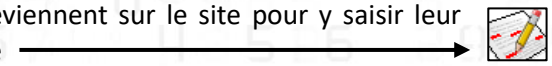
Il ne s'agit pas là pour nos élèves de devenir des programmeurs chevronnés, juste de découvrir quelques concepts fondamentaux indépendants des machines et des logiciels utilisés. Cette initiation peut se faire de trois façons complémentaires :

- *Sans ordinateur*, au travers d'activités fortement liées aux disciplines fondamentales (lecture, mathématiques). On peut aborder au cycle 3, de façon ludique, les concepts élémentaires qui sont à la base de l'informatique, tout en mobilisant des compétences acquises dans les autres domaines d'enseignement. En voici une liste non exhaustive :
  - Les informations et leurs représentations (textes, images).
  - Leur mémorisation sous forme binaire.
  - Le stockage et son impact sur l'environnement.
  - La recherche et le tri des données.
  - La détection des erreurs.
  - L'utilisation d'arbres et de graphes pour résoudre des problèmes.
- *Avec un ordinateur ou une tablette*, à l'aide de logiciels dédiés qui permettent d'aborder :
  - Les déplacements programmés sur quadrillage (Lightbot, Tuxbot, …)
  - Les tracés géométriques (Tortue LOGO, Geogebra, …)
- *A l'aide de robots éducatifs* programmables, qui permettent de s'affranchir de l'écran et d'entrer dans le monde réel (Les défis-programmation n'utilisent pas de robot, juste du papier et un crayon, et quelques logiciels simples dans les derniers défis).

Par ailleurs, la notion d'*algorithme* est une formidable opportunité de faire le lien entre l'informatique, les autres domaines enseignés et les situations vécues par les élèves. Un algorithme n'est qu'une suite d'instructions permettant d'obtenir un résultat attendu. Les techniques opératoires, les règles de conjugaison, la lecture d'un code-barres à la caisse du supermarché, la réalisation d'une recette de cuisine, et bien d'autres actions, font appel à des algorithmes relativement simples et compréhensibles par des enfants d'âge scolaire.

Fonctionnement

Les Défis-programmation proposent d'aborder les notions de codage et d'algorithme de façon simple, ludique et progressive :

1. Toutes les 2 semaines, les élèves se connectent au site et découvrent un nouveau thème avec un problème à résoudre. La difficulté des situations proposées augmente progressivement tout au long de l'année scolaire.
2. La classe s'organise pour trouver une solution puis les élèves reviennent sur le site pour y saisir leur réponse et demander la correction en cliquant sur le bouton dédié 
3. La classe est avertie du résultat par courriel. Si le défi n'a pas été validé, les élèves peuvent modifier leur réponse et solliciter une nouvelle correction autant de fois que nécessaire. Il n'y a pas de date limite à respecter.
4. Un bilan permet à chaque classe de suivre sa progression tout au long de l'année scolaire.

Vous pouvez inscrire votre classe à tout moment sur https://cerp-lechapus.net/defis_programmation/ en cliquant sur « Inscription » et en complétant le formulaire.

Liste des défis proposés

N°	Titre	Objectif	Compétences travaillées
1	Des zéros et des uns	– Comprendre le système binaire utilisé en informatique	– Convertir des nombres du système binaire vers le système décimal et inversement – Calculer mentalement : additions, soustractions et multiplications
2	Communication lumineuse		
3	Du noir et du vert	– Découverte du principe d'affichage des images numériques en noir & blanc	
4	Le monde des couleurs	– Découverte du principe d'affichage des images numériques en couleur	– Se repérer sur un quadrillage – Coder/décoder une image numérique
5	On manque de place	– Découverte l'impact du stockage des données sur l'environnement – Découverte d'un algorithme de compression des données	– Appliquer un algorithme simple – Coder/décoder une image numérique
6	Rétrécir un texte	– Découverte d'un algorithme de compression à dictionnaire	– Appliquer un algorithme simple – Coder/décoder un texte
7	Attention aux erreurs !	– Découverte d'un algorithme de détection des erreurs	– Appliquer un algorithme de calcul – Multiplier mentalement. – Additionner par écrit.
8	Recherche rapide	– Découverte de l'algorithme de recherche binaire	– Application d'un algorithme de recherche
9	La ville embourbée	– Utilisation d'un graphe pour résoudre un problème – Appréhender les notions de réseau et de connexion à un réseau.	– Schématiser une situation problème sous la forme d'un graphe. – Utiliser ce graphe pour résoudre un problème
10	Range-moi tout ça !	– Découverte d'un algorithme de tri.	– Comparer des nombres et des masses. – Appliquer un algorithme de tri.
11	Fourmi cherche du miel !	– Programmer un déplacement sur quadrillage	– Programmer une série d'instructions pour coder un trajet. – Maîtriser les déplacements relatifs sur un quadrillage.
12	La tortue sui savait dessiner	– Programmer un tracé géométrique	– Identifier une figure géométrique – Décomposer une figure complexe
13	Un café s'il vous plait !	– Programmer un automate	– Programmer une série d'actions et de tests pour faire fonctionner un automate.

http://cerp-lechapus.net/defis_programmation/

christian.vinent@ac-poitiers.fr

Des zéros et des uns

Pour compter et représenter les nombres, nous utilisons *dix* chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. C'est pour cela que notre système de numération est appelé *système décimal*.
Mais dans un ordinateur, le stockage des informations se fait sous forme *binaire* : Binaire car chaque cellule de mémoire, appelée bit, ne peut prendre que *deux* valeurs :

- **1** (le courant passe)
- **0** (le courant ne passe pas).

Pour convertir un nombre décimal en nombre binaire, ou bien un nombre binaire en nombre décimal, on peut utiliser un tableau comme celui-ci :



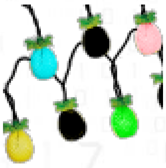


16	8	4	2	1		
0	0	0	0	0	=	0
0	0	0	0	1	=	1
0	0	0	1	0	=	2
0	0	0	1	1	=	3
1	0	1	0	1	=	21
1	1	0	1	0	=	...
...	...	...	...	...	=	14

1. Calculez le nombre du système décimal manquant à l'avant-dernière ligne.

2. Calculez le nombre binaire manquant à la dernière ligne.

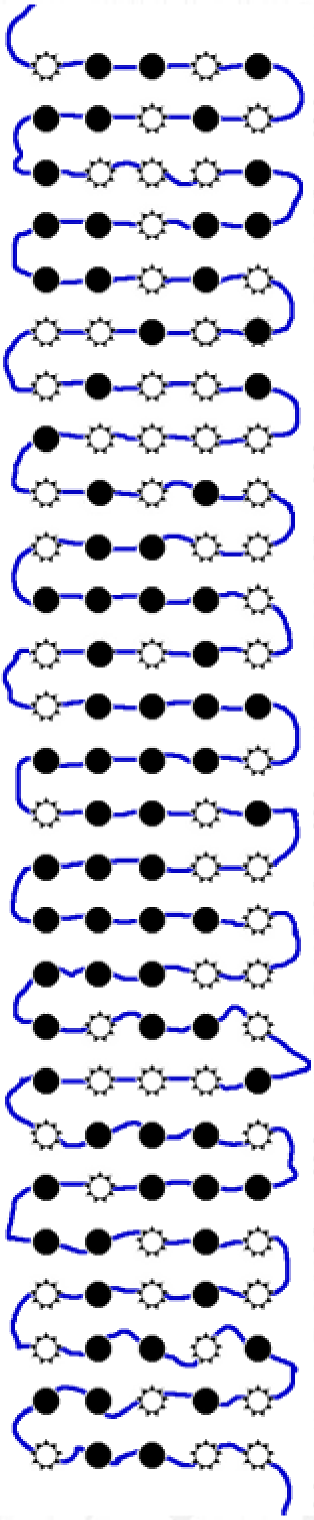
3. Quel est le plus grand nombre du système décimal que l'on puisse coder avec 5 chiffres binaires ?



Communication lumineuse

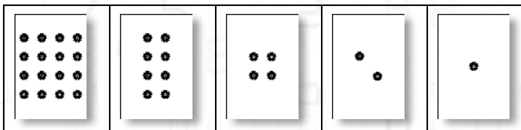
Marc et son ami Pierre habitent dans la même rue. Ils ont mis au point un système pour communiquer en secret lorsqu'ils ne peuvent pas sortir et ne peuvent pas téléphoner. Ils utilisent une guirlande de Noël. Une *ampoule allumée* représente un **1**, une *ampoule éteinte* un **0**.
Grâce au système binaire, et en regroupant les ampoules par 5, ils peuvent coder et afficher un message qu'ils seront les seuls à pouvoir lire. Chaque groupe de 5 ampoules représente un nombre binaire qui, une fois converti dans le système décimal, donne une lettre de l'alphabet.

☼ Désigne une ampoule allumée, ● Désigne une ampoule éteinte.

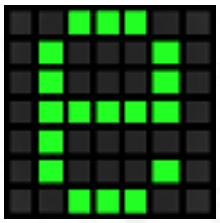


18

Sauras-tu décrypter le message de Marc ?



16	8	4	2	1		
1	0	0	1	0	18	R

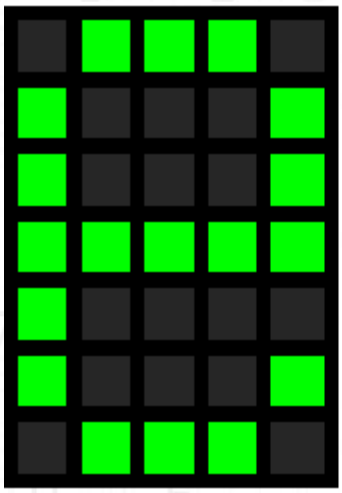


Du noir et du vert

Il fut un temps où les ordinateurs ne pouvaient afficher que du *texte en noir et blanc*. Plutôt que du blanc, on utilisait du vert pour ne pas fatiguer les yeux. L'écran était quadrillé de petits points lumineux appelés pixels. Un pixel pouvait être soit allumé, soit éteint.

- Un *pixel allumé* était codé avec un **1**,
- Un *pixel éteint* avec un **0**.

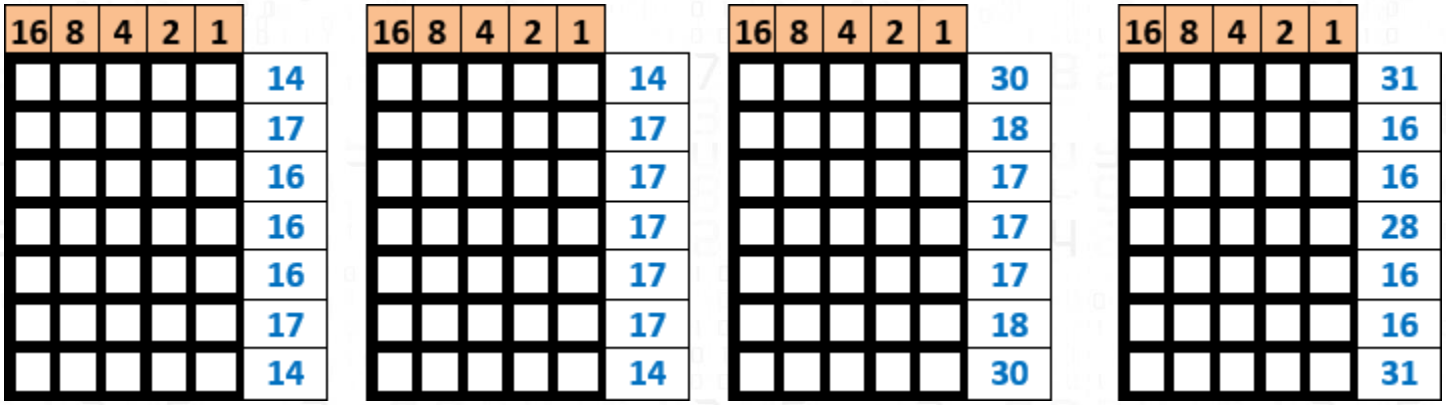
Voici comment un ordinateur représentait à l'écran la lettre e en allumant les pixels correspondants



0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	0
1	0	0	0	1
0	1	1	1	0

16	8	4	2	1	
0	1	1	1	0	14
1	0	0	0	1	17
1	0	0	0	1	17
1	1	1	1	1	31
1	0	0	0	0	16
1	0	0	0	1	17
0	1	1	1	0	14

A vous maintenant de décoder le mot de 4 lettres ci-dessous :



16	8	4	2	1	
					14
					17
					16
					16
					16
					17
					14

16	8	4	2	1	
					14
					17
					17
					17
					17
					17
					14

16	8	4	2	1	
					30
					18
					17
					17
					17
					18
					30

16	8	4	2	1	
					31
					16
					16
					28
					16
					16
					31

Le monde en couleurs !

Coder une image en couleur n'est pas plus compliqué que pour une image en noir et blanc. Il faut par contre utiliser beaucoup plus de bits (0/1) pour chaque pixel. L'image ci-dessous utilise 8 couleurs différentes, numérotées de **0** à **7**. Pour enregistrer un nombre allant de 0 à 7, l'ordinateur devra utiliser 4 bits, comme dans tableau ci-contre.

	Binaire				Couleur
0	0	0	0	0	Noir
1	0	0	0	1	Bleu
2	0	0	1	0	Violet
3	0	0	1	1	Rouge
4	0	1	0	0	Orange
5	0	1	0	1	Jaune
6	0	1	1	0	Vert
7	0	1	1	1	Blanc

Essayons de coder cette petite image de 26 x 26 pixels : 
Cela donnera le tableau de chiffres suivant :

7	7	7	7	7	7	7	7	0	0	0	7	7	7	7	6	6	6	6	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	0	0	0	7	6	6	6	6	6	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	7	0	0	6	6	6	6	6	7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	7	7	0	6	6	6	7	7	7	7	7	7	7	7	7	7	7
7	7	7	7	7	0	0	0	0	0	7	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7
7	7	7	7	0	4	4	4	4	4	0	7	0	7	7	7	0	0	0	0	0	0	7	7	7	7
7	7	7	0	4	4	4	4	4	4	4	0	0	0	0	0	4	4	4	4	4	4	0	7	7	7
7	7	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	2	0	7	7
7	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	2	2	0	7
7	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	2	0	7
0	4	4	4	4	0	0	0	0	4	4	4	4	4	4	4	0	0	0	0	4	4	4	4	0	7
0	4	4	4	0	7	1	1	7	0	4	4	4	4	4	0	7	1	1	7	0	4	4	4	4	0
0	4	4	4	0	7	1	0	1	7	0	4	4	4	0	7	1	0	1	7	0	4	4	4	4	0
0	4	4	4	4	0	7	1	1	7	0	4	4	4	0	7	1	1	7	0	4	4	4	4	4	0
0	4	4	4	4	4	0	0	0	0	4	4	4	4	4	0	0	0	0	4	4	4	4	4	4	0
0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	0
0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	7
7	0	4	4	4	4	4	4	4	4	1	1	1	1	1	4	4	4	4	4	4	4	4	4	0	7
7	0	4	4	4	4	4	4	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	0	7	7
7	7	0	4	4	4	4	4	4	1	1	7	7	7	1	1	4	4	4	4	4	4	4	0	7	7
7	7	0	4	4	4	4	4	4	4	1	1	1	1	1	4	4	4	4	4	4	4	0	7	7	7
7	7	7	0	4	4	4	4	4	4	4	1	1	1	4	4	4	4	4	4	4	0	7	7	7	7
7	7	7	7	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	0	7	7	7	7	7
7	7	7	7	7	0	4	4	4	4	4	4	4	4	4	4	4	4	4	0	7	7	7	7	7	7
7	7	7	7	7	7	0	4	4	4	4	0	0	0	4	4	4	4	0	7	7	7	7	7	7	7
7	7	7	7	7	7	7	0	0	0	0	7	7	7	0	0	0	0	7	7	7	7	7	7	7	7

Question n° 1 : Combien faut-il de nombres pour coder l'image ci-dessus ?

Question n° 2 : Décodez l'image de la page suivante. Que représente-t-elle ?
Vous pouvez aussi poster une photo de votre coloriage en utilisant le formulaire en ligne sur le site.

7	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	0	0	0	7
0	0	5	0	0	0	7	7	7	7	0	0	7	7	0	0	7	7	7	7	0	0	0	5	0	0
0	0	5	5	5	0	0	7	7	7	7	0	7	7	0	7	7	7	7	0	0	5	5	5	0	0
7	0	5	5	5	5	0	0	7	7	7	0	7	7	0	7	7	7	0	0	5	5	5	5	0	7
7	0	0	0	0	0	0	0	0	7	7	0	0	0	0	7	7	0	0	0	0	0	0	0	0	7
7	7	0	1	1	0	3	3	0	0	7	7	0	0	7	7	0	0	3	3	0	1	1	0	7	7
7	7	0	0	0	0	0	3	3	0	0	7	0	0	7	0	0	3	3	0	0	0	0	0	7	7
7	7	0	4	4	4	0	0	3	3	0	0	6	6	0	0	3	3	0	0	4	4	4	0	7	7
7	7	0	4	4	4	0	0	0	3	3	0	0	0	0	3	3	0	0	0	4	4	4	0	7	7
7	7	0	4	4	4	0	4	0	0	3	3	0	0	3	3	0	0	4	0	4	4	4	0	7	7
7	7	0	4	4	4	4	4	4	4	0	0	0	0	0	0	4	4	4	4	4	4	4	0	7	7
7	7	0	0	4	4	0	0	0	0	4	0	0	0	0	4	0	0	0	0	4	4	0	0	7	7
7	7	7	0	0	0	2	2	2	2	0	0	0	0	0	0	2	2	2	2	0	0	0	7	7	7
7	7	7	0	2	2	2	2	2	2	2	0	0	0	0	2	2	2	2	2	2	2	0	7	7	7
7	7	7	0	2	2	2	2	2	2	2	0	0	0	0	2	2	2	2	2	2	2	0	7	7	7
7	7	0	2	2	2	2	2	2	2	2	2	0	0	2	2	2	2	2	2	2	2	2	0	7	7
7	7	0	2	2	2	2	2	2	2	2	2	0	0	2	2	2	2	2	2	2	2	2	0	7	7
7	7	0	2	2	2	2	2	2	2	2	2	0	0	2	2	2	2	2	2	2	2	2	0	7	7
7	7	0	2	2	2	2	2	2	2	2	0	0	0	0	2	2	2	2	2	2	2	2	0	7	7
7	7	0	2	2	2	2	2	2	2	2	0	6	6	0	2	2	2	2	2	2	2	2	0	7	7
7	7	7	0	2	2	2	2	2	2	2	0	6	6	0	2	2	2	2	2	2	2	0	7	7	7
7	7	7	0	2	2	2	2	2	2	0	7	0	0	7	0	2	2	2	2	2	2	0	7	7	7
7	7	7	0	2	2	2	2	2	2	0	7	0	0	7	0	2	2	2	2	2	2	0	7	7	7
7	7	7	7	0	2	2	2	2	2	0	7	0	0	7	0	2	2	2	2	2	0	7	7	7	7
7	7	7	7	7	0	2	2	2	0	7	7	7	7	7	7	0	2	2	2	0	7	7	7	7	7
7	7	7	7	7	7	0	0	0	7	7	7	7	7	7	7	7	0	0	0	7	7	7	7	7	7

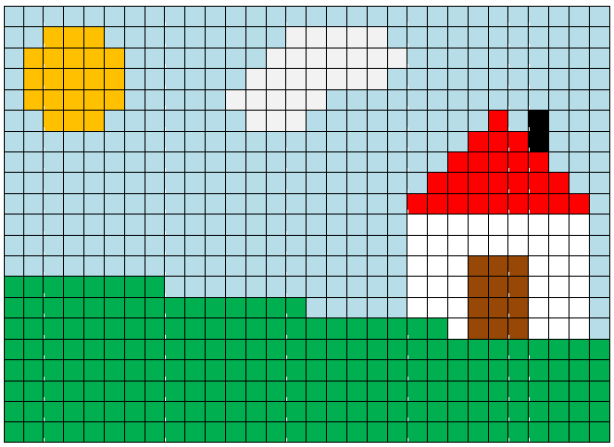
On manque de place !

Rappelez-vous : dans le défi précédent, nous avions besoin de 26 x 26 = 676 nombres pour enregistrer la petite image de la pomme. Mais il s'agissait d'une toute petite image, avec seulement 8 couleurs différentes.
Avec une plus grande image et plus de couleurs, nous aurions besoin de beaucoup plus d'espace de stockage.

L'image de la Joconde (ci-contre) utilise beaucoup plus de couleurs.
Prise avec un appareil photo qui produit des images de 4000 x 3000 pixels avec 16 millions de couleurs différentes, il faudrait stocker 36 000 000 (36 millions !) de nombres pour l'enregistrer.

Les données numériques stockées sur Internet, dans nos ordinateurs et nos téléphones prennent de plus en plus de place. Visionnez le petit film qui accompagne ce défi. Il s'agit d'un extrait d'un journal télévisé qui alerte sur les problèmes causés par ce stockage de données en forte augmentation.

Observez maintenant l'image ci-dessous.
- Combien de nombres sont nécessaires pour la stocker dans un ordinateur ?
- Pouvez-vous imaginer un moyen de réduire ce nombre ?



Observez la première ligne : elle est constituée de 30 pixels bleus. Le bleu est codé par le nombre 1.
Au lieu d'enregistrer 30 fois le nombre 1, nous pouvons inventer une méthode plus économique.
1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
deviendra 1-30.
Le premier nombre (1) indique la couleur,
Le deuxième (30) le nombre de cases successives de cette couleur.
Au lieu de 30 nombres, nous n'en utilisons plus que 2.
Nous avons économisé 30 - 2 = 28 nombres.

La deuxième ligne serait codée ainsi : 1-2-2-3-1-9-3-5-1-11
Que l'on peut traduire par : bleu x 2, jaune x 3, bleu x 9, gris x 5, bleu x 11.

Question n° 1 : Codez la troisième ligne.

Question n° 2 : Décodez l'image page suivante, puis prenez-la en photo et joignez cette photo à votre réponse.

Question n° 3 : Combien de nombres avons-nous économisés en codant cette image ?

La méthode que nous venons d'inventer pour diminuer la quantité de données nécessaire à l'enregistrement de l'image s'appelle un algorithme. Un algorithme, c'est une suite d'instructions permettant de résoudre un problème.
Comme il s'agissait de réduire (comprimer) la taille des données, nous l'appelons un algorithme de compression.

											0 4 7 3 0 4
											0 3 7 5 0 3
											0 2 7 3 5 1 7 3 0 2
											0 2 7 2 5 1 1 1 5 1 7 2 0 2
											0 2 7 3 5 1 7 3 0 2
											0 3 7 5 0 3
											0 4 7 3 0 4
											0 5 4 1 0 5
											0 5 4 1 0 5
											0 4 4 1 0 2 4 3 0 1
											0 4 4 5 0 2
											0 3 4 1 0 7
											0 2 4 1 0 8
											0 1 4 1 0 9
0=blanc 1=noir 2=bleu 3=rouge 4=vert 5=jaune 6=orange 7=violet 8=gris											

Rétrécir un texte

Lors du précédent défi, nous avons découvert une méthode permettant de diminuer la quantité de nombres à stocker pour représenter une image. Nous l'avons déduite en observant attentivement la composition de cette image. Nous allons maintenant essayer d'appliquer la même technique pour « comprimer » un texte.

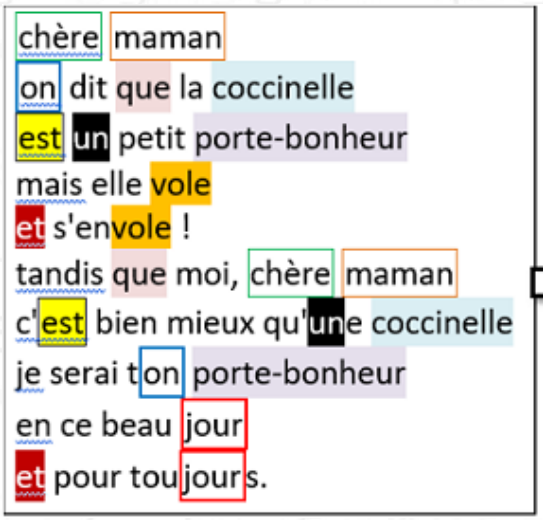
chère maman
on dit que la coccinelle
est un petit porte-bonheur
mais elle vole
et s'envole !
tandis que moi, chère maman
c'est bien mieux qu'une coccinelle
je serai ton porte-bonheur
en ce beau jour
et pour toujours.

Observez le texte ci-contre. Il contient 207 caractères et 38 mots. Comment réduire le nombre de caractères à enregistrer dans l'ordinateur, tout en conservant le texte identique ?

Certains mots se répètent.
Certaines suites de lettres se retrouvent dans plusieurs mots.
Essayez de les repérer en les surlignant ou en les entourant de différentes couleurs.

Comment utiliser ces répétitions pour réduire la longueur du texte ? En s'inspirant du codage des couleurs vu précédemment, on va attribuer un nombre à chaque séquence qui se répète et constituer une légende comme celle utilisée pour les couleurs. Comme il s'agit de mots et de lettres, nous l'appellerons un dictionnaire.

Voici un exemple de codage du texte, avec le dictionnaire associé. Chaque mot du texte présent dans le dictionnaire est remplacé par son entrée, précédée d'un signe qui permet de différencier une lettre du texte d'une entrée du dictionnaire.


chère maman
on dit que la coccinelle
est un petit porte-bonheur
mais elle vole
et s'envole !
tandis que moi, chère maman
c'est bien mieux qu'une coccinelle
je serai ton porte-bonheur
en ce beau jour
et pour toujours.

Dictionnaire	
A	chère
B	maman
C	on
D	que
E	coccinelle
F	est
G	un
H	porte-bonheur
I	vole
J	et
K	jour

@A @B
@C dit @D la @E
@F @G petit @H
mais elle @I
@J s'en@I !
tandis @D moi, @A @B
c'@F bien mieux qu'@Ge @E
je serai t@C @H
en ce beau @K
@J pour tou@Ks.

A vous maintenant de décoder le texte de la page suivante.

Remarquez que pour pouvoir décoder un texte, il faut connaître le dictionnaire utilisé. Celui-ci devra donc être enregistré avec le texte codé. Pour la poésie "Chère maman", nous n'avons pas gagné beaucoup de place. Et si nous ajoutons le dictionnaire, nous en avons même perdu ! C'est parce que notre texte est court. Dans un texte plus long, les mots se répètent plus souvent, ce qui permet de gagner plus de place. Et puis nous avons fait une erreur en réalisant notre dictionnaire... Certains codages sont inutiles et ne font pas gagner de place...

Pouvez-vous trouver des codages inutiles dans le dictionnaire de la poésie "Chère maman" ?

Décode le texte suivant.
Le dictionnaire utilisé est en bas de page

¤A ¤E

J'ai rencontré ¤A ¤E
¤B s'en all¤F ¤Ltable au dos
¤H ¤K le pré trois lima¤Cs
¤B dis¤F par co¤Hr leur le¤C.

¤H ¤K un champ, qua¤I lézards
¤B écriv¤F un long devoir.

Où p¤Ht se trouver l¤Hr école ?
¤M mili¤H des avoines fol¤G ?

Et leur maî¤I est-il ce corb¤D
Que je vois dessiner là-h¤Mt
De bel¤G let¤Is au tabl¤D ?

M¤Mrice ¤Lême

Dictionnaire	
A	trois
B	qui
C	çon
D	eau
E	Es¤Lgots
F	aient
G	les
H	eu
I	puis
J	tre
K	dans
L	car
M	au
N	et

Recopie ici le texte en clair

Attention aux erreurs !

Les ordinateurs traitent beaucoup de données. Ces données circulent d'un endroit à un autre, en passant par des câbles, par Internet, par radio, en WiFi, ou sur le réseau des téléphones portables, etc. Il arrive fréquemment que le transfert soit perturbé. Des erreurs se glissent alors dans les données. Il faut donc détecter ces erreurs.
Si une erreur est détectée, alors il faut recommencer le transfert.

Supposons que l'ordinateur A envoie une série de nombres à l'ordinateur B. Comment ce dernier peut-il savoir qu'il y a une erreur dans les données qu'il a reçues ? C'est impossible !

Pour résoudre ce problème, l'ordinateur A va envoyer, pour chaque nombre transmis, une information supplémentaire qui permettra à B de vérifier qu'il s'agit bien du bon nombre. Cette information s'appelle un code de contrôle.

Ainsi, chaque fois qu'une information est transmise, elle comporte un code supplémentaire qui va permettre au récepteur de vérifier que l'information reçue est bien exacte.

Lorsque la caissière balaye un code-à-barres avec son lecteur, celui-ci est systématiquement contrôlé. Si aucune erreur n'est trouvée, un bip avertit la caissière qu'elle peut passer à l'article suivant. Pour vérifier qu'il n'y a pas eu d'erreur de lecture, l'ordinateur de la caisse utilise le dernier chiffre qui est en fait un code de contrôle.

Un code à barres de 13 chiffres comporte donc 12 chiffres pour identifier l'article, plus un chiffre de contrôle placé à la fin. Ce chiffre a été calculé en utilisant les 12 premiers. L'ordinateur refait le même calcul et vérifie que son résultat correspond au chiffre de contrôle.

Vous pouvez facilement vérifier vous-mêmes un code à barres car l'algorithme utilisé n'est pas compliqué.

- Munissez-vous de deux livres et ouvrez dans votre traitement de texte le document "vérification d'un code à barres" joint à ce défi.
- Suivez les indications et travaillez directement à l'ordinateur, sans imprimer le document. C'est un bon entraînement à l'usage du traitement de texte.
- Quand vous avez terminé, enregistrez le document dans votre ordinateur, sur le bureau par exemple.
- Joignez ce document à votre réponse et demandez la correction.

Algorithme de vérification d'un code à barres

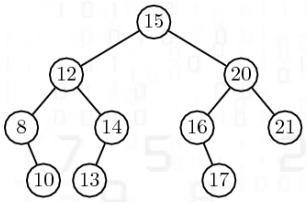
	X	1	=	
	X	3	=	
	X	1	=	
	X	3	=	
	X	1	=	
	X	3	=	
	X	1	=	
	X	3	=	
	X	1	=	
	X	3	=	
	X	1	=	
	X	3	=	
10	-		=	

1. Inscris verticalement, dans la première colonne, le code à barres du livre que tu as choisi.
2. Effectue les calculs :
   - D'abord les multiplications,
   - Puis l'addition de la 5ème colonne,
   - Reporte le chiffre des unités dans l'autre case verte,
   - Effectue la soustraction.
3. Le chiffre de contrôle (1ère case rouge) et le résultat de l'algorithme (2ème case rouge) doivent être égaux.

Algorithme de vérification d'un code à barres

	X	1	=	
	X	3	=	
	X	1	=	
	X	3	=	
	X	1	=	
	X	3	=	
	X	1	=	
	X	3	=	
	X	1	=	
	X	3	=	
	X	1	=	
	X	3	=	
10	-		=	

1. Inscris verticalement, dans la première colonne, le code à barres du livre que tu as choisi
2. Effectue les calculs :
   - D'abord les multiplications,
   - Puis l'addition de la 5ème colonne,
   - Reporte le chiffre des unités dans l'autre case verte
   - Effectue la soustraction
3. Le chiffre de contrôle (1ère case rouge) et le résultat de l'algorithme (2ème case rouge) doivent être égaux



Recherche rapide

Les ordinateurs sont souvent utilisés pour rechercher des informations. Reprenons l'exemple de la caissière qui scanne un code à barres à la caisse. Pour identifier l'article et connaître son prix, l'ordinateur de la caisse va devoir rechercher ce code parmi ceux de tous les articles du supermarché ! Supposons que le magasin ait 20 000 articles en rayon, et que l'ordinateur puisse vérifier 100 codes par seconde.

Question n° 1 : *Combien de temps va-t-il mettre pour examiner tous les codes du magasin ?*

Question n° 2 : *Combien te temps faudra-t-il à la caissière pour scanner tout un chariot de 50 articles ?*

Les réponses à ces deux questions montrent clairement qu'il faut trouver un moyen d'accélérer la vérification des codes, ou bien le supermarché ne tardera pas à faire faillite !

Heureusement, il existe des algorithmes qui permettent d'accélérer la recherche. Dans ce défi, nous allons découvrir l'*algorithme de recherche binaire*.

Observez la suite d'articles ci-dessous étiquetés de A à M. Pour retrouver l'un de ces codes et identifier l'article correspondant, l'ordinateur va devoir les examiner un par un, méthodiquement, depuis le début. S'il a de la chance, il tombera rapidement sur le bon. Mais il peut aussi perdre du temps si le code recherché est à la fin.

A	B	C	D	E	F	G	H	I	J	K	L	M
25	634	85221	512	5300	125	48	256	800	7	12640	26	874

Question n° 3 : *Quel est le nombre de tentatives maximum pour trouver le bon code ?*

Pour accélérer la recherche, le logiciel de gestion de l'ordinateur range les codes par ordre croissant. Cela demande plus de calculs lorsque l'on ajoute un article, car il faudra replacer son code au bon endroit dans la liste. Mais cela va aussi permettre d'accélérer considérablement la recherche :

A	B	C	D	E	F	G	H	I	J	K	L
7	25	26	48	125	256	512	634	800	874	5300	12640

Supposons que la caissière ait scanné le code 5300.

L'ordinateur commence par le comparer à celui de l'article G *situé au milieu* : 512

Comme 5300 est plus grand que 512 et que les codes sont rangés par ordre croissant, l'ordinateur peut éliminer tous les articles situés avant G :

~~A~~	~~B~~	~~C~~	~~D~~	~~E~~	~~F~~	~~G~~	H	I	J	K	L
~~7~~	~~25~~	~~26~~	~~48~~	~~125~~	~~256~~	~~512~~	634	800	874	5300	12640

Divisons à nouveau la partie restante en 2 et comparons le code recherché avec celui de l'article J situé au milieu : 874. Comme 5300 est plus grand que 874, on peut maintenant éliminer 3 articles supplémentaires :

~~A~~	~~B~~	~~C~~	~~D~~	~~E~~	~~F~~	~~G~~	~~H~~	~~I~~	~~J~~	K	L
~~7~~	~~25~~	~~26~~	~~48~~	~~125~~	~~256~~	~~512~~	~~634~~	~~800~~	~~874~~	5300	12640

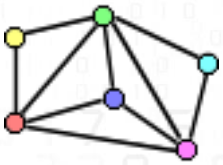
Il ne reste plus que 2 codes à tester. L'ordinateur aura donc identifié le bon article en *4 essais au maximum*, au lieu de 11 s'il avait dû examiner tous les codes l'un après l'autre.

Cette méthode de recherche qui consiste à diviser en 2 à chaque étape s'appelle *recherche binaire*.

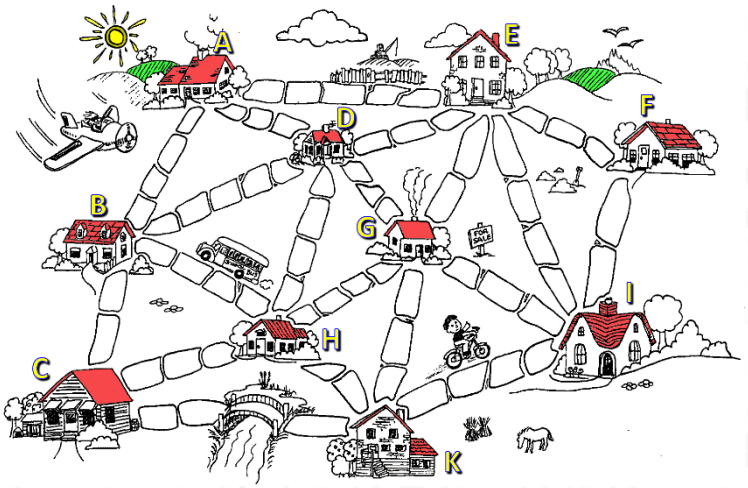
A vous de vous entraîner à la recherche binaire en jouant sur cette page. Vous devriez pouvoir trouver n'importe quel nombre en 5 coups maximum.

http://cerp-lechapus.net/defis_programmation/recherche_binaire/

La ville embourbée



Dans cette petite ville, il était très difficile de circuler car il n'y avait pas de rues. Dès qu'il pleuvait, le sol devenait boueux, les voitures s'embourbaient et ne pouvaient plus avancer. Pour résoudre ce problème, le maire décida de paver certaines rues. Mais la ville n'ayant pas beaucoup d'argent à dépenser, il choisit de paver le moins de rues possible. De plus, certaines rues étaient plus longues que d'autres !



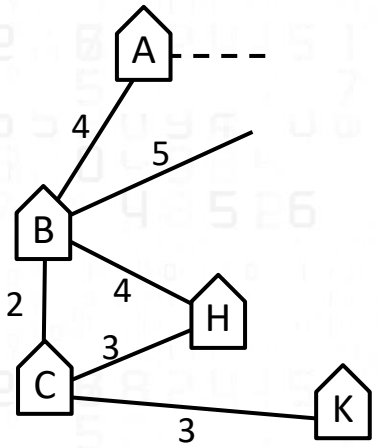
Il demanda donc à son directeur des travaux de respecter les 2 conditions suivantes :

1. *Paver le moins de rues possible*, mais suffisamment pour que chaque habitant puisse se rendre n'importe où depuis chez lui.
2. *Dépenser le moins d'argent possible*. Sur le plan de la ville, le nombre de « sections à paver » entre chaque maison représente la dépense à engager. Plus il y a de sections, plus la rue coûtera cher à aménager. Il faut 850 € pour paver une section.

Question n° 1 : Complétez votre plan d'aménagement ci-dessous, puis scannez-le (ou photographiez-le) et joignez-le à votre réponse.

Plan d'aménagement de la ville

1. En observant le plan du village ci-dessus, termine le schéma :

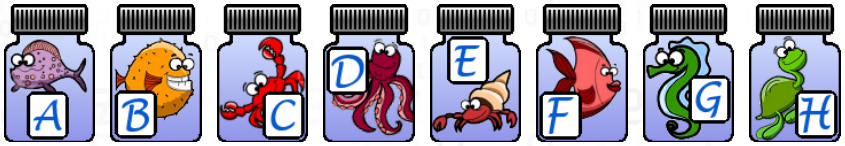


2. Puis recherche au crayon les rues à paver en respectant les 2 conditions fixées par le maire de la ville.
3. Quand ton plan est terminé, repasse les rues sélectionnées au feutre ou au surligneur.
4. Calcule le prix de revient de ton aménagement.
5. Compare ton plan avec ceux réalisés par les autres élèves et sélectionnez le meilleur.
6. Scannez ou photographiez ce plan et joignez-le à votre réponse sur le site des défis-programmation, en n'oubliant pas d'indiquer son coût.

Range-moi tout ça !

Nous avons vu dans le défi n°8 qu'il était beaucoup plus facile de retrouver un nombre dans une liste si cette liste avait été au préalable triée par ordre croissant. Mais trier des nombres, ça peut prendre du temps, surtout s'il y en a beaucoup. Si vous avez déjà essayé de remettre dans l'ordre un jeu de 52 cartes à jouer, vous avez pu vous rendre compte de la difficulté.

Damien est océanographe. De son dernier voyage, il a ramené les 8 spécimens ci-dessous, étiquetés de A à H. Pour mener à bien ses expériences, il doit d'abord les ranger du plus léger au plus lourd :



Pour faire ce tri, Damien dispose d'une balance à deux plateaux qui va lui permettre de comparer les pots deux à deux.

"*Hou là là !... Mais je ne sais pas par où commencer !*" s'exclame-t-il.
- Facile, le rassure Caroline, son assistante. *Il existe une méthode pour ne pas faire d'erreur. C'est l'algorithme du tri à bulle.*
- Le tri à bulle ??? Mais comment cela fonctionne-t-il ?
- C'est simple, explique Caroline. *On va commencer par trouver le plus léger. On prend un pot, par exemple le premier, et on le compare sur la balance avec tous les autres, un par un. A chaque pesée, on ne garde que le plus léger et on repose l'autre. A la fin, on aura trouvé le plus léger et on pourra le ranger en premier.*
- Ah oui. Ce n'est pas mal. J'aurai trouvé le plus léger avec 7 pesées seulement. Mais après ?
- Et bien on recommence avec les 7 pots restants. On trouve le plus léger en 6 pesées et on le range en deuxième position.

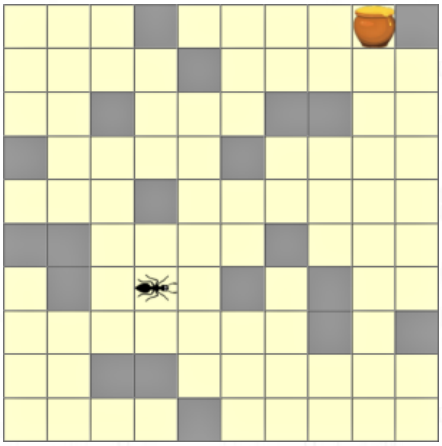
Question n°1 : *Combien de pesées seront nécessaires pour trier les 8 pots ?*

Question n°2 : *Sur cette page, vous allez vous-mêmes trier les 8 pots. Quand vous aurez réussi, une phrase-clef s'affichera à l'écran : recopiez-la dans votre réponse.*

http://cerp-lechapus.net/defis_programmation/tri/

- Mais... euh... Caroline, pourquoi appelle-t-on cette méthode le tri à bulle ?
- Et bien c'est parce qu'à chaque série de pesées pour trouver le plus léger, on teste tous les pots en remontant, un peu comme une bulle d'air qui remonte à la surface de l'eau.

La fourmi et le pot de miel



Cette petite fourmi a bien senti une odeur appétissante...
Mais comment arriver jusqu'au pot de miel ?

Vous allez devoir l'aider en *programmant son trajet* à l'aide des commandes suivantes :

Attention, lorsque la fourmi tourne à gauche, il s'agit de **sa** gauche et non de la vôtre ! Et quand elle avance, c'est devant elle et pas forcément vers le haut de l'écran.
Pour programmer le trajet vous devrez vous mettre à la place de la fourmi.

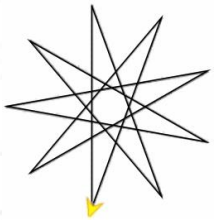
Quand vous penserez que votre programme et terminé, cliquez sur **GO** pour le lancer.
S'il y a une erreur, il faudra le modifier : cliquez sur une commande à effacer puis sur la gomme.

Lorsque la fourmi aura atteint son but, l'écran affichera une *phrase-mystère que vous devrez recopier* dans votre réponse comme preuve de votre réussite, puis vous pourrez demander la correction du défi.

Vous êtes prêts ? Cliquez sur ce lien :
http://cerp-lechapus.net/defis_programmation/lafourmi/index.htm?j=5&g=48361.96590566635&d=4&m=1

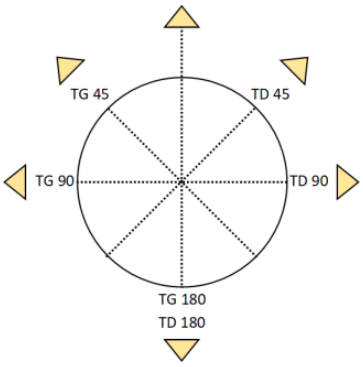
Lorsque vous aurez terminé ce défi, n'hésitez pas à tester les autres jeux proposés par la fourmi !

La tortue qui sait dessiner



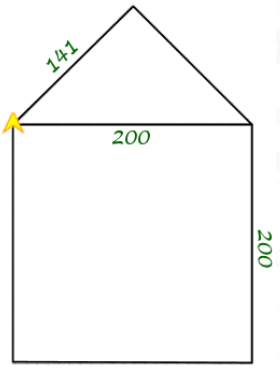
Vous connaissez peut-être déjà la célèbre "*tortue LOGO*". LOGO est un langage informatique conçu pour apprendre à programmer facilement, notamment avec la tortue graphique qui permet de dessiner sur l'écran de belles figures géométriques. Ci-contre, vous pouvez voir à l'œuvre un programme dessinant une étoile à 9 branches. Pour commander la tortue, LOGO propose des commandes simples dont voici les principales. Vous allez pouvoir les tester en ligne sur la page LOGO des défis-programmation : http://cerp-lechapus.net/defis_programmation/logo/
Prenez le temps de lire attentivement les explications qui suivent, puis connectez-vous sur la page LOGO et testez-les. Attention, les instructions et les nombres doivent toujours être séparés par un espace !

Commande	Traduction	Exemples
AV	AVance	AV 100 : le nombre qui suit AV indique de combien de pixels la tortue va avancer
RE	REcule	RE 70 : la tortue va reculer de 70 pixels
TD	Tourne à Droite	TD 90 : La tortue tourne à droite à angle droit (90°). Le nombre correspond à l'angle mesuré en degrés (lisez les explications sur les angles)
TG	Tourne à Gauche	TG 90 : La tortue tourne à gauche à angle droit (90°).
LC	Lève Crayon	LC : Après cette instruction, la tortue se déplacera sans tracer de trait.
BC	Baisse Crayon	BC : Après cette instruction, la tortue va de nouveau tracer des traits.
VE	Vide Ecran	VE : Tous les tracés sont effacés. Pratique pour repartir de zéro avec un écran vierge et composer un nouveau programme.
REPETE	Répète	REPETE 4 [AV 100 TD 90] : Cette instruction est très puissante. Dans cette exemple, la tortue va répéter 4 fois toutes les instructions entre les crochets. On obtiendra un carré. Vous pouvez le vérifier en copiant le texte de ce petit programme puis en le collant sur la page LOGO, dans la zone de saisie



Les angles: Vous connaissez déjà l'*angle droit* que l'on retrouve dans beaucoup de figures géométriques que vous avez étudiées, comme le rectangle ou le carré. Mais une route ou un chemin ne tourne pas toujours à angle droit, ce serait bien monotone. Alors on a inventé une unité pour mesurer les angles : le *degré*. Et il se trouve que *l'angle droit mesure exactement 90 degrés*. Voilà pourquoi TD 90 fait tourner la tortue à droite d'un angle droit.
Pour réaliser ce défi, vous aurez besoin d'utiliser des angles à *45 degrés* (la moitié d'un angle droit) et peut-être aussi à *180 degrés* (angle plat ou demi-tour). Remarquez que si vous tournez de 360 degrés, vous n'aurez pas changé de direction, juste fait un tour sur vous-même. Ci-contre pour vous aider, voici le rapporteur de Michel, gradué avec ces angles. **Attention**, comme pour la fourmi, il faut vous mettre à la place de la tortue pour savoir s'il faut tourner à gauche ou bien à droite.

Quand vous vous serez bien entrainés, vous pourrez vous attaquer au défi suivant : Il s'agit de construire un programme LOGO qui affichera la figure ci-contre. Pour vous aider, j'ai indiqué quelques dimensions en pixels et signalé la position de départ de la tortue. Voici comment procéder :



1. Connectez-vous à la page LOGO et écrivez votre programme, en mettant l'instruction VE sur la première ligne. Ainsi, chaque fois que vous lancerez votre programme pour le tester, l'écran sera effacé.
2. Écrivez, modifiez, corrigez votre programme, en laissant toujours la commande VE en premier. Chaque fois que vous cliquez sur "LANCER", le programme s'exécute : vous pouvez voir le résultat et faire les modifications nécessaires.
3. Quand votre programme est au point, sélectionnez tout votre texte et copiez-le (Clic droit => copier) puis ouvrez le défi, collez le texte dans votre réponse et demandez la correction.

Un café s'il vous plait !

Les ordinateurs ne sont pas intelligents. Ils se contentent de suivre les instructions que l'on a programmées pour eux. Et comme ils ne sont pas très futés, il faut tout prévoir à leur place. Exemple : si je vous dis de passer par une porte, je n'ai pas besoin de préciser qu'il faut l'ouvrir avant. Mais si je programme un robot et que j'oublie de lui dire d'ouvrir la porte avant d'avancer, il risque de la défoncer !

Les ordinateurs sont utilisés pour faire fonctionner la plupart des machines. A chaque fois, il faut programmer les bonnes instructions et vérifier que l'appareil réagit bien comme prévu. Prenons l'exemple d'un distributeur de boissons :

- Le programme doit être capable de déclencher des actions : *Faire tomber un gobelet, Le remplir de café ou de chocolat.*
- Mais il doit aussi tester les choix de l'utilisateur : *Quelle boisson a-t-il choisie ? Veut-il du sucre ? A-t-il mis assez d'argent dans le monnayeur ?*
- Il devra aussi afficher des messages pour guider l'utilisateur.

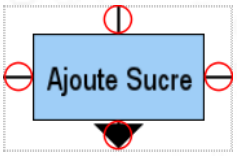
Votre mission pour ce dernier défi consiste à programmer correctement un distributeur de café/chocolat.
Vous le trouverez sur cette page : http://cerp-lechapus.net/defis_programmation/automate/

- L'utilisateur insère des pièces en cliquant sur le monnayeur, puis il fait ses choix.
- Le bouton vert permet ensuite de lancer le programme et d'obtenir la boisson demandée.
- On retire la boisson en cliquant sur le gobelet.

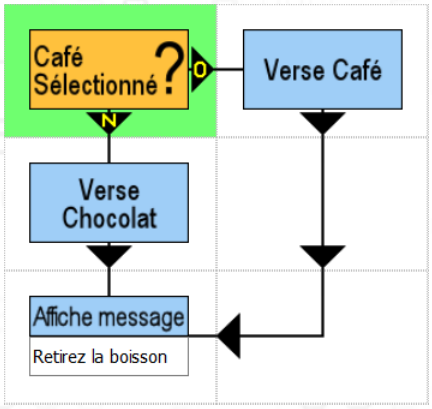
Voici quelques précisions à lire attentivement

- Commencez par ouvrir les portes du distributeur pour accéder au programme : il suffit de cliquer sur les poignées et les panneaux coulissent.
- A droite se trouvent les instructions à votre disposition : les actions en bleu, les tests en orange, les flèches en noir.
- Pour réaliser votre programme, vous faites glisser à la souris les instructions sur la partie gauche.
- Votre programme devra toujours commencer à la case Départ.
- En cas d'erreur, vous pouvez supprimer une instruction en la faisant glisser sur la poubelle.

Vous avez remarqué que les instructions doivent s'enchaîner :



- Chaque action (en bleu) possède des entrées et une sortie.
- Chaque test (en orange) possède des entrées, une sortie OUI (si la réponse est oui) et une sortie NON (si la réponse est non).
- Si les entrées ou les sorties ne correspondent pas à ce que vous voulez faire, vous pouvez les changer facilement : cliquez sur une entrée ou une sortie (voir ci-contre) et choisissez dans la liste.
- Faites de même avec les flèches pour les adapter à vos besoins.



Essayez de reproduire le petit programme ci-contre qui préparera une boisson en fonction du choix de l'utilisateur. Remarquez que les entrées et sorties des instructions ont été modifiées pour qu'une sortie corresponde toujours à une entrée. La sortie de la dernière instruction, « Affiche message », a été supprimée. En effet, une sortie qui ne déboucherait sur rien provoquerait une erreur !

Quand vous aurez fait plusieurs essais et compris comment programmer le distributeur, vous pourrez réaliser un programme plus complet. Le petit programme précédent distribue des boissons gratuitement, ne propose pas de sucre, etc. Vous allez devoir penser à toutes les situations et les traiter dans le bon ordre, en commençant par le prix bien sûr. Voici un petit plan de travail pour vous aider à démarrer votre programme :

1. Le client a-t-il mis suffisamment d'argent dans le monnayeur ?
   - Si NON, alors il faut tout arrêter et afficher un message le signalant
   - Si OUI, alors on peut continuer :
2. Il faut maintenant retirer le prix de la boisson (0,50€) de l'argent du monnayeur.
3. Ensuite, il faut tester un par un les choix de l'utilisateur et programmer les actions correspondantes (ou ne pas les programmer si la réponse est NON).
   - A-t-il choisi café ou chocolat ?
   - Veut-il du sucre ?
   - A-t-il besoin d'une touillette ?
4. ...
5. Lorsque tout a été examiné, il faut afficher un message demandant à l'utilisateur de retirer sa boisson, et penser à supprimer la dernière sortie.

Testez votre programme en cliquant sur le bouton vert. S'il ne convient pas ou ne fonctionne pas bien, corrigez-le et testez-le à nouveau.

Lorsque vous aurez écrit un programme intéressant, vous devrez me l'envoyer.

1. Sauvegardez-le sur votre ordinateur en cliquant sur le bouton "ENREGISTRER".
2. Enregistrez le fichier *sauvegarde.txt* sur le bureau de l'ordinateur (vous pouvez lui donner un autre nom).
3. Ouvrez à nouveau le défi, joignez ce fichier à votre réponse et demandez la correction.

Les boutons "ENREGISTRER" et "CHARGER" vous permettent de créer autant de programmes que vous le désirez, et de les reprendre pour les modifier. Si vous travaillez en ateliers, chaque élève pourra ainsi créer son propre programme.

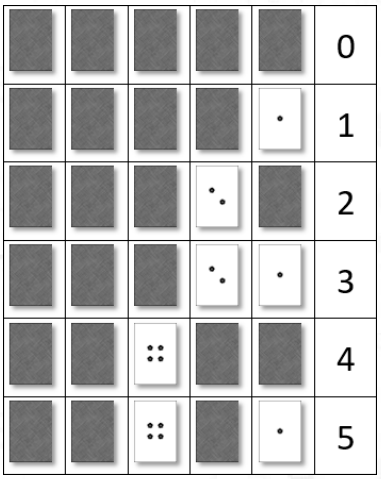
Il s'agit du dernier défi-programmation de l'année. C'est aussi le plus difficile ! Mais vous avez du temps pour réfléchir et m'envoyer de super-programmes. Vous obtiendrez peut-être des résultats surprenants alors n'oubliez pas : Un ordinateur n'est qu'une machine, c'est à vous de penser à tout !

Voici quelques compléments d'informations et des propositions d'activités pour chacun des défis

Des zéros et des uns

Les nombres proposés ont été volontairement limités à 5 bits, ce qui permet de coder des nombres du système décimal de 0 à 31. Vous pouvez bien sûr aller plus loin avec des élèves de cycle 3. Les systèmes informatiques utilisent des nombres à 8 bits, appelés octets, qui permettent de coder des nombres du système décimal de 0 à 255. Il peut être intéressant d'aborder la numération binaire sous la forme d'un jeu de cartes comme celles présentées sur l'image ci-dessous. Voici quelques idées d'activités mathématiques sur le système binaire :

- Une recherche sur les puissances de 2 pour continuer la suite géométrique :
  1 => 2 => 4 => 8 => 16 => 32 => 64 => 128 => …
- Compter en binaire avec un jeu de cartes :
  L'observation de ce comptage doit permettre de dégager un algorithme « *Ajouter 1 à un nombre binaire* » à partir des remarques suivantes :
  - Les cartes de droite (petits nombres) sont plus souvent retournées que celles de gauche.
  - La carte [1] est systématiquement retournée.
  - On arrête les retournements dès qu'une carte cachée est retournée. Cette dernière remarque permet de formuler l'algorithme :
    « *Pour ajouter 1, retourner les cartes en partant de la droite, jusqu'à ce que l'on retourne une carte qui était cachée*. »



- De la même façon, on peut aussi rechercher d'autres algorithmes :
  - « Pour retrancher 1, retourner les cartes en partant de la droite, jusqu'à ce que l'on retourne une carte visible »
  - « Pour multiplier par deux, décaler toutes les cartes d'une place vers la gauche »

Ces différents algorithmes correspondent exactement aux calculs effectués par le processeur de l'ordinateur qui ne traite que des 1 et des 0. Par exemple « *Pour ajouter 1, retourner les cartes en partant de la droite, jusqu'à ce que l'on retourne une carte qui était cachée*. » traduit en langage informatique donne « *Pour ajouter 1, inverser les bits en partant de la droite, jusqu'à ce que l'on inverse un 0 en 1*. »

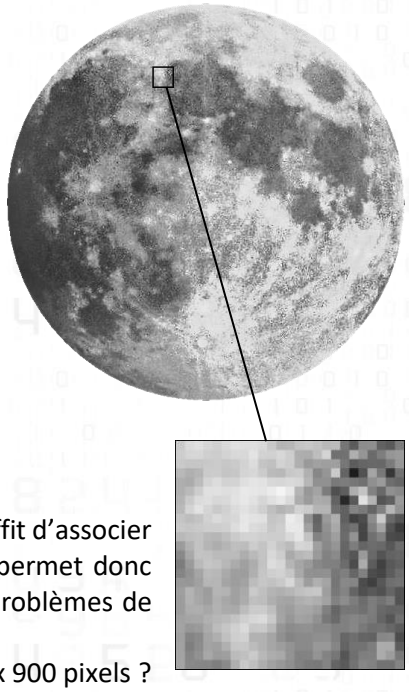
Communication lumineuse

Cette activité introduit la notion de codage des lettres. L'ordinateur ne traite que des nombres et chaque lettre est donc représentée par un nombre particulier. Un code international a été créé pour cela : le code ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange ou Code américain normalisé pour l'échange d'information). Dans ce défi, chaque lettre est associée à son rang dans l'alphabet, ce qui n'est pas exactement le cas dans le code ASCII.
Comme vous pouvez le voir sur le tableau ci-contre, le bit 7 de tous les codes est positionné à 1, ajoutant donc 64 au rang de la lettre dans l'alphabet.
A=65, B=66, C=67, ... … Z=90.

ASCII Alphabet			
A	1000001	N	1001110
B	1000010	O	1001111
C	1000011	P	1010000
D	1000100	Q	1010001
E	1000101	R	1010010
F	1000110	S	1010011
G	1000111	T	1010100
H	1001000	U	1010101
I	1001001	V	1010110
J	1001010	W	1010111
K	1001011	X	1011000
L	1001100	Y	1011001
M	1001101	Z	1011010

Le codage des lettres est l'occasion d'aborder la cryptographie sous la forme d'une recherche mathématique, comme dans « Le chiffre de César » proposé en annexe.

Du noir et du vert



Pour expliquer aux élèves le principe d'affichage des images numériques, il suffit de pointer une loupe sur un moniteur, ou bien sur l'écran d'un vidéoprojecteur. On peut alors apercevoir les fameux « pixels » qui sont les points composant l'image. Chacun de ces pixels peut prendre une couleur différente, représentée par un nombre (ou une série de nombres). Ces données numériques sont stockées en mémoire, en général sur une puce électronique de la carte graphique. 60 fois par seconde, le contenu de cette mémoire est transféré à l'écran : c'est le « rafraichissement » de l'image.

Une image numérique se définit par trois données principales :

- Sa largeur mesurée en pixels,
- Sa hauteur mesurée en pixels,
- Le nombre maximal de couleurs que chaque pixel peut prendre.

Les images monochromes (ou bicolores) sont très faciles à stocker, puisqu'il suffit d'associer 1 bit à chaque pixel. Ces bits sont regroupés par 8 dans un octet. Un octet permet donc d'afficher 8 pixels bicolores. On peut à partir de ces données proposer des problèmes de conversion du type :

- Combien faut-il d'octets pour enregistrer une image bicolore de 1200 x 900 pixels ?

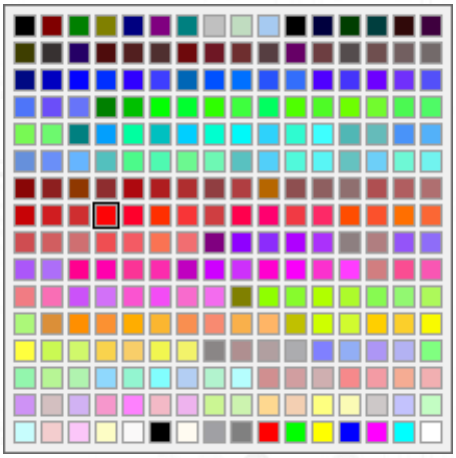
Le monde en couleurs

Les progrès technologiques ont permis progressivement d'afficher de plus en plus de couleurs, jusqu'à finir par supplanter la photographie argentique classique basée sur la chimie. Cela s'est fait par étapes :

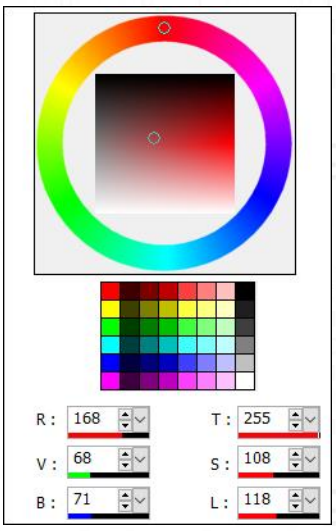
- Les premiers ordinateurs gérant la couleur (TO7, ORIC, MO5, etc.) n'affichaient que 8 couleurs et produisaient des images de même type que celle utilisée dans ce défi. Chaque pixel était stocké sur 3 bits et sa valeur pouvait donc varier de 0 à 7. Un circuit spécialisé convertissait ensuite ce nombre en une couleur particulière : noir, blanc, 3 couleurs primaires et 3 couleurs secondaires.
- La capacité de stockage des ordinateurs augmentant régulièrement, la palette des couleurs disponibles est progressivement passée à 16 (4 bits pas pixel), puis 256 (8 bits pas pixel).
- L'étape suivante a nécessité une refonte du système de codage car on ne pouvait agrandir indéfiniment la palette. Dans le système de codage appelé « vraies couleurs », la valeur du pixel ne fait plus référence à une palette de couleurs mais indique directement les niveaux de bleu, rouge et vert de la couleur à afficher. Chaque niveau est stocké sur un octet et peut donc varier de 0 à 255. Il faut trois octets de mémoire pour chaque pixel de l'image et l'on peut donc afficher 256 x 256 x 256 = 16777216 couleurs différentes, valeur souvent arrondie à « 16 millions ». Ce nombre est supérieur au nombre de couleurs que peut discerner un œil humain !

Palette 16 couleurs

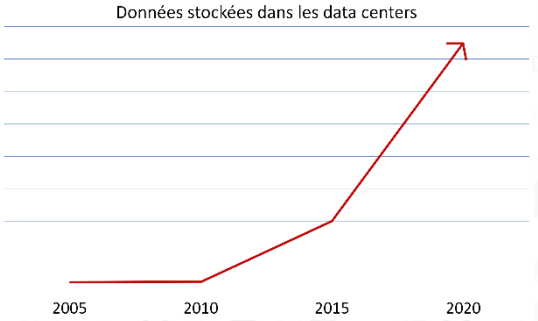
Palette 256 couleurs



Système RVB



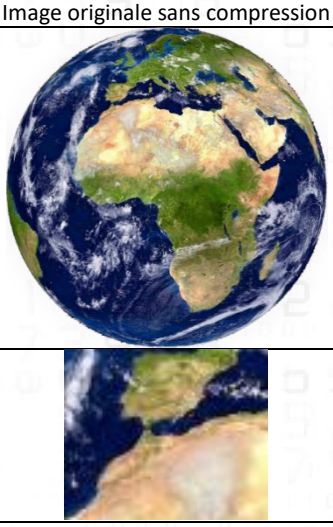
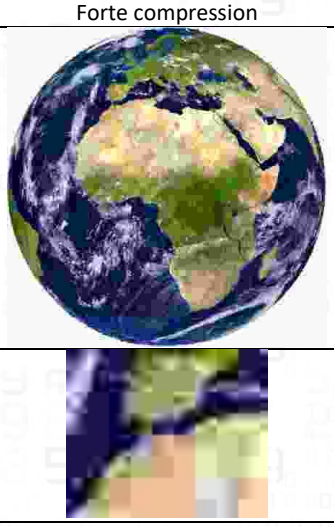
On manque de place



Avec l'augmentation exponentielle de la quantité de données stockées dans les data centers, la compression est devenue incontournable. L'algorithme découvert dans ce défi est simple, mais ne permet pas de gagner beaucoup d'espace pour le stockage d'une image. Il n'est efficace que si cette dernière comporte des plages de pixels de couleur identique, ce qui est rarement le cas d'une photographie par exemple. Mais il a l'avantage de ne pas modifier l'image. En décodant le fichier, on retrouve l'original.

On touche là le principal problème de la compression des données :

- Soit on veut conserver l'original et la compression sera limitée,
- Soit on veut gagner beaucoup de place, des données seront perdues.

Joconde.bmp Image originale sans compression	Joconde.png Compression sans perte	Joconde.jpg Compression moyenne	Joconde.jpg Forte compression
			
574 Ko	414 Ko	17 Ko	7 Ko

Le problème corollaire au stockage des données, c'est la consommation électrique et la chaleur induite qu'il faut dissiper dans l'atmosphère, ou bien dans les océans. Il est important de sensibiliser les élèves à ce problème. Voici quelques données chiffrées pour mesurer l'impact des centres de stockage sur l'environnement :

- 2,5 milliards de Go de données s'ajoutent quotidiennement et doivent être traités pour être disponibles.
- Le refroidissement d'un centre de stockage représente environ 40% de sa consommation énergétique. (Pour 100 euros dépensés, 40 servent uniquement à évacuer la chaleur produite par les installations !)
- L'ensemble des data centers représente 5% de la consommation énergétique mondiale, et augmente de 5% par an.
- Un data center de taille moyenne consomme entre 300 et 500 millions de litres d'eau par an, soit autant que 3 hôpitaux.
- La société Microsoft a testé récemment un centre de stockage immergé dans l'océan. Avantage, il n'a plus besoin d'être refroidi, donc la consommation d'électricité diminue. Mais c'est l'océan qui absorbe la chaleur produite par les ordinateurs du centre.
- Dans une entreprise, l'informatique représente en moyenne 25% de l'énergie consommée.

Le codage-décodage d'une image simple peut être adapté pour fonctionner en binôme : chaque élève crée une image puis la code. Il transmet ensuite son code à l'autre qui doit reconstituer l'image d'origine. Pour finir, les élèves échangent leurs images pour vérifier. (Cf. fiche page suivante)

0=blanc 1=noir 2=bleu 3=rouge 4=vert 5=jaune 6=orange 7=violet 8=gris

0=blanc 1=noir 2=bleu 3=rouge 4=vert 5=jaune 6=orange 7=violet 8=gris

0=blanc 1=noir 2=bleu 3=rouge 4=vert 5=jaune 6=orange 7=violet 8=gris

0=blanc 1=noir 2=bleu 3=rouge 4=vert 5=jaune 6=orange 7=violet 8=gris

Rétrécir un texte

La compression d'un texte doit se faire *sans perte de données*, afin de pouvoir le relire après décompression. Les algorithmes à dictionnaire sont très présents et vous les avez sûrement déjà utilisés (ZIP, RAR, etc.).
Cet exercice est l'occasion d'aborder simplement la notion d'évaluation d'une procédure. Le texte donné en exemple a été composé pour cela :

1. On mesure l'efficacité en comparant la quantité de nombres du texte avant et après la compression. Il ne faut pas oublier de compter le dictionnaire !

Texte d'origine	Dictionnaire	Texte compressé
chère maman on dit que la coccinelle est un petit porte-bonheur mais elle vole et s'envole ! tandis que moi, chère maman c'est bien mieux qu'une coccinelle je serai ton porte-bonheur en ce beau jour et pour toujours.	1 Chère 2 Maman 3 On 4 Que 5 Coccinelle 6 Est 7 Un 8 Porte-bonheur 9 Vole A Et B jour	@1 @2 @3 dit @4 la @5 @6 @7 petit @8 mais elle @9 @A s'en@9 ! tandis @4 moi, @1 @2 c'@6 bien mieux qu'@7e @5 je serai t@3 @8 en ce beau @B @A pour tou@Bs.
207 caractères	**64 caractères**	**145 caractères**

2. Le rendement n'est pas bon, il est même négatif puisqu'on passe de 207 à 209 caractères !
3. On cherche les causes du problème. Voici 3 déductions que les élèves peuvent facilement faire :
   - Coder de longs mots ou de longues séquences est plus intéressant car on « gagne » beaucoup de caractères.
   - Chaque code compte pour 2 caractères : le signe spécial plus le numéro dans le dictionnaire. Il ne sert donc à rien de coder des séquences d'1 ou 2 caractères : on ne « gagne » rien.
   - Notre texte est court : les mots ou les séquences se répètent peu. Avec un texte plus long, l'efficacité serait bien meilleure.

Attention aux erreurs !

Le principe du contrôle d'erreur est indissociable de l'informatique. Tous les transferts de données comportent systématiquement un code de contrôle obtenu en appliquant un algorithme particulier. Le plus simple et le plus utilisé est le « bit de contrôle ». Je ne l'ai pas choisi pour ce défi car il n'est pas adapté à cette forme d'exercice, mais vous pouvez l'aborder en classe sous la forme du jeu ci-dessous :

Matériel nécessaire

- Jeux de cartes collectif et individuels, papier, crayon, gomme. Pour réaliser le jeu de carte, imprimer recto-verso les deux pages en annexe, plastifiez puis découpez les cartes.

Compétences travaillées

- Tables de multiplication, parité des nombres, reste de la division euclidienne (modulo), division par 10
- Découverte et application d'un algorithme.

Déroulement de la séance

❖ *Observation :*

- Commençons par un tour de *magie* :
  - Demandez à un élève de disposer les cartes en carré de 5x5, en alternant les faces 0 ou 1 de façon aléatoire.
  - « Pour compliquer encore », ajoutez vous-même une sixième ligne et une sixième colonne.

0	1	0	1	0
1	1	0	0	1
0	1	1	1	0
0	1	0	1	0
0	1	1	1	0

0	1	0	1	0	0
1	1	0	0	1	1
0	1	1	1	0	1
0	1	0	1	0	0
0	1	1	1	0	1
1	1	0	0	1	1

  - Les cartes ajoutées ne sont bien sûr pas choisies au hasard : *elles indiquent la parité du nombre situé sur la ligne ou la colonne correspondante.*
  - Fermez les yeux ou tournez-vous dos au tableau, puis demandez à un élève de retourner une carte au hasard.
  - Vous retrouvez instantanément la carte retournée en comptant le nombres de 1.

❖ *Recherche collective :*

- Quel est le « truc » utilisé pour réaliser ce tour ?
  - Les élèves devinent immédiatement que vous n'avez pas disposé vos cartes au hasard.
- Qu'est-ce qui a changé dans la ligne et la colonne de la carte retournée ?
  - Le nombre de 1 et de 0.
- Cela permet-il d'identifier la case ?
  - Non, à moins d'avoir mémorisé toute la grille, ce qui n'est pas le cas.
- Comptons les nombres de 1 pour chaque ligne et chaque colonne
  - Les nombres sont tous pairs, sauf pour la ligne et la colonne de la carte retournée !

0	1	0	1	0	0	2
1	1	0	0	1	1	4
0	1	0	1	0	1	3
0	1	0	1	0	0	2
0	1	1	1	0	1	4
1	1	0	0	1	1	4
2	6	1	4	2	4	

❖ *Application :*

- Les élèves en binômes font quelques parties pour bien assimiler l'algorithme.

Cet algorithme couramment utilisé en informatique s'appelle « bit de parité ». Il consiste à placer à la fin du nombre binaire un bit supplémentaire de façon à obtenir un nombre pair de 1. Ensuite le nombre est transmis. A l'arrivée, on vérifie le nombre de 1 qui doit être pair. Dans le cas contraire le transfert n'est pas bon, il faut le recommencer.
Pour tester la validité d'un seul et unique nombre, cet algorithme n'est pas très performant. En effet, il suffit de modifier 2 bits du nombre et la parité reste inchangée !

Dans l'exercice précédent, nous croisons deux informations qui nous permettent de localiser précisément l'erreur. Cette localisation est nécessaire pour notre tour de magie, mais dans un contrôle de parité effectué lors d'un transfert de données par l'ordinateur, *il n'est pas important de savoir où se trouve l'erreur*. Il suffit juste de savoir qu'une erreur est présente pour renouveler le transfert dans ce cas.

❖ *Recherches complémentaires :*

1. Si je retourne 2 cartes, l'erreur peut-elle passer inaperçue ? (Non)
2. Si je retourne 3 cartes, l'erreur peut-elle passer inaperçue ? (Non)
3. Si je retourne 4 cartes, l'erreur peut-elle passer inaperçue ? (Oui)
4. Cherchez des exemples de retournement de 4 cartes qui passeront inaperçus.

L'algorithme de vérification des codes à barres utilise l'arithmétique modulaire. Si vous avez utilisé la « preuve par 9 » à l'école, vous avez fait de l'arithmétique modulaire. Cette branche de l'arithmétique, qui s'intéresse aux restes de la division euclidienne, est très utilisée en informatique et plus particulièrement en Cryptologie où elle sert principalement à créer des fonctions qui ne sont pas inversibles. Exemple :

5. Pour me connecter à mon serveur d'entreprise, je dois envoyer mon mot de passe sur le réseau. Mais il pourrait être intercepté !
6. Avant de l'envoyer, mon logiciel lui applique une « fonction modulo » et envoie le résultat.
7. Si le résultat est intercepté, il est impossible de retrouver mon mot de passe car la fonction n'est pas inversible.
8. A l'arrivée le serveur, qui connaît mon mot de passe, lui applique la même fonction et compare le résultat avec mon envoi.

Le célèbre algorithme de cryptage RSA, aussi appelé « cryptage à clef publique/clé privée », utilise une fonction modulo particulière qui ne peut être inversée qu'avec la clef privée :

9. Pour écrire à Louise, je crypte son message avec sa clé publique que tous ses correspondants connaissent.
10. Pour décrypter le message, Louis utilise sa clé privée qui seule permet d'inverser la fonction modulo utilisée lors du cryptage.

Le RSA et sa fonction modulo ont révolutionné la sécurité de tous les échanges mondiaux. Auparavant, les deux correspondants devaient échanger une clef avant de pouvoir communiquer en secret. On parlait de cryptage symétrique. Cette clef devait être souvent changée, et pouvait être interceptée. Avec le RSA chaque correspondant possède une clé publique connue de tous, et une clef privée connue de lui seul. Plus d'échange de clefs ni de risque d'interception.

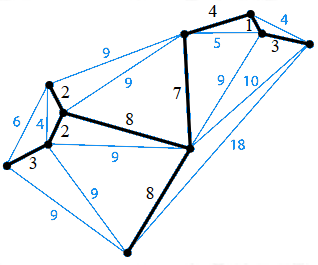
Si la cryptologie vous intéresse, je vous conseille le livre de Simon Singh : *Histoire des codes secrets*.

Recherche rapide

L'algorithme de recherche binaire n'est pas le seul utilisé en informatique, mais il est simple à mettre en œuvre, et surtout : c'est le plus rapide ! Cette rapidité de recherche se paie toutefois par une perte de temps lors de l'ajout de nouvelles données. Celles-ci doivent être intercalées au bon endroit pour respecter l'ordre croissant sans lequel le tri rapide ne fonctionne plus. Cela peut s'avérer complexe, mais les programmeurs ont plus d'un tour dans leur sac :

11. Imaginons que vous vouliez enregistrer stocker tous les titres des livres de votre bibliothèque dans une base de données.
12. A chaque ajout d'un nouveau titre, va-t-il falloir déplacer tous les titres situés après dans l'ordre alphabétique pour l'intercaler au bon endroit ?
13. La solution consiste à créer des index. Chaque titre sera associé à un numéro : un nombre qui augmente d'une unité à chaque nouveau titre.
14. Ce sont les index qui sont triés à la place des titres. Déplacer un nombre est beaucoup plus rapide que de décaler tout un texte !

La ville embourbée



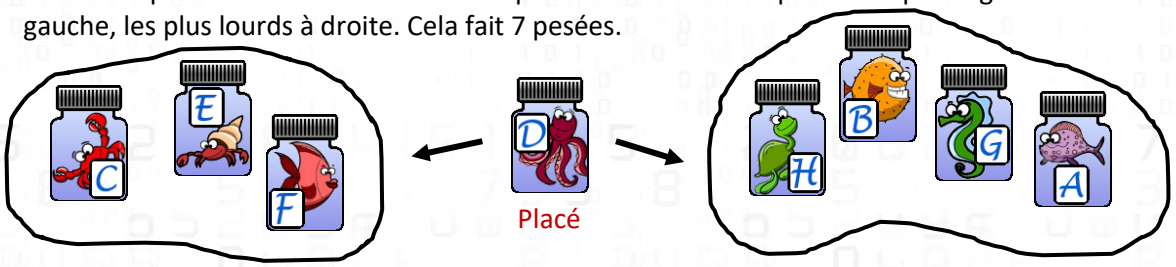
C'est bien connu, on réfléchit mieux avec un bon schéma. L'objectif de cet exercice est d'apprendre à représenter schématiquement une situation problème pour faciliter la recherche de la solution. Le problème présenté ici est emprunté et adapté de la publication « Computer Science Unplugged » et reprend le thème du réseau routier. Il peut facilement être adapté à d'autres types de réseau : distribution d'eau, d'électricité, etc. La technique des « arbres couvrants » est très utilisée en informatique pour résoudre des problèmes de la vie courante.

Ce type re problème ne comporte pas de solution unique : les élèves trouveront sans doute de nombreuses réponses équivalentes. On peut pousser la réflexion : existe-t-il une méthode pour obtenir une solution optimale ? Joseph Kruskal, un mathématicien américain, en a proposé une en 1956, connue sous ne nom d'*Algorithme de Kruskal* et d'une simplicité qui le met à portée d'un élève de primaire : *il suffit de sélectionner les arêtes par ordre croissant de leur longueur*. On commence par les plus petites et on complète petit à petit, en gardant à l'esprit que *le nombre optimal d'arêtes est égal au nombre nœuds moins un*.

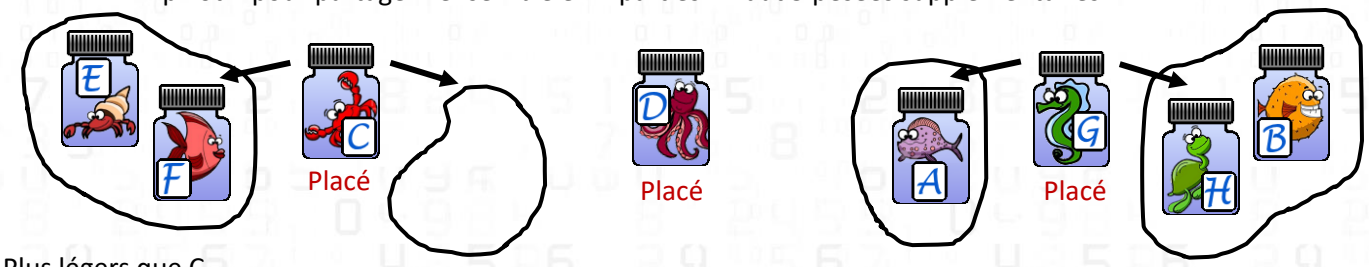
Range-moi tout ça !

L'algorithme du « *tri à bulle* » présenté ici est facile à comprendre. Mais il n'est plus utilisé car très lent ! Pour trier les 8 pots avec cet algorithme, il faut effectuer 7 + 6 + 5 + 4 + 3 + 2 + 1 = 29 comparaisons. Les programmes informatiques utilisent un autre algorithme, le *tri rapide*, dont le fonctionnement est à rapprocher de l'algorithme de recherche binaire découvert précédemment. Vous pouvez l'aborder en classe de façon ludique et collective, comme ceci par exemple :

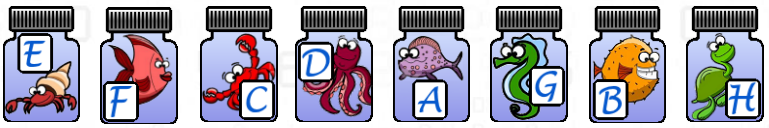
1. Présenter 8 pots réalisés en récupérant des tubes de comprimés qui auront été remplis d'une quantité aléatoire d'eau ou de sable.
2. Choisir un pot au hasard : tous les autres pots vont lui être comparés. Les plus légers seront mis à gauche, les plus lourds à droite. Cela fait 7 pesées.



3. Recommencer le tri avec chaque groupe, en choisissant à chaque fois au hasard un pot qui servira de « pivot » pour partager l'ensemble en 2 parties. Il faut 5 pesées supplémentaires



4. Recommencer une dernière fois avec les deux sous-ensembles de pots restant à départager, ce qui nécessite dans ce cas 2 pesées supplémentaires. Les pots sont maintenant triés du plus léger au plus lourd. Nous avons effectué 7 + 5 + 2 = 14 pesées, soit 2 fois moins qu'avec l'algorithme du tri à bulle ! Le *tri binaire*, ou *tri rapide* est vraiment très efficace.



La fourmi et le pot de miel

Les nouveaux programmes 2016 pour les cycles 2, 3 et 4 intègrent l'apprentissage des algorithmes de programmation. Quelques extraits :

Au cycle 2 :

- Mettre en œuvre un **algorithme** de calcul posé pour l'addition, la soustraction, la multiplication
- Au CP, la représentation des lieux et le **codage des déplacements** se situent dans la classe ou dans l'école, puis dans le quartier proche, et au CE2 dans un quartier étendu ou le village.
- Dès le CE1, les élèves peuvent **coder des déplacements à l'aide d'un logiciel de programmation adapté**, ce qui les amènera au CE2 à la **compréhension, et la production d'algorithmes simples**.
- Réaliser des déplacements dans l'espace et les **coder** pour qu'un autre élève puisse les reproduire.
- **Programmer les déplacements** d'un robot ou ceux d'un personnage sur un écran."

Au cycle 3 :

- Pratiquer des langages : Exploiter un document constitué de divers supports (texte, schéma, graphique, tableau, **algorithme simple)**.

- **En CM1 et CM2** on se limitera aux signaux logiques transmettant une information qui ne peut avoir que deux valeurs, niveau haut ou niveau bas.
- En classe de **sixième**, **l'algorithme en lecture introduit la notion de test** d'une information (vrai ou faux) et l'exécution d'actions différentes selon le résultat du test."
- Les élèves découvrent **l'algorithme** en utilisant des **logiciels d'applications visuelles et ludiques**. Ils exploitent les moyens informatiques en pratiquant le travail collaboratif.
- **Initiation à la programmation** : Une initiation à la programmation est faite à l'occasion notamment **d'activités de repérage ou de déplacement** (programmer les déplacements d'un robot ou ceux d'un personnage sur un écran), ou **d'activités géométriques** (construction de figures simples ou de figures composées de figures simples).

« La fourmi et le pot de miel » propose 5 activités de codage et de décodage, de difficulté croissante.

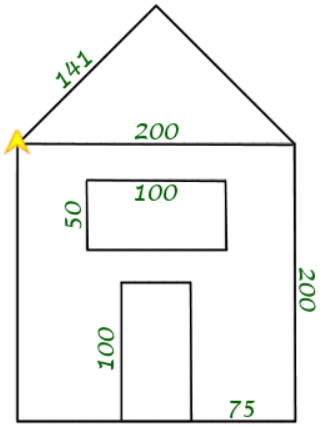
Activité	Cycle	Description
1	2	- Codage simple des déplacements. - La suite des instructions s'affiche au fur et à mesure. - Repère centré sur l'élève.
2	2	- Codage simple des déplacements. - La suite des instructions s'affiche au fur et à mesure. - Repère centré sur la fourmi (déplacements relatifs) - Introduction de la direction du déplacement
3	2 & 3	- Décodage d'un algorithme (appliquer une série de déplacements) - Repère centré sur l'élève. - 3 niveaux de difficulté (longueur de l'algorithme)
4	3	- Décodage rétroactif d'un algorithme (trouver la source d'une série de déplacements) - Repère centré sur l'élève. - 3 niveaux de difficulté (longueur de l'algorithme)
5	3	- Création et test d'un algorithme - Repère centré sur la fourmi (déplacements relatifs)

La tortue qui savait dessiner

La langage LOGO, créé en 1966, est fortement inspiré des travaux de Piaget sur le développement cognitif de l'enfant. En opposition aux thèses behavioristes qui prédominent à l'époque, LOGO propose une approche constructiviste : l'élève ne doit plus apprendre en répétant mais devenir acteur de ses apprentissages. L'ordinateur est là pour l'aider à établir des procédures et à en vérifier le résultat. Pour cela, il dispose d'une tortue, sorte de robot dont il va pouvoir programmer les déplacements en s'appuyant sur la connaissance intuitive de ses propres mouvements. Il va ainsi être amené à émettre des hypothèses puis à les vérifier. Pour reproduire une figure complexe, il devra la décomposer en éléments plus petits, repérer des redondances, etc.
LOGO est un langage complet et complexe qui ne se limite pas à la tortue qui l'a rendu célèbre. Cette tortue est d'ailleurs présente dans d'autres langages de programmation.

Les liens entre la tortue LOGO et la géométrie sont évidents, mais tout de même assez réduits à l'école primaire, principalement à cause de la mesure des angles qui n'est pas au programme. On peut tout de même mettre en place quelques activités de renforcement des connaissances, comme la reproduction d'une figure complexe. Pour reproduire la figure ci-contre, l'élève va mettre en œuvres plusieurs compétences :

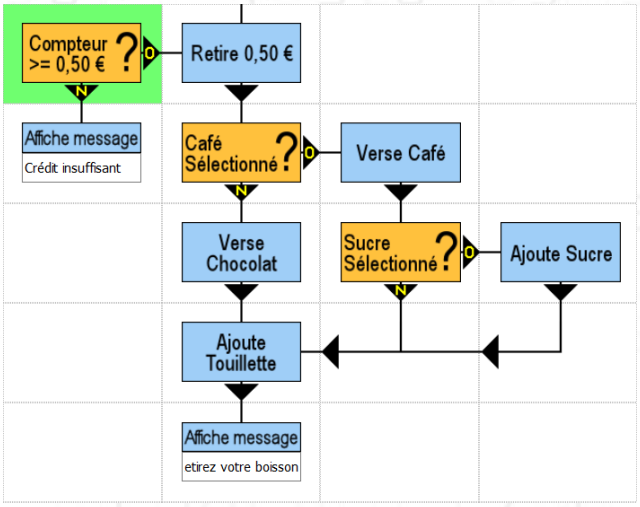
- La décomposer en figures simples et connues : un carré, un triangle isocèle et deux rectangles.
- Repérer le « point de départ » et « l'orientation de départ » de chaque sous-figure.
- Déduire les longueurs manquantes à partir des propriétés des différentes figures.



Un café s'il vous plait !

L'informatique a envahi presque tous les objets du quotidien.
La programmation d'un automate semble donc un bon moyen d'ancrer l'informatique dans le réel. Malheureusement, les applications pratiques de programmation proposées aux élèves de primaire se résument généralement aux déplacements d'un petit robot. L'application « Scratch » offre une timide ouverture en proposant de manier des sons et des images, pour la réalisation de jeux par exemple. Mais elle est tout de même très centrée sur les déplacements et le langage utilisé reste complexe.

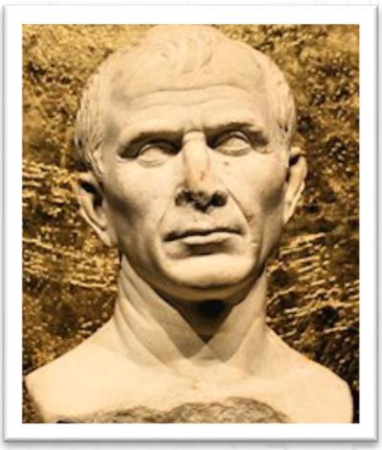
Le « distributeur de boissons » est une tentative d'initiation à la programmation, en lien avec un automate que tous les élèves connaissent. Les choix sont limités et les instructions ont été volontairement simplifiées pour se concentrer sur la construction logique d'un programme informatique. Pas de déplacements, d'angles, ni de calculs. Juste quelques instructions simples à ordonner correctement, et permettant de nombreuses solutions différentes au problème posé. En voici une ci-contre qui tient compte de toutes les options.



Si vous travaillez ce défi en ateliers ou individuellement, il sera intéressant de comparer les différents programmes composés par les élèves et les tester collectivement pour dénicher les erreurs ou les oublis. Ce sera l'occasion de vérifier que des démarches différentes peuvent aboutir au même résultat et que l'imagination est bien le propre de l'homme.

Christian.vinent@ac-poitiers.fr

Le chiffre de César



Pour communiquer avec les nombreuses légions réparties sur son empire, Jules César utilisait un code très simple : le décalage alphabétique. Puis il l'envoyait porter par un messager. Si ce messager était capturé, l'ennemi trouvait un message illisible.

Pour comprendre le chiffre de César, tu dois :
- connaître l'alphabet
- savoir additionner et soustraire mentalement

Supposons que l'on ait choisi un décalage de 7 lettres vers l'avant (+7) :
Le A devient donc un H, le B un I, le C un J, etc.
On peut résumer le décalage (+7) dans un tableau comme celui-ci :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G

Pour décoder le message, le destinataire devra effectuer l'opération inverse : décaler de 7 lettres vers l'*arrière* (-7).
Voici 1 message à décoder. Il a été codé avec un décalage de (+4)
Attention, pour renforcer le cryptage et éviter que l'on reconnaisse les mots d'après leur nombre de lettres, celles-ci ont été regroupées par 5. Vous devrez donc remettre aux bonnes places les espaces séparant les mots, ainsi que la ponctuation, de façon à obtenir une phrase claire.

Message n° 1 : **AMPPM EQPIK VERHL EFMXI WYVPM PIHIW YPPMZ ER**

Texte en clair :

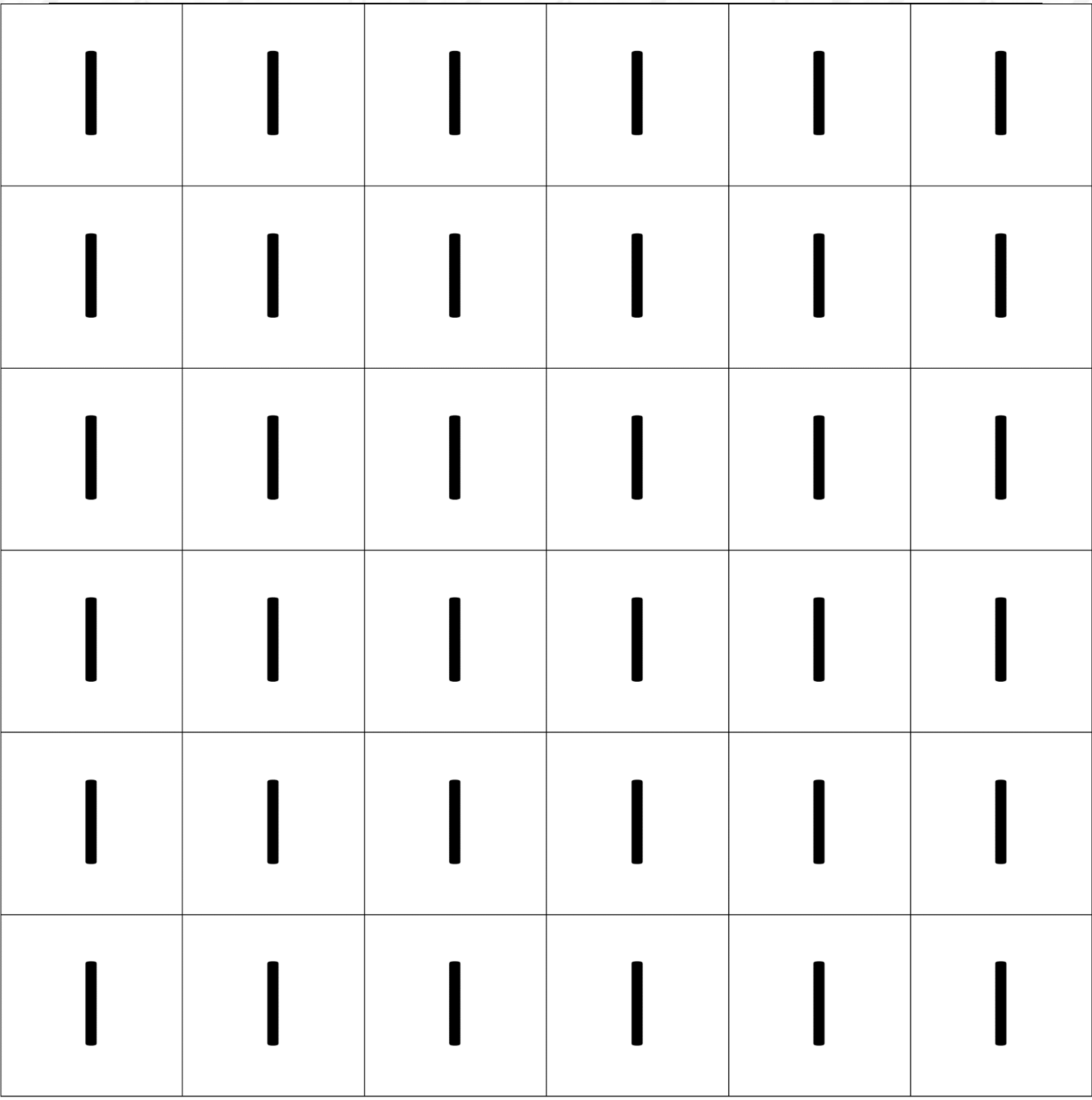
Cette méthode, bien qu'efficace à l'époque de Jules César, n'est pourtant pas très sûre. On peut assez facilement décrypter un message codé par décalage alphabétique, sans connaître le décalage utilisé, en se basant sur la **fréquence des lettres.**
Cela se fait en 3 étapes :

1. La lettre la plus utilisée en français est le **E**. Il faut donc rechercher la lettre la plus fréquente dans le message.
2. Cette lettre remplaçant le E, on peut calculer le *décalage*.
3. Il suffit ensuite d'appliquer le même décalage à toutes les lettres du message.

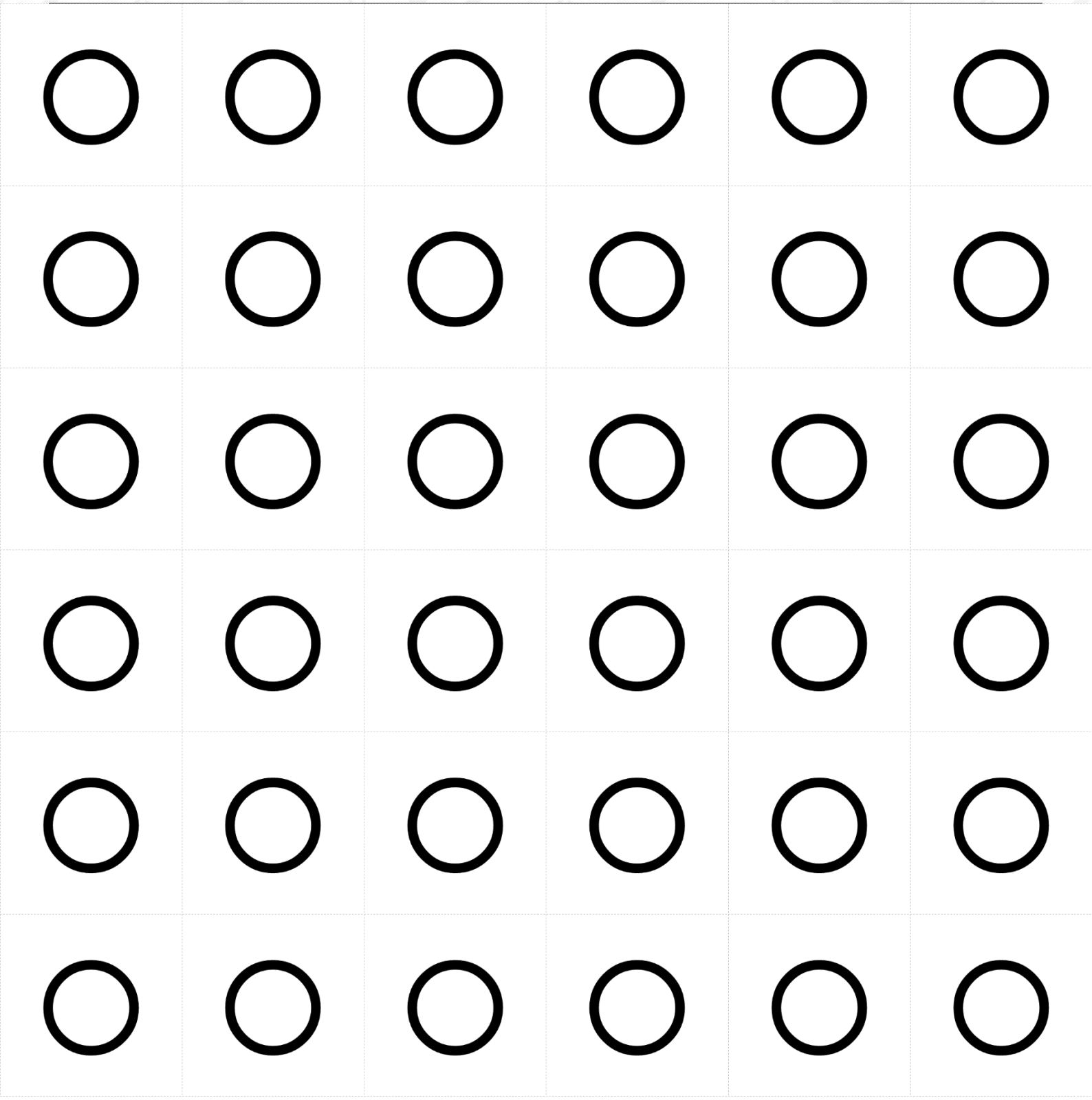
Le message suivant a été crypté avec le chiffre de César. A vous de le décoder.

OZQJX HJXFW KZYJR UJWJZ WIJXL FZQJX

Texte en clair :



Imprimez recto-verso avec la page suivante, plastifiez puis découpez.



Bibliographie

- Prim à bord : le portail du numérique pour le premier degré : coder programmer
  http://eduscol.education.fr/primabord/codage
- Computer Science Unplugged
  http://csunplugged.org/
- Comment apprendre le code informatique à l'école :
  https://culturenum.info.unicaen.fr/blogpost/bgkw9d4gp2c/view
- 1, 2, 3, Codez ! Sur le site de « La main à la pâte »
  http://www.fondation-lamap.org/123codez
- 7 activités « informatique débranchée »
  http://www-irem.ujf-grenoble.fr/spip/spip.php?article146
- Lightbot, programmation de déplacements :
  https://lightbot.com/flash.html
- Tuxbot, idem, application à installer pour Windows ou Androïd :
  http://appli-etna.ac-nantes.fr:8080/ia53/tice/ressources/tuxbot/index.php
- Scratch :
  En ligne : https://scratch.mit.edu/projects/editor/?tip_bar=getStarted
  Installation hors ligne : https://scratch.mit.edu/scratchr2/static/sa/Scratch-456.0.4.exe
  Scratch Junior (application tablette) : https://www.scratchjr.org/
- Le robot Thymio :
  https://www.thymio.org/fr:thymio