human processor!

Manuel Enseignant

LIVRET 1

Tutoriel

HEP Lausanne - Juin 2023

http://human-processor.xyz

Livret 1 - Tutoriel

Le dispositif didactique débranché "*human processor!*" et ses exercices sont inspirés du jeu "*human resource machine*" développé en 2015 par Kyle Gabler, Allan Blomquist et Kyle Gray de la société Tomorrow Corporation.

Les plans des pièces en bois de la boîte de jeu, le présent manuel ainsi que la liste des exercices et les fiches des stratégies cognitives, sont disponibles sur :
http://human-processor.xyz

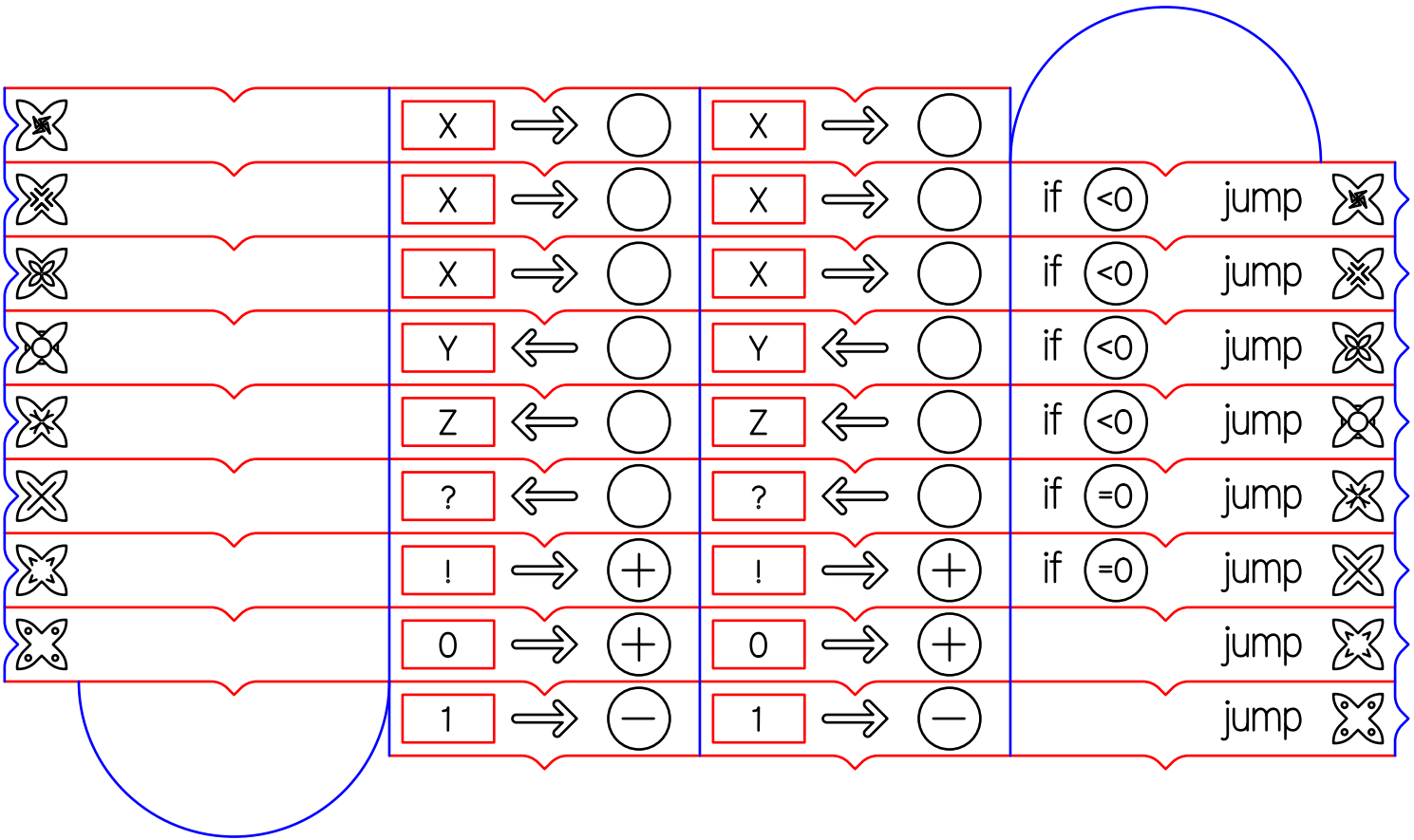
Le présent manuel propose, en alternance :

- 6 activités débranchées (hors tutorial) sur le thème de l'algorithmie qui utilisent les pièces en bois pour programmer ;
- 8 activités réflexives ayant pour but d'aider les enseignants à développer l'intelligence algorithmique[1] de leurs élèves.

Ce manuel est accessible au format numérique sur Google Doc. Il est possible de faire des commentaires directement sur le document en ligne :
https://bit.ly/manuel_enseignants

Pour toute information complémentaire :
christian.blanvillain@hepl.com

[1] Nous définissons l'intelligence algorithmique comme étant le produit des trois intelligences logico-mathématique, pratique et créative (**Robert J. Sternberg.** 2015. *Beyond I.Q.: A triarchic theory of human intelligence.* New York: Cambridge University Press.)


X
X
X
Y
Z
?
!
0
1
X
X
X
Y
Z
?
!
0
1
if <0 jump
if <0 jump
if <0 jump
if <0 jump
if =0 jump
if =0 jump
jump
jump

00. Métacognition

L'idée est de partir des procédures que les élèves maîtrisent pour les faire réfléchir sur les enjeux pédagogiques et les savoirs à apprendre qui se cachent derrière les procédures afin de les initier à la métacognition. L'activité proposée[1] se déroule en quatre étapes.

1. On commence par proposer aux élèves deux schémas côte-à-côte (figure 0), présentant d'une part l'architecture d'un ordinateur avec le nom des composants et de leur rôle et d'autre part le même schéma vide. L'élève a pour consigne de compléter le schéma vide.

2. On distribue ensuite le même exercice, mais avec les deux schémas au recto et au verso de la feuille au lieu d'être côte-à-côte. Une contrainte supplémentaire est fournie : il n'est autorisé de retourner la feuille qu'une seule et unique fois lorsque l'on aura commencé l'exercice.

3. L'élève a ensuite pour consigne de réfléchir seul et de noter par écrit les différences qu'il a identifiées entre la première activité et la seconde, et à l'incidence de ce changement de consigne sur ses apprentissages.

[1] Inspirée de **Greta Pelgrims** et **Sylvie Cèbe**, "Aspects motivationnels et cognitifs des difficultés d'apprentissage" dans Crahay, M. Dutrévis, M.. (2015). *Psychologie des apprentissages scolaires*. Bruxelles: De Boeck Supérieur.

4. La dernière étape consiste à mettre en commun les réflexions individuelles. Après avoir fait cette mise en commun, on demande aux élèves quel est, selon eux, l'intérêt de réfléchir sur leurs manières d'apprendre ? Présenter à ce moment le concept de métacognition et expliquer que cette attitude réflexive conduit à une meilleure connaissance de soi qui va leur permettre, en particulier, d'apprendre comment ils font pour apprendre et qu'en faisant ça, ils deviendront plus performants.

Pour réaliser la première tâche, les élèves n'ont pas besoin de solliciter leur mémoire court terme puisqu'il suffit de recopier les informations sans forcément chercher à les mémoriser. C'est cette attitude passive que l'on cherche à dénoncer. S'il est nécessaire que l'enseignant impose de ne pas retourner la feuille pour que l'élève se mette à apprendre, alors que le savoir est déjà là, prêt à être appris dès le premier exercice, que se passe-t-il pour tous les autres savoirs à apprendre que l'enseignant n'explicite pas ? La bonne attitude proactive est de toujours se demander quels sont les savoirs en jeu et de se donner soi-même des objectifs d'apprentissage qui semblent pertinents.

En programmation c'est pareil : il ne suffit pas de se contenter de comprendre un code qui marche pour apprendre à coder. Il faut aussi mobiliser son attention sur les gestes mentaux qui vous ont conduit à concevoir et écrire la solution. Cette attitude métacognitive vous permet de prendre conscience de votre intelligence algorithmique et va vous aider à la développer. C'est ce que nous allons travailler durant toute la durée de cette formation.

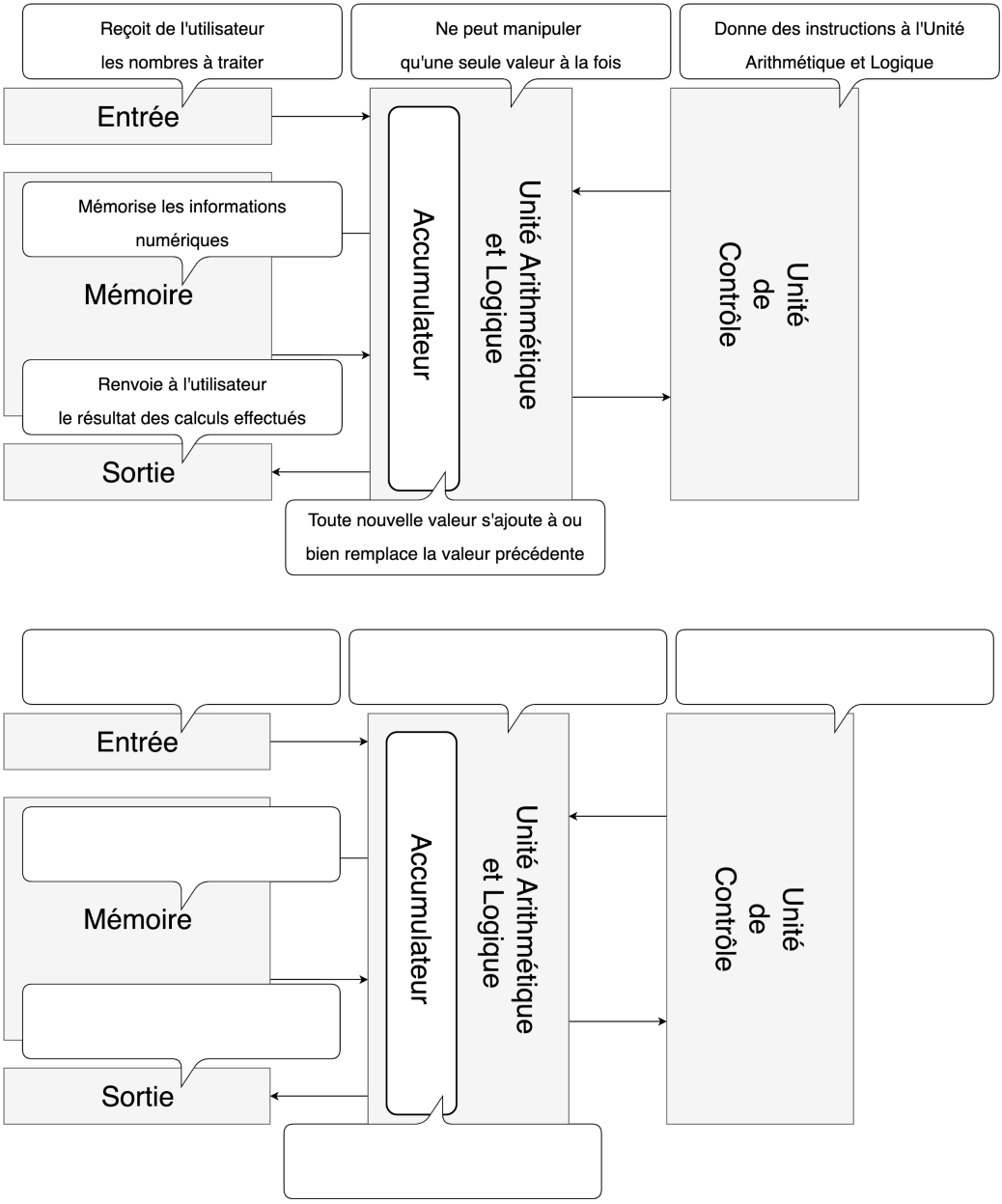


Fig. 1 - Architecture de Von Neumann

01. Architecture de Von Neumann

L’analyse vidéo des causes d’échecs des élèves en difficulté, nous ont permis de comprendre que la durée d’attention d’un élève TDHA est de l’ordre d’une minute et que le décrochage se fait déjà durant l’écoute des consignes de la première heure d’activité qui sert à se construire les concepts et les représentations mentales sur lesquelles vont être bâtis tout le reste de la formation. C’est pourquoi, nous vous proposons une courte accroche spécifique pour cette formation, à lire à vos élèves avant le début du premier cours :

« *Nous sommes tous capables d'apprendre. Il y a des différences dans la vitesse de nos apprentissages, mais il n'y a pas de différence entre nous dans notre aptitude à apprendre. Nous savons comment fait le cerveau pour apprendre et c'est très simple : il suffit de vouloir. Un élève qui veut apprendre va essayer suffisamment de fois pour que son cerveau finisse par créer des chemins qui rendront sa tâche plus facile. C'est le temps de fabrication de ces chemins qui varie selon les élèves. Ce message s'adresse à ceux qui mettent du temps à créer de nouveaux chemins dans leurs cerveaux. En informatique, la première minute d'explication est la plus importante et la première activité est la plus importante.* ***Car tout est construction.*** *Soyez curieux ! Offrez-vous une première minute d'écoute, concentrez-vous particulièrement durant cette première leçon, et vous pourrez découvrir l'étrange beauté des algorithmes.* »

L'action précède la conceptualisation. C'est donc en faisant que l'élève va, petit à petit, se construire une représentation mentale de ce qu'il manipule. À terme, lorsque les concepts liés aux objets manipulés ont été acquis, il pourra les manipuler avec son esprit, anticiper et planifier ses actions avant de les faire en vrai. C'est le stade que nous aimerions atteindre avec chaque élève d'ici à la fin de cette formation.

L'activité initiatique du micro-monde[1] qui va nous servir d'environnement didactique tout au long de cette formation, reprend les principaux éléments architecturaux d'un processeur. Nous faisons jouer aux élèves les rôles des composants de l'architecture de Von Neumann (Fig. 1) : entrées, sorties, mémoire, unité arithmétique et logique (avec son accumulateur), unité de contrôle.

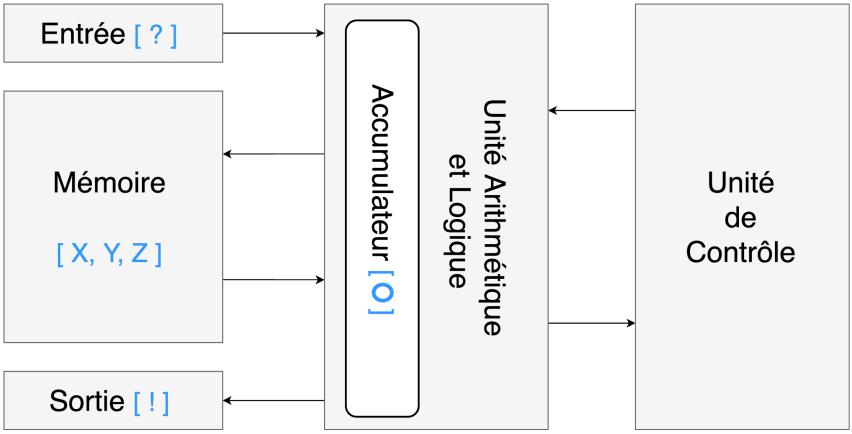


Fig. 2 - Architecture de Von Neumann

[1] **Papert**, S. (1981). *Jaillissement de l'esprit.* Paris : Flammarion.

Les élèves sont répartis en groupes de 4 dans l'espace de la salle de classe. Chaque groupe reçoit une fiche avec le nom du composant dont il doit jouer le rôle. Sur cette fiche sont précisés le rôle et les actions possibles. Après avoir choisi de jouer le rôle d'un des composants de l'architecture, lisez attentivement les actions possibles et attendez de recevoir un problème à résoudre pour commencer. Il y a toujours plusieurs manières de résoudre un problème donné. On préfèrera celle qui prend le moins d'étapes, c-à-d ayant le plus petit "quand" possible à la fin de l'algorithme.

Constituer des groupes de 4 personnes :

- L'unité de contrôle
- La mémoire [X], [Y], [Z]
- L'entrée [?]
- La sortie [!]

Matériel :

- Une assiette blanche représentant l'accumulateur de l'unité arithmétique et logique.
- Un feutre effaçable + de quoi écrire sur papier et tableau.

Ces composants, mis ensemble, constituent une architecture de Von Neumann (Fig. 1). Cette architecture a été inventée pendant la 2ème guerre mondiale par le mathématicien **Alan Turing** et proposée après la guerre par le mathématicien **John Von Neumann** pour construire l'un des premiers ordinateurs. Elle est toujours utilisée dans les processeurs d'ordinateurs de nos jours (2020).

La mémoire, l'entrée et la sortie ne peuvent pas s'échanger des informations directement. Elles doivent utiliser l'accumulateur pour s'échanger une donnée (un nombre). L'accumulateur est représenté par un disque blanc (une assiette) qui ne peut contenir qu'une seule valeur, effaçable à chaque fois qu'il est modifié. L'accumulateur circule sur la table et est échangé entre la mémoire, l'entrée et la sortie. Au début, il est posé au milieu de la table. Le groupe doit alors déterminer ce qui doit être fait et dans quel ordre, pour résoudre les problèmes proposés.

L'enseignant déterminera les problèmes à résoudre, en fonction du niveau des élèves et du temps disponible. Les exercices *photocopieur* et *inverseur* sont adaptés pour de jeunes élèves. Les exercices *additionneur*, *doubleur*, *quadriplicateur* sont adaptés pour des élèves d'âge moyen. Avec des étudiants familiers avec l'informatique, on peut aller jusqu'à *l'octoplicateur* et au *décaplicateur*, de manière à pouvoir présenter l'algorithme de la *multiplication égyptienne*.

L'unité de contrôle

Rôle :

- Elle synchronise les échanges entre les différents composants de l'architecture de Von Neumann.
- Elle s'assure que toutes les opérations effectuées sont licites et se déroulent dans un ordre logique.
- Elle note toutes les opérations entre les différents composants de l'architecture de Von Neumann (mémoire, entrée, sortie) et l'accumulateur.

Actions :

- Elle utilise un tableau avec quatre colonnes :

  quand, un compteur qui numérote chacune des interactions du système / chaque ligne du tableau

  quel composant de l'architecture de Von Neumann : mémoire [X], [Y], [Z], l'entrée [?], la sortie [!]

  quelle interaction :
  - lire une valeur dans l'accumulateur →O
  - écrire la valeur de l'accumulateur ←O

  quelle valeur est dans l'accumulateur

L'entrée → `représentée par [ ? ]` *(on s'interroge)*

Rôle :

- Elle simule les interactions d'un humain sur un clavier, en générant des nombres aléatoires entre 0 et 255.
- Elle décide, avec la mémoire et la sortie, de la prochaine opération à réaliser avec l'accumulateur et en informe l'unité de contrôle (UC).
- Elle vérifie que l'historique des valeurs produites est bien notée dans l'accumulateur par l'UC.

Actions :

- Elle recopie une valeur différente à chaque fois dans l'accumulateur (en remplaçant la valeur précédente).
- Elle ajoute une valeur différente à chaque fois à la valeur déjà présente dans l'accumulateur (sans la remplacer).

La sortie → `représentée par [ ! ]` *(on s'exclame)*

Rôle :

- Elle simule les interactions d'une imprimante en affichant le résultat des calculs.
- Elle décide, avec l'entrée et la mémoire, de la prochaine opération à réaliser avec l'accumulateur et en informe l'unité de contrôle (UC).
- Elle vérifie que l'historique des valeurs reçues en provenance de l'accumulateur, est bien notée par l'UC.

Actions :

- Elle peut lire la valeur contenue dans l'accumulateur.

La mémoire → représentée par [X], [Y], [Z]

Rôle :

- Elle conserve jusqu'à trois valeurs en mémoire nommées X, Y, Z.
- Elle décide, avec l'entrée et la sortie, de la prochaine opération à réaliser avec l'accumulateur et en informe l'unité de contrôle (UC).
- Elle vérifie que l'historique des valeurs envoyées vers et reçues de l'accumulateur, est bien conforme à ce qui est notée par l'UC.

Actions :

- Elle peut lire la valeur contenue dans l'accumulateur et la conserver dans une de ses mémoires.
- Elle peut recopier la valeur d'une de ses mémoires dans l'accumulateur en remplaçant la valeur précédente.
- Elle peut ajouter la valeur d'une de ses mémoires à la valeur déjà présente dans l'accumulateur (sans la remplacer).

En fin d'activité, nous présentons l'interface du jeu *Human Resource Machine*[2] (Fig. 2) et du petit personnage qui déplace les valeurs numériques dans sa main. C'est une étape fondamentale car elle permet de renforcer le modèle mental que les élèves élaborent en réalisant ce jeu de rôle. L'analogie "prendre la valeur dans la main" sera utilisée par la suite durant toute la durée de la formation. Des renvois explicites à cette représentation de l'échange des informations entre les composants, sont fait durant toute la durée de la formation et aident à bien comprendre la dynamique du dispositif débranché.

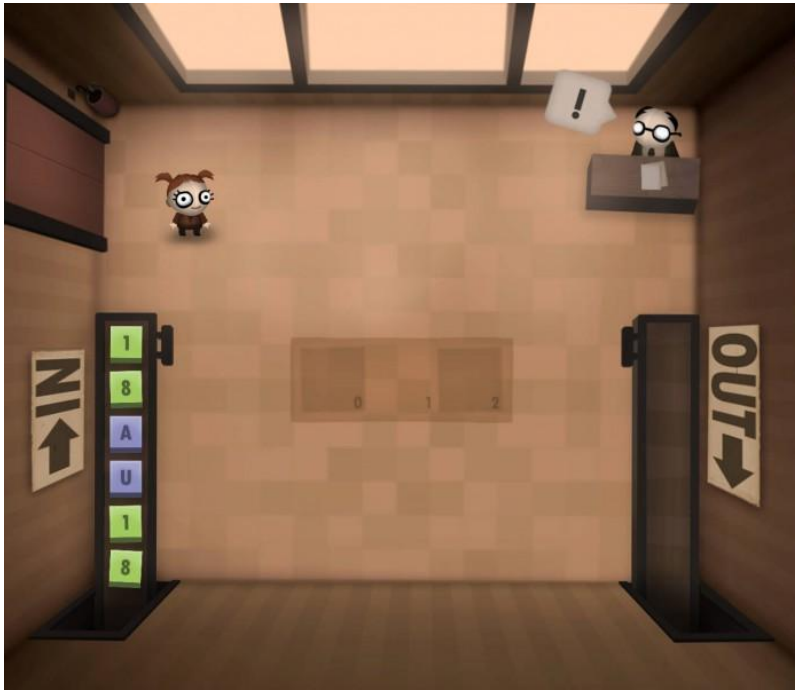


Fig. 3 - Interface du jeu Human Resource Machine

[2] Version (limitée) gratuite : http://tomorrowcorporation.com/human-resource-machine-hour-of-code-edition

02. Tutorial

Les premiers exercices introduisent les instructions de base du langage qui sont celles utilisées pour le recopiage d'une valeur d'une source vers une destination (pièces courtes). Les exercices suivants présentent la seconde instruction du langage qui est le saut dans la séquence des instructions (pièces longues).

Dans l'activité précédente nous prenions une assiette dans la main pour déplacer une valeur. Sur les pièces courtes (Fig. 5) l'assiette est symbolisée par un rond dessiné sur le côté droit des pièces. Sur le côté gauche, il y a un trou rectangulaire qui correspond au paramètre de l'instruction. Ce paramètre indique soit une entrée "?", soit une sortie "!", soit encore une des trois cases mémoire "X", "Y", "Z".

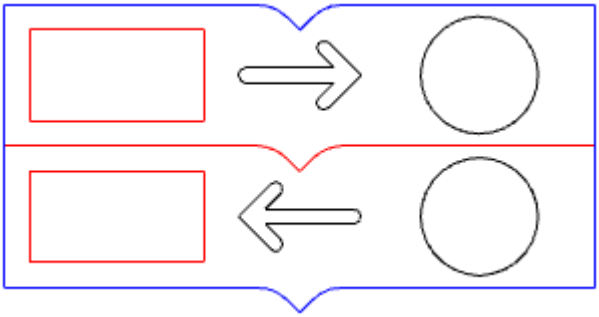


Fig. 4 - READ[ ] & WRITE[ ]

Chacune des pièces à une représentation textuelle qui est indiquée dans la légende des figures présentées dans ce chapitre tutoriel. Cette représentation textuelle sera utilisée pour écrire les solutions aux problèmes posés. En classe, nous recommandons avec de jeunes élèves de n'utiliser qu'une représentation graphique du code. Mais avec des élèves plus âgés, la représentation textuelle favorise la transition vers le langage Python.

Le début du programme est indiqué par un demi disque avec une flèche sur le bas et la fin du programme par un demi disque avec une encoche sur le haut (Fig. 6). Ces délimiteurs sont "obligatoires" pour habituer l'élève à la notion de bloc de code.

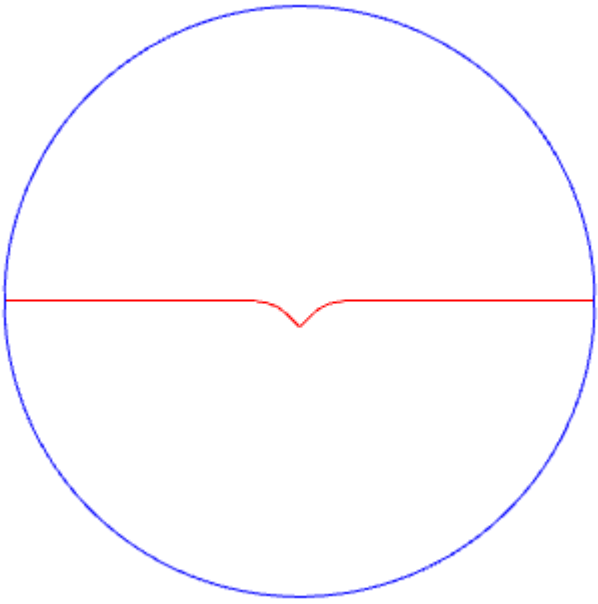


Fig. 5 - BEGIN & END

00000. Photocopieur

Recopie deux valeurs de l'entrée sur la sortie.

Ce premier exercice est très simple. Il permet de découvrir le dispositif et à faire le lien avec l'architecture de Von Neumann. Ce que l'élève écrit, ce sont les instructions de l'unité de contrôle. L'accumulateur de l'unité arithmétique et logique est représenté par le jeton transparent qui va se déplacer sur la liste des instructions en transportant les valeurs d'un lieu à l'autre de l'ordinateur. **L'accumulateur correspond à l'attention** chez l'humain : tous les deux sautent d'un endroit à l'autre et sont indivisibles c'est-à-dire qu'ils ne peuvent gérer qu'une seule chose à la fois. À des fins didactiques, il est possible d'écrire sur le jeton les valeurs transportées avec un feutre pour tableau blanc. Ce jeton permet de savoir à quelle étape de l'algorithme on se situe. Les entrées - sorties sont symbolisées par le point d'interrogation (on demande une valeur numérique, i.e. l'élève choisit un nombre entier) et le point d'exclamation (on écrit un résultat).

Une des difficultés identifiée auprès de certains élèves, est de comprendre que la flèche indique le sens dans lequel l'information se déplace. On ne peut pas écrire sur l'entrée et l'on ne peut pas lire sur la sortie. Il n'est donc pas inutile d'expliciter de quel côté de la flèche se situent la source et la destination des données copiées. Remarquez que **les données ne se déplacent pas : c'est l'information qui transite.** Ci-dessous un extrait réel d'un enregistrement d'une interaction de remédiation au sujet du sens de la flèche entre le prof (P) et l'élève (E) en difficulté :

Contexte :
Nous sommes en train de résoudre le problème “octoplicateur” qui multiplie un nombre par 8. Deux lignes de code sont supposées faire la même chose : la première écrit la valeur de l'accumulateur dans la mémoire X, la seconde dans la mémoire Y. Problème : les flèches sont en sens opposés. L'action de remédiation de l'enseignant est détaillée ci-dessous.

P> Regarde cette pièce là. Est-ce qu'elle est juste ou pas ?

E> Mmmm...

P> On a déjà écrit dans X deux lignes plus haut.

E> Mmmm... Je crois qu'elle est juste.

P> Donc écrire dans X et écrire dans Y sont identiques ? C'est juste X et Y qui changent entre ces deux lignes de code ?

E> Oui

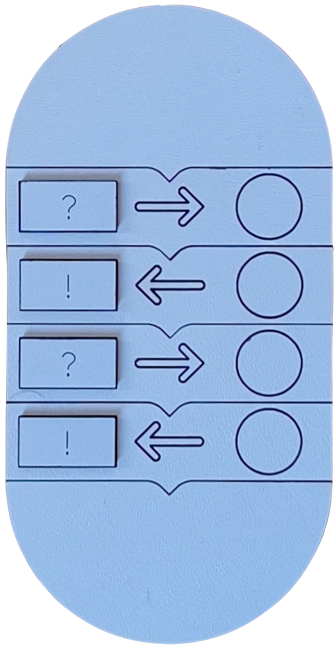
P> Il n'y a aucune autre différence à part X et Y ? Le sens de la flèche c'est le même ?

E> Oui

P> Tu vois bien que le sens de la flèche n'est pas identique. Alors il y a peut-être un problème. Parce que ça ça écrit dans X et ça ça lit ce qu'il y a dans Y. Le sens de la flèche indique le sens dans lequel va l'information. Si je te donne une valeur, elle va de chez moi vers chez toi (en faisant le geste avec la main). Ça va de ce que tu as dans l'assiette vers la X. Comme dans le jeu de la première semaine. Le jeu qu'on a fait et qui était très important pour comprendre le sens des flèches. Tu vois quand je te donne quelque chose, la flèche elle va dans ce sens. Si tu comprends bien le sens de la flèche c'est facile à écrire après.

solution

```
BEGIN
   READ [?]
   WRITE[!]
   READ [?]
   WRITE[!]
END
```



Le code de gauche est donné dans la syntaxe *hp!* que nous introduisons ici et à l'aide des figures ci-dessus, mais qui n'est pas présentée aux élèves dans leurs feuillets d'exercices. C'est donc à l'enseignant d'introduire cette syntaxe au tableau surtout si un passage sur Python est envisagé. Le code de droite est une photographie des pièces de la boite du jeu *hp!*

L'affectation fait partie des grosses difficultés du débutant en programmation avec les langages de programmation textuels traditionnels. Est-ce une égalité mathématique ? Dans quel sens est-ce que l'affectation fonctionne ? Le `READ` et le `WRITE` de *hp!* apportent une solution simple et élégante à ce problème, puisque l'on décompose l'affectation en deux temps : prendre une valeur sur l'assiette que l'on a dans la main et apporter la valeur que l'on a dans l'assiette vers une autre case mémoire ou la sortie.

00001. Inverseur

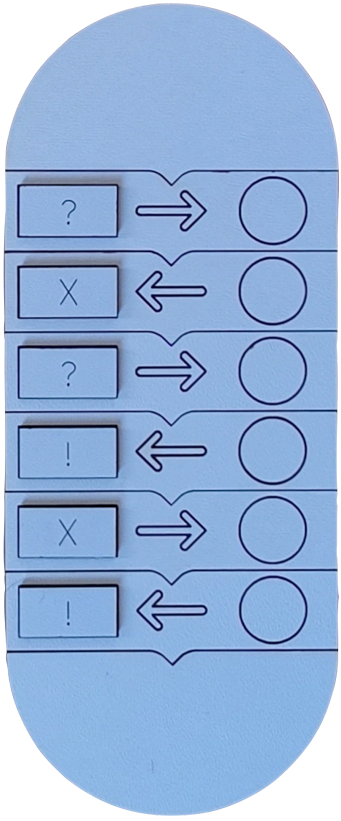
Recopie deux valeurs sur la sortie, mais en changeant l'ordre.

Pour faire cet exercice il faut utiliser une des trois cases mémoire. L'objectif de cet exercice est donc d'apprendre à utiliser la mémoire et de s'habituer, dors à déjà, à utiliser un bloc-note pour garder une trace de l'historique des valeurs en mémoire. Cette habitude nous permettra de déboguer le code lorsque les algorithmes deviendront plus complexes.

N.B. : on ne peut pas lire une mémoire avant d'y avoir inscrit quelque chose dedans (elles ne sont pas initialisées par défaut).

solution

```
BEGIN
    READ [?]
    WRITE[X]
    READ [?]
    WRITE[!]
    READ [X]
    WRITE[!]
END
```



00010. Additionneur

Lis deux entiers. Calcule leur somme.
Affiche le résultat sur la sortie.

Lorsque l'on recopie une valeur dans l'accumulateur (i.e. copier la valeur dans l'assiette), il est possible d'ajouter (ou de soustraire) la nouvelle valeur à la valeur existante, au lieu de la remplacer (Fig. 7). Les valeurs numériques "s'accumulent". Les deux exercices suivants travaillent avec l'addition de deux valeurs consécutives.

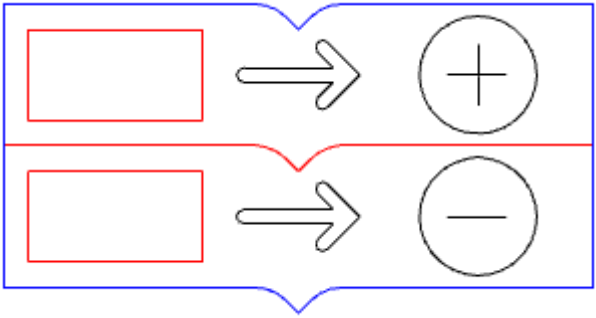
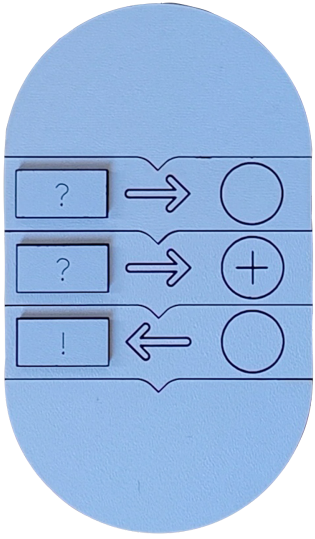


Fig. 6 - ADD[ ] & SUB[ ]

solution

```
BEGIN
    READ [?]
    ADD  [?]
    WRITE[!]
END
```



00011. Additionneur infini

Additionne toutes les valeurs reçues jusqu'à l'infini. Affiche la somme sur la sortie après chaque calcul.

Les pièces longues sont utilisées par paires (Fig. 8). Elles correspondent à un saut dans la séquence des instructions. Les motifs dessinés sur les extrémités des pièces sont uniques et sont en décalage par rapport au reste du code, pour permettre de les identifier plus rapidement. C'est un clin d'œil à l'indentation du code que les élèves apprendront avec Python.

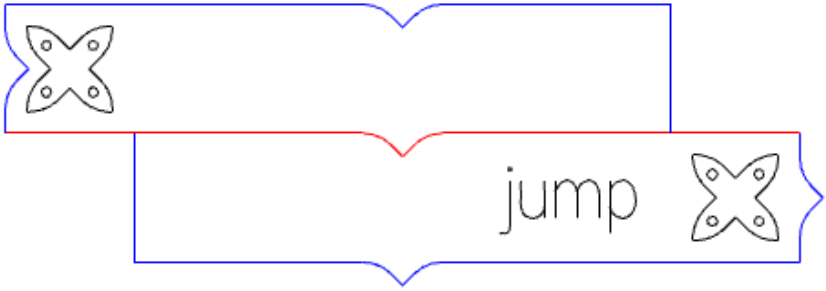
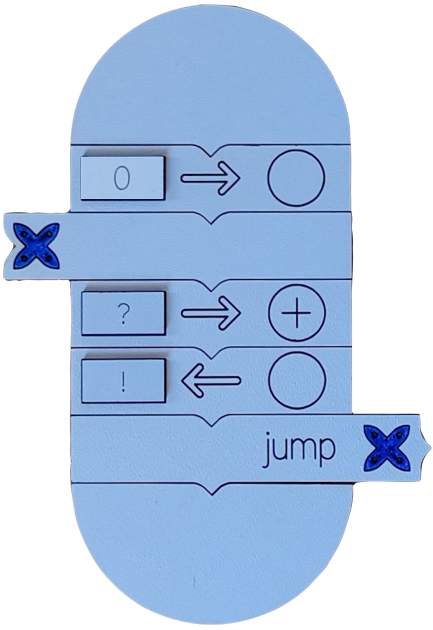


Fig. 7 - JUMP[ ]

Le dispositif ne contient que deux instructions `jump`. Si le code imaginé par l'élève nécessite plus que deux `jump`, c'est qu'il n'est pas optimal et qu'on doit pouvoir l'améliorer. Les `jump` (`goto`) ont été bannis des langages de programmation dans les années 1970 pour laisser place à la programmation structurée. Mais dans le code généré par les compilateurs de ces langages, ils existent toujours ! Ils correspondent donc bel et bien à ce qui se passe dans l'ordinateur et ont, de fait, une valeur didactique importante pour la compréhension des concepts. On invite cependant les élèves à ne pas en abuser.

solution

```
BEGIN
   READ [0]
[0]:
   ADD  [?]
   WRITE[!]
   JUMP [0]
END
```



00100. Comptable

Additionne toutes les valeurs positives et négatives. Affiche la somme sur la sortie après chaque calcul. Arrête toi dès que le total est nul.

La grosse différence entre une machine à calculer traditionnelle et un ordinateur, c'est qu'un ordinateur est capable de faire des choix en fonction de son état interne. Concrètement, ce choix se traduit par un saut qui ne s'effectue que si la condition est remplie (contrairement au `jump` qui saute systématiquement). Par exemple, si la valeur que nous avons dans la main (c-à-d dans l'accumulateur) est zéro, alors on saute, sinon on continue.

Pour bien comprendre le programme que les élèves doivent écrire, on peut leur proposer cette analogie. Il y a un sac de cerises. Chaque cerise est représentée par un nombre qui indique combien de jours il nous reste pour la manger. Si une cerise est pourrie, ce nombre est zéro : on la jette. Si une cerise est encore bonne, ce nombre est plus grand que zéro : on la mange.

La nouvelle instruction permettant de réaliser les sauts conditionnels, est illustrée en figure 9. En bleu est représenté le chemin si "ce que l'on a dans l'assiette" est égal à zéro. Dans ce cas, c'est le JUMP qui est exécuté et l'on continuera à l'instruction qui suit l'étiquette (le dessin) correspondante. En rouge est représenté le chemin si "ce que l'on a dans l'assiette" n'est pas égal à zéro. Dans ce cas, c'est l'instruction suivante qui est exécutée.

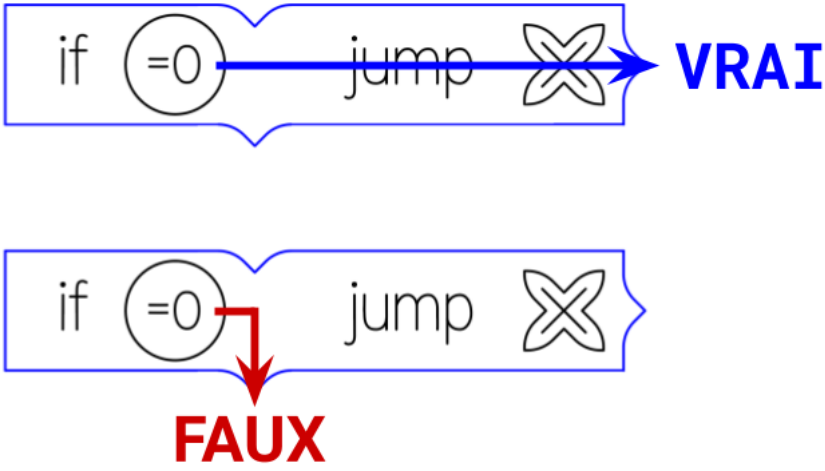
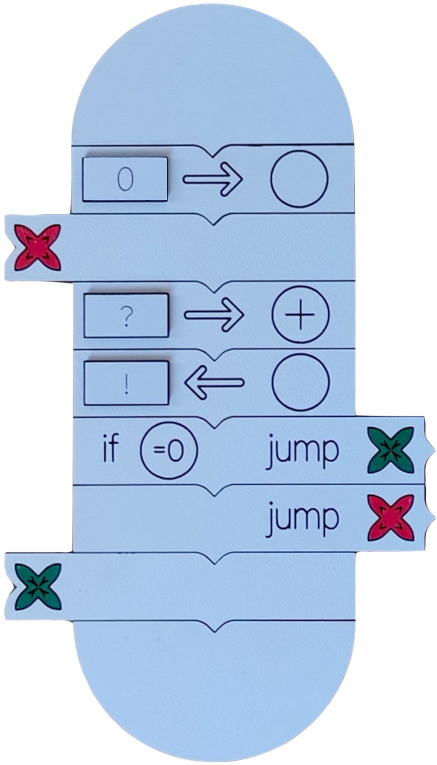


Fig. 8. IF [=0] JUMP[ ] : Le cas vrai et le cas faux.

solution

```
BEGIN
   READ [0]
[0]:
   ADD  [?]
   WRITE[!]
   if  [=0] JUMP [1]
   JUMP [0]
[1]:
END
```



03. Architecture du cerveau

Le cerveau, tout comme l'ordinateur, possède une architecture basée sur un organe dédié à la pensée (l'unité de contrôle), un organe spécialisé dans les opérations mentales (unité arithmétique et logique), un organe de mémorisation court terme (l'accumulateur) et long terme (la mémoire), des organes d'entrée (œil, oreille et tous les autres sens), des organes de sortie (mouvement, parole). Ce n'est pas un hasard. Les premiers ordinateurs étaient appelés cerveaux électroniques et les intentions de leurs créateurs[1] étaient clairement de créer une machine capable de reproduire l'activité intellectuelle de l'homme. Ainsi, ils se sont inspirés de ce qu'ils connaissaient le mieux pour définir l'architecture des machines que nous utilisons encore aujourd'hui.

Les deux figures (Fig. 9 et Fig. 10) suivantes présentent un modèle du fonctionnement du cerveau humain proposé par Pierre Vianin[2] suivi (page suivante) de l'interface du jeu *Human Resource Machine* qui a inspiré le dispositif *human processor!* Les similitudes sont frappantes. L'intelligence du cerveau humain est proportionnelle à la liste des stratégies cognitives maîtrisées. Une stratégie cognitive se définit comme la manière dont s'articulent l'ensemble des opérations mobilisant plusieurs processus cognitifs qu'il convient donc de développer chez l'élève.

---

[1] **Turing**, A. (1950). *Computing machinery and intelligence.* In Mind a quarterly review of psychology and philosophy. Vol. 59, N° 236. Manchester: Victoria University. **Von Neumann**, J. (1992). *L'ordinateur et le cerveau.* Paris : Édition de la découverte.

[2] **Vianin**, P. (2009). *L'aide stratégique aux élèves en difficulté scolaire – Comment donner à l'élève les clés de sa réussite ?* Bruxelles : De Boeck.

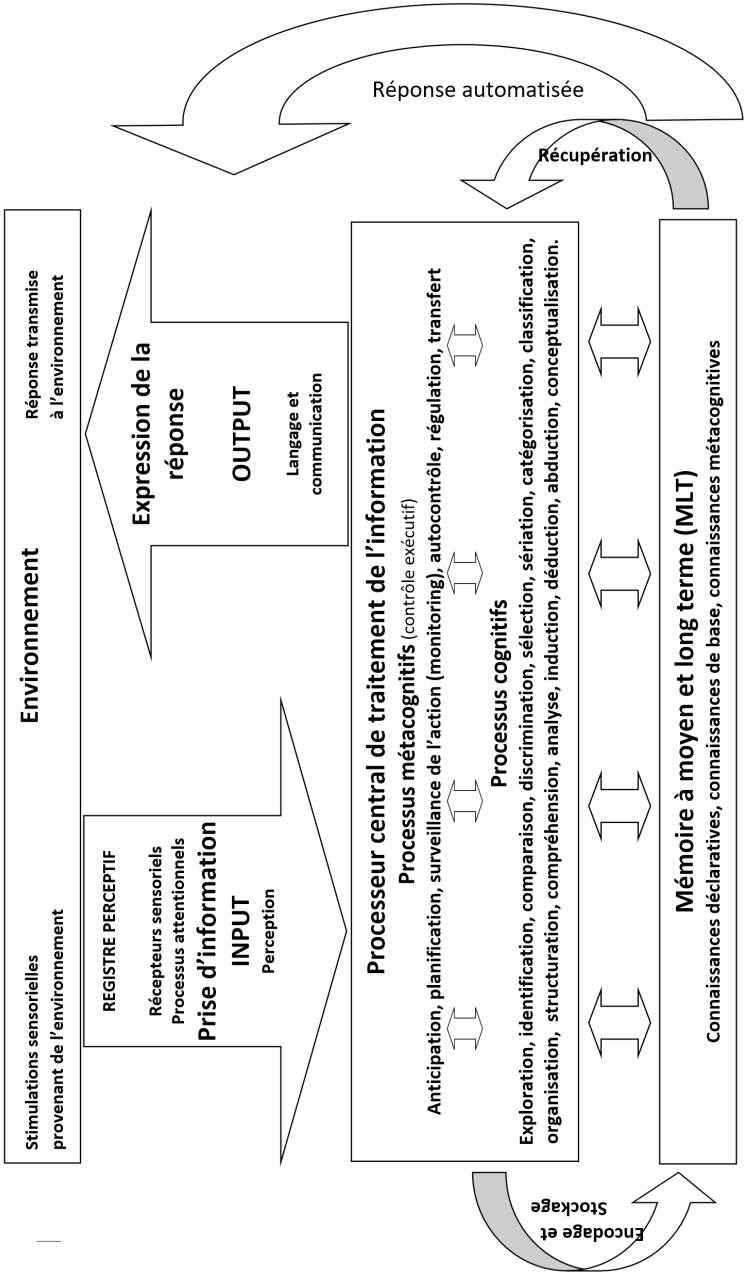


Fig. 9 - Modèle du fonctionnement du cerveau par P. Vianin

Fig. 10 - Interface du jeu *Human Resource Machine*

Les neurosciences[3] nous apprennent que l'algorithme utilisé par le cerveau pour apprendre comprend 4 étapes : l'engagement actif, l'attention soutenue, le retour sur erreur (l'erreur fait partie intégrante du processus d'apprentissage), la répétition (pour mémoriser puis automatiser les acquis). Pour apprendre à apprendre, il s'agit donc pour l'élève de s'efforcer à développer, chaque jour, chacune de ces aptitudes.

Pour cultiver les sentiments favorables à l'engagement actif auprès des élèves, vous pouvez stimuler leur motivation intrinsèque en excitant leur curiosité, en valorisant leur aptitude à expliquer à un autre élève ce qu'ils ont compris, en tissant des liens entre les activités faites en classe et ce qu'ils connaissent du monde. Joëlle Proust (pp. 180 - 185) nous dit que "chaque élève va se demander si l'activité est intéressante ou plaisante en elle-même, quel niveau d'effort elle demande (engagement), s'il ou si elle a des chances de réussir en s'appliquant (confiance en soi), si l'activité a des conséquences importantes pour lui ou pour elle (compatibilité avec ses objectifs, son image de soi, ses valeurs, ses stéréotypes)". Pour désactiver le stéréotype de genre, vous pouvez présenter aux élèves le documentaire "*The Computers*"[4] qui retrace l'histoire des premières programmeuses de l'histoire.

Pour l'attention, faire de la méditation en classe[5] peut beaucoup aider. Il faut être attentif pour comprendre. Il faut comprendre pour mémoriser. Ces trois aspects : l'attention, la compréhension et la mémorisation, sont développés dans le chapitre 9.

---

[3] **Dehaene**, S. (2019). *La science au service de l'école.* Paris : Canopé/Odile Jacob.
[4] **Kleiman**, K. (2016). *The Computers: The Remarkable Story of the ENIAC Programmers.* En ligne : <http://eniacprogrammers.org>
[5] https://www.jupiter-films.com/film-happy-la-meditation-a-l-ecole-100.php

04. Exercices optionnels

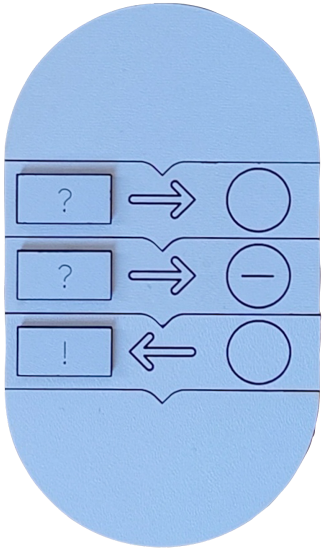
00101. Soustracteur

L'addition est commutative, c'est-à-dire qu'ajouter une valeur avec celle qui est dans l'assiette ou bien ajouter la valeur qui est dans l'assiette avec la valeur reçue, donne la même chose. **Comment fera-t-on avec la soustraction ?** Réponse : il faut définir une convention. La convention choisie est de soustraire la nouvelle valeur avec celle qui est déjà dans l'assiette.

Lis deux entiers. Calcule leur différence.
Affiche le résultat sur la sortie.

solution

```
BEGIN
   READ [?]
   SUB  [?]
   WRITE[!]
END
```

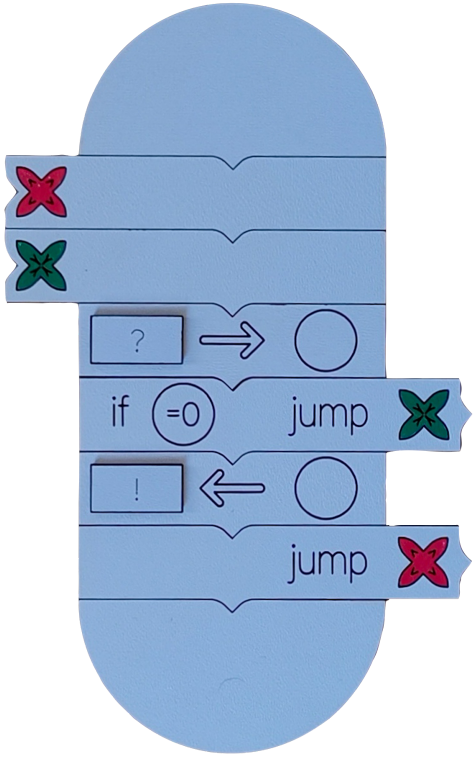


00110. Exterminateur de zéro infini

Recopie tous les entiers non nuls sur la sortie.

solution

```
BEGIN
[1]:
[0]:
   READ [?]
   if  [=0] JUMP [0]
   WRITE[!]
   JUMP [1]
END
```

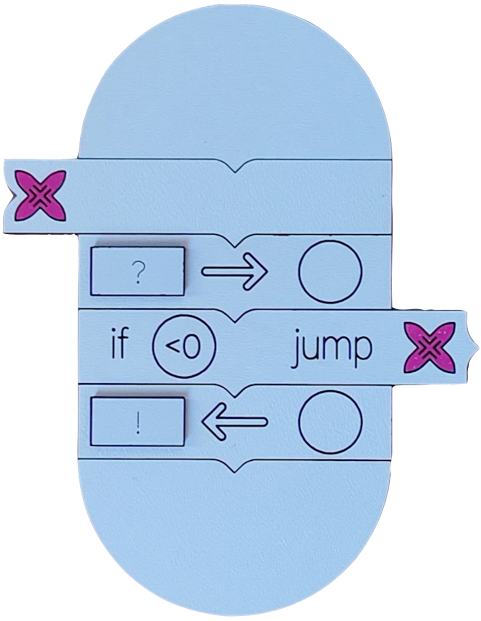


00111. Exterminateur de négatifs

Ne recopie un entier sur la sortie que s'il est positif.

solution

```
BEGIN
[0]:
   READ [?]
   if  [<0] JUMP [0]
   WRITE[!]
END
```

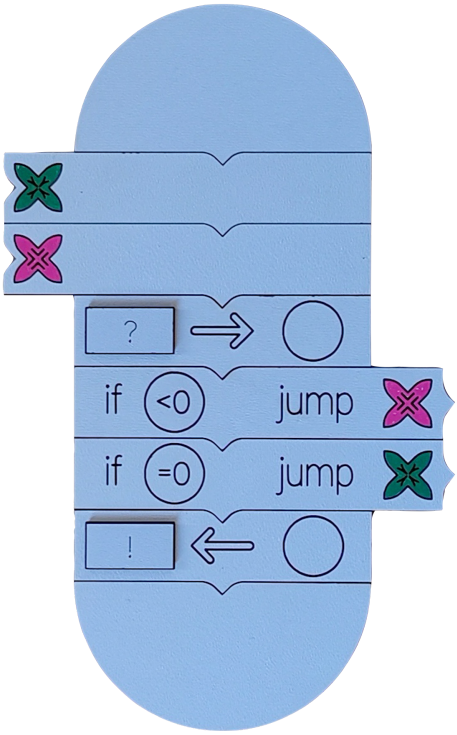


01000. Exterminateur de négatifs ou nuls

Ne recopie un entier sur la sortie que s'il est strictement positif (positif et non nul).

solution

```
BEGIN
[0]:
[1]:
   READ [?]
   if  [<0] JUMP [1]
   if  [=0] JUMP [0]
   WRITE[!]
END
```

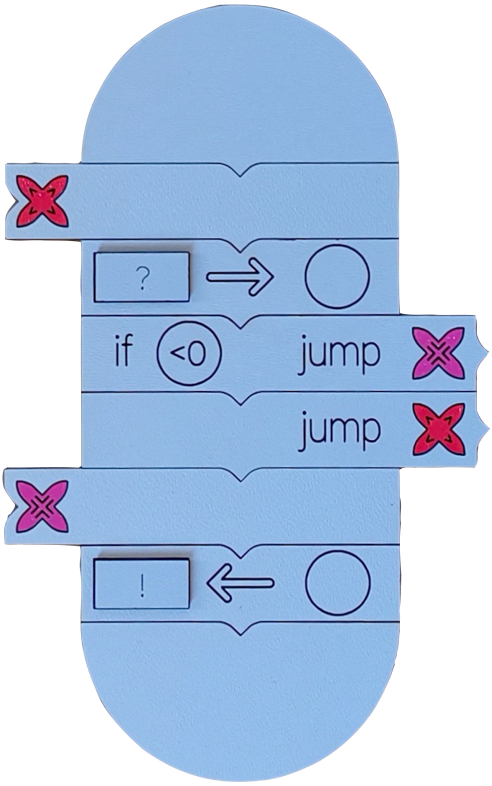


01010. Exterminateur de positifs

Ne recopie un entier sur la sortie que s'il est négatif.

solution

BEGIN
[1]:
  READ [?]
  if [<0] JUMP [0]
  JUMP [1]
[0]:
  WRITE[!]
END

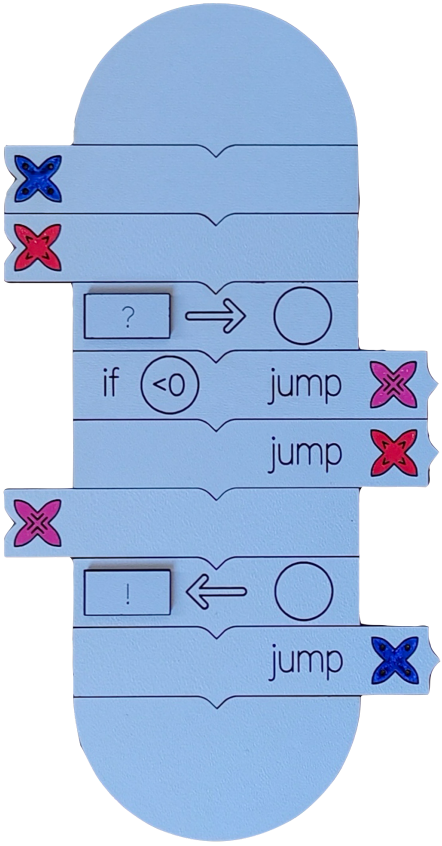


01011. Exterminateur de positifs infini

Recopie tous les entiers négatifs sur la sortie.

solution

```
BEGIN
[2]:
[1]:
   READ [?]
   if  [<0] JUMP [0]
   JUMP [1]
[0]:
   WRITE[!]
   JUMP [2]
END
```



05. Détecter les élèves en difficulté

Certains élèves n'arrivent pas à développer les compétences nécessaires pour penser les algorithmes. Qu'est-ce qui dans leur attitude en classe nous permettrait de les identifier de manière précoce ? Nous avons découvert que, quel que soit leur âge, les élèves présentant des difficultés d'apprentissage de l'algorithmique et qui ne réussissent pas à apprendre, avaient un comportement similaire dès les premières leçons.

Si l'enseignant en science informatique souhaite étendre la portée de son action formative, il doit chercher à comprendre l'origine des difficultés d'apprentissage des élèves pour y remédier. Problème de motivation ? De confiance en soi ? De trouble d'apprentissage ? La question n'est pas toujours simple et pour certains élèves, elle est même plutôt complexe. Lorsqu'on propose à l'élève de résoudre un problème algorithmique, on va directement solliciter ses compétences cognitives nécessaires pour penser les algorithmes informatiques qui mobilisent essentiellement les intelligences pratique, créative et analytique. Détecter précocement les difficultés d'apprentissage des élèves permettra à l'enseignant d'effectuer des remédiations dans ces trois dimensions.

En comparant l'activité en classe d'élèves en difficulté d'apprentissage qui réussissent à apprendre grâce à l'aide de l'enseignant et du temps, avec ceux qui n'y arrivent pas, nous

avons observé que les élèves en difficultés qui n'arrivent pas à apprendre n'entrent quasiment jamais dans une phase de réflexion ou de compréhension de ce qu'ils font ou de ce qu'il faut faire, et donc ne développent pas de modèle mental leur permettant de cultiver leur pensée algorithmique. Ils restent intellectuellement passifs durant toute l'activité et n'apprennent donc pas grand chose, car en informatique il ne suffit pas d'observer pour apprendre à coder. Il faut coder soit même pour s'entraîner à penser le code de manière à réussir à se construire un modèle mental de la machine notionnelle correspondant au dispositif didactique utilisé, c'est-à-dire une représentation mentale efficiente et cohérente pour pouvoir développer sa pensée algorithmique.

S'ils programmaient derrière un écran au lieu de programmer sur un dispositif didactique débranché, ce serait encore moins détectable puisqu'ils seraient probablement physiquement actifs à manipuler l'interface tout en restant intellectuellement passifs et donc sans apprendre. Le dispositif que nous utilisons à l'avantage de rendre visible les réflexions des élèves et donc de permettre de détecter très tôt ces élèves pour pouvoir essayer de les aider à apprendre à penser les algorithmes.

Nous avons observé qu'il n'était pas bon signe qu'un élève reste trop longtemps dans l'action de codage ou dans un semblant de réflexion. Il est certainement en train de rêvasser ou de penser à autre chose tout en jouant avec les pièces. S'il était en train de raisonner sur le problème en programmant, il faudrait en effet qu'il arrête de manipuler les pièces à un moment donné, pour réfléchir à nouveau avant de se remettre à tester ses idées. Ainsi, ce qui est important d'observer pour l'enseignant, c'est la fréquence de l'alternance des états de l'élève. Un élève qui est en

train de réfléchir, teste ses idées fréquemment et communique sur celles-ci lorsqu'il trouve quelque chose, ou bien va demander de l'aide pour comprendre si ce qu'il a fait fonctionne ou pas (et sera alors parfaitement attentif au moment où il recevra ces explications). Un élève qui manipule ses pièces pour coder de manière continue durant une trop longue période a juste besoin d'aide.

Les élèves qui présentent des difficultés d'apprentissage et qui ne réussissent pas à apprendre ont, dès les tout premiers exercices du début de la formation, une attitude en classe qui ne favorise pas le développement de leurs compétences à penser les algorithmes. Une chose remarquable est que, quel que soit leur âge, leurs comportements sont similaires : ils n'entrent jamais dans des états mobilisant leur intelligence algorithmique. Si les élèves travaillent sur écran, la seule manière pour l'enseignant de voir une différence dans le comportement des élèves en difficulté est d'être attentif aux moments où l'élève va spontanément communiquer sur ce qu'il a réussi à faire. Une bonne pratique consisterait alors à favoriser les échanges en faisant des interruptions fréquentes au cours de l'activité de résolution pour leur demander individuellement d'expliquer le problème, ce qu'ils sont en train de faire, mais aussi d'expliquer ce qu'ils ont déjà réussi à faire. L'enseignant pourra alors détecter assez tôt les élèves qui ne communiquent pas sur leurs travaux et qui n'arrivent donc pas bien à apprendre.

Pour aider ces élèves, l'enseignant devra identifier les fonctions cognitives défaillantes mobilisées dans l'acte de trouver des solutions algorithmiques aux problèmes proposés (voir *chap.11 - Apprendre à résoudre*, *chap.24 - Penser les algorithmes* et la figure 11 ci-dessous). C'est une remédiation délicate qui requiert du

temps et de l'expertise. S'il n'est pas toujours possible de travailler sur ces fonctions cognitives avec un élève en particulier au sein du groupe classe, il peut être envisagé d'accorder un temps de soutien spécifique en dehors de la période de cours, pour renforcer les aptitudes cognitives des élèves présentant des troubles d'apprentissage de l'algorithmique et expliciter les processus et les stratégies cognitives qui aident à résoudre un problème algorithmique, ce qui leur permettra de développer leur intelligence algorithmique pour réussir à penser les algorithmes.

Source : https://bit.ly/PosterDidapro9

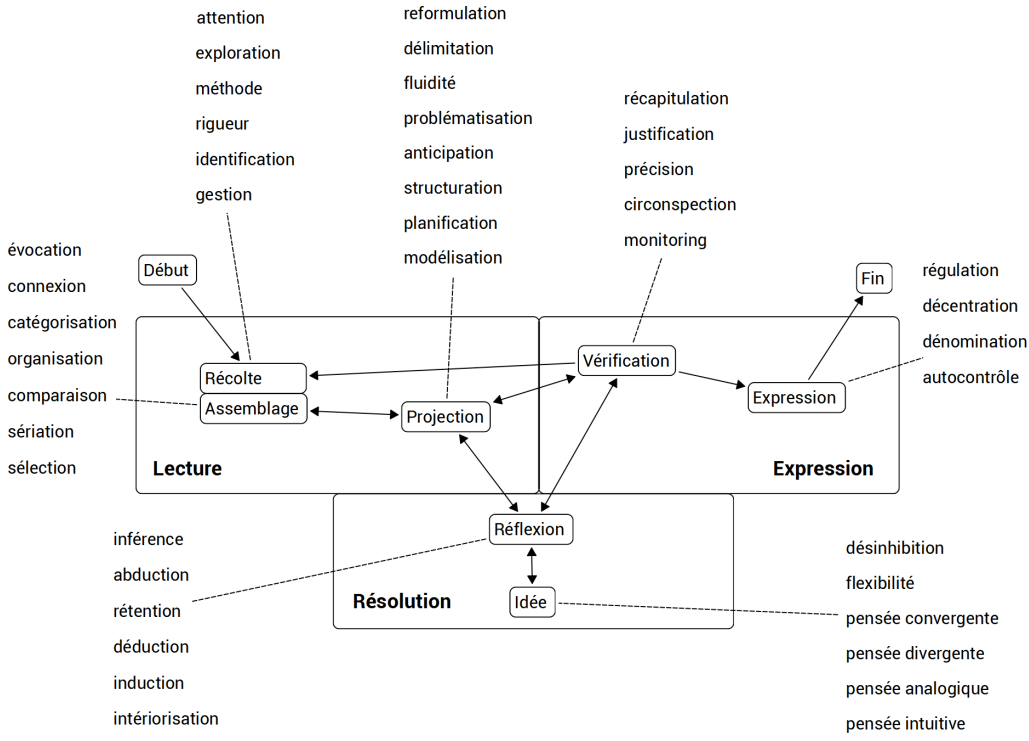


Fig. 11 - Les trois phases de résolution d'un problème algorithmique

06. Conservateur de zéro.

Commencer par une révision des concepts vus durant le tutoriel, en faisant émerger du groupe classe les notions mémorisées et en les complétant si nécessaire. Puis faire réfléchir les élèves sur les questions suivantes :

A votre avis, comment sont représentées les lettres dans la mémoire d'un ordinateur ? Réponse : avec des nombres. Par exemple le A correspond au nombre 1, le B au 2.

Mais alors, comment l'ordinateur fait pour distinguer les nombres des lettres ? Réponse : il ajoute un deuxième nombre qui indique si c'est un nombre ou une lettre ou autre chose.

A votre avis, comment sont représentés les nombres dans un ordinateur ? Réponse : en binaire, c'est à dire uniquement avec des 0 et des 1 regroupés par paquets de 8 : des octets.

A votre avis, comment est représenté le code en mémoire ? Réponse : toujours avec des nombres. Par exemple, le JUMP correspond au nombre 1, etc.

A votre avis, est-ce que nous, humains, on code et décode les informations ? Réponse : oui ! On travaille avec des concepts, pas avec des sons et des images brutes, mais avec l'idée de ce qu'on a vu ou entendu.

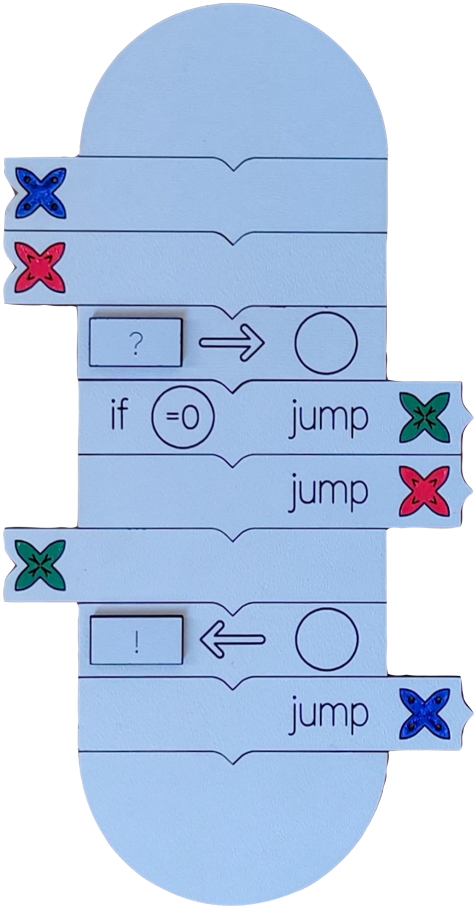
01011. Conservateur de zéro.

Recopie uniquement les zéros sur la sortie.
Challenge : 5 instructions.

Cet exercice reprend, à titre de révision, la difficulté de l'exercice *01010. Exterminateur de positifs infinis.* Nous n'avons pas l'opérateur "différent de". Il s'agit donc de bien comprendre le principe du saut conditionnel.

une solution possible

```
BEGIN
[0]:
[1]:
   READ [?]
   IF  [=0] JUMP [2]
   JUMP [1]
[2]:
   WRITE[!]
   JUMP [0]
END
```



100% des personnes qui se sont contenté de comprendre le codage ont échoué leurs examens d'informatique. **Il est indispensable de comprendre mais ce n'est pas suffisant.** C'est dans l'effort mental fait pour concevoir du code que se forge la pensée algorithmique. Plus vous passez de temps à penser le code, plus vite vous apprendrez à programmer. Les premiers exercices de ce cours sont une construction : il faut en avoir acquis les concepts pour pouvoir réaliser les exercices suivants. N'hésitez donc pas à refaire les premiers exercices parfaitement avant de continuer.

07. Histoire des ordinateurs

De Turing, à Zuse et à l'ENIAC.

Les premiers ordinateurs étaient mécaniques : des rouages et des engrenages permettaient à la machine de réaliser des calculs en interprétant un programme. C'est la comtesse Ada Byron de Lovelace qui conçut le premier véritable programme informatique lors de son travail sur une de ces machines dans les années 1840, une centaine d'années avant que le premier ordinateur, construit en 1940 par Alan Turing, ne fût opérationnel. A la même époque, Konrad Zuse mettait au point le Z3 qui, contrairement au Z1 (un ordinateur mécanique qui n'a jamais bien fonctionné) était électronique. Le Z3, achevé en 1941 était programmable et travaillait en binaire, mais ne pouvait pas faire de saut conditionnel. C'est à dire qu'une fois un programme chargé en mémoire, il exécutait toujours la même séquence de calculs. C'est le Z4, construit en 1945 qui fût le premier ordinateur au monde vendu[1] à l'ETHZ en 1950. La version commercialisée était équipée d'un lecteur de cartes perforées. Sa vitesse était de 40 Hertz. Il mettait 400 millisecondes pour faire une addition. Sa mémoire vive était mécanique et comportait 64 mots de 32 bits. On était bien loin des capacités (et de la taille) de nos smartphones ! Mais à cette époque, il était encore possible de comprendre de A à Z comment fonctionnaient ces machines.

[1] http://www.ethistory.ethz.ch/rueckblicke/departemente/dinfk/weitere_seiten/angewandte_mathematik/index_DE/popupfriendly/

Les premières programmeuses[2] au monde avaient même réussi à inventer la programmation à partir des plans de l'ENIAC !
Si vous ne l'avez pas déjà fait en début de formation, nous vous conseillons vivement de regarder avec toute la classe le reportage sur leur travail extraordinaire (durée 20 minutes) : vous pouvez faire naître quelques vocations chez les filles.

C'est cette relative accessibilité qui a inspiré le dispositif didactique que vous utilisez et en particulier son architecture, inspirée de l'EDVAC, conçue en 1945 par John von Neumann et dont la construction s'est achevée en 1949. Pourquoi ce retour aux sources ? Pour que vous puissiez reconstruire mentalement le chemin que les pionniers de l'informatique ont fait. Ils ont en effet poussé les limites du langage rudimentaire que hp! utilise, en créant des langages plus concis et plus puissants, mais également plus complexes à apprendre. Le Dr. Stuart Madnick à inventé en 1965 un ordinateur à vocation éducative, le Little Man Computer, qui pouvait être programmé avec un code machine simplifié. C'est ce langage qu'utilise *hp!* et nous l'avons encore plus simplifié. Vous apprenez-donc un langage proche de la machine, basé sur l'architecture des ordinateurs d'aujourd'hui, qui est simple et qui vous permet de comprendre entièrement la manière dont les données sont déplacées dans le processeur pour réaliser les calculs demandés. Notre objectif est que vous réussissiez à vous construire **un modèle mental de la machine dans lequel vous pourrez opérer comme le fait un processeur**, en déplaçant les données pour résoudre, dans votre tête, les problèmes proposés. Notre hypothèse est que la simplicité de la machine et du langage utilisé va permettre de faire naître en vous une pensée algorithmique et favoriser votre apprentissage de la programmation.

[2] http://eniacprogrammers.org/

08. Doubleur. Triplicateur.

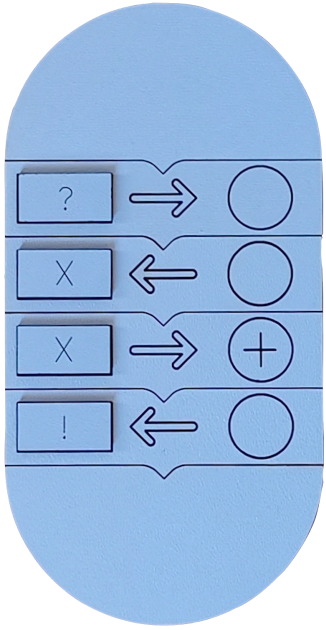
01100. Doubleur

Multiplie par deux la valeur entrée.
Affiche le résultat sur la sortie.

Indice : X + X = 2.X

solution

```
BEGIN
   READ [?]
   WRITE[X]
   ADD  [X]
   WRITE[!]
END
```



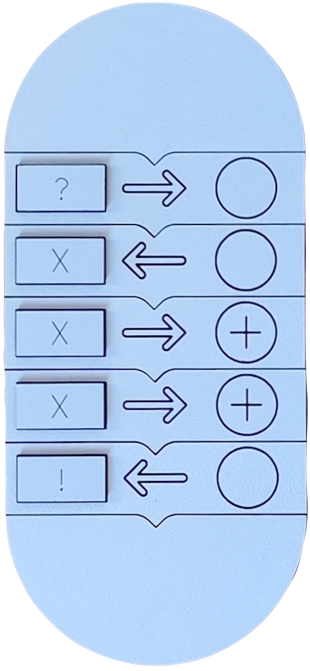
01101. Triplicateur.

Multiplie par trois la valeur entrée.
Affiche le résultat sur la sortie.

Indice : rappeler aux élèves que multiplier par deux c'est ajouter autant que ce qu'il y a déjà.

solution

```
BEGIN
   READ [?]
   WRITE[X]
   ADD  [X]
   ADD  [X]
   WRITE[!]
END
```



09. Apprendre à apprendre

La principale différence entre un élève qui réussit à l'école et un élève qui a des difficultés, tient dans son aptitude à saisir le sens des concepts de haut niveau à partir des expériences et informations vues en classe, chose que les autres ne perçoivent pas. Apprendre à apprendre consisterait donc[1] à apprendre à donner du sens à ce que l'on perçoit en classe. Or si la motivation intrinsèque de l'élève en difficulté détourne son attention de l'action qui se passe en classe, il n'y a aucun espoir qu'il puisse réussir à construire du sens avec ce qui se passe ici et maintenant. Si la clé de la compréhension est l'attention, la clé de l'attention est la motivation Si la clé de la motivation est l'estime de soi, la clé de l'estime de soi est l'affect. Dans cette construction pédagogique holistique, votre rôle d'enseignant est complexe. Nous n'avons hélas pas de recette magique. Vous pouvez varier l'approche "branché / débranché" et tenter d'introduire le jeu sur tablette pour motiver les élèves qui auraient des difficultés à trouver du sens dans les activités proposées. Mais très vite, dès que la simplicité inhérente aux premiers exercices s'estompera (vers l'exercice "Initiative de conservation de zéro"), les élèves se retrouveront seuls, sans l'aide des stratégies de résolution prodiguées par le groupe ou l'enseignant et peut-être sans avoir vraiment bien compris les bases.
Vous aurez certainement besoin de répondre à la question : "à quoi ça sert d'apprendre à programmer ?". La réponse que nous

[1] **Bransford**, J. D. **Brown**, A. L. **Cocking**, R. R. (2000). *How people Learn. Brain, Mind, Experience, and School.* Washington: National Academy Press.

vous proposons de donner est assez claire : probablement *à rien*. En tout cas, pas plus que l'ensemble des autres matières que l'on étudie à l'école, dès lors que l'on sait lire et compter. Cependant, dans un monde où l'on vit de manière quotidienne en interaction avec des algorithmes, il ne nous semble pas complètement inutile d'apprendre leur logique et d'en comprendre les fondements. Mais c'est davantage la pensée algorithmique qui est intéressante en soi que le codage. C'est-à-dire que le plus important dans cette formation est d'apprendre la manière dont il faut raisonner lorsque l'on est face à un ordinateur ou un robot.

La vraie finalité de cet enseignement se situe au niveau des apprentissages métacognitifs et de la découverte des stratégies cognitives. Apprendre à résoudre des problèmes est une compétence transversale qui elle sert tout au long de la vie. Dans cette formation, nous fournissons non seulement les clés pour apprendre à résoudre des problèmes mais nous avons aussi l'ambition d'aider les élèves à développer leur intelligence, algorithmique, c'est-à-dire leurs intelligences logico-mathématique, pratique et créatrice.

Cette leçon se concentre sur trois aspects de l'apprendre à apprendre : être attentif, comprendre ce qui est attendu des exercices proposés et mieux mémoriser les concepts présentés. Nous en profiterons pour faire des parallèles entre la mémoire humaine et la mémoire des ordinateurs, afin d'introduire quelques concepts issus des neurosciences de l'éducation.

Nous terminerons par une check-list métacognitive que vous pouvez utiliser pour déjà apporter de l'aide à ceux qui en ont besoin, lors des différentes étapes de résolution d'un problème, qui sont :

- avant : Que faire pour pouvoir se mettre au travail ?
- pendant : Comment réfléchir, comment se débloquer ?
- après : Qu'est-ce que j'ai appris et que dois-je retenir ?

Mieux mémoriser.

L'être humain, tout comme l'ordinateur, possède plusieurs types de mémorisations qui travaillent sur des durées plus ou moins longues. Le disque dur de l'ordinateur correspond à notre mémoire long terme. La mémoire vive de l'ordinateur correspond à notre mémoire de court terme. Les registres correspondent à la mémoire sensorielle (ce que l'on vient juste d'entendre ou de voir, que l'on peut se repasser comme un enregistrement audio ou un film). Mais le corps a aussi sa mémoire (implicite), la partie du cerveau qui s'occupe de la musique est également distincte du reste. Ainsi, une manière de mieux mémoriser les choses est d'associer différents systèmes de mémoire : par exemple apprendre à l'aide d'une mélodie, ou apprendre en associant des gestes. Mais dans tous les cas, le meilleur moyen pour mémoriser quelque chose, c'est de faire l'exercice de s'en rappeler. Le plus souvent on retrouve une chose en mémoire et plus ça devient facile d'y accéder. Donc, pour mémoriser des savoirs scolaires, le mieux c'est de s'entraîner à se souvenir de ses savoirs. Tous les moyens sont bons et il existe plein de techniques[2].

[2] **Eustache**, F. & **Guillery-Girard**, B. (2016). *La neuroéducation. La mémoire au coeur des apprentissages.* Paris : Odile Jacob. Chapitre 4.

La courbe de l'oubli nous apprend que si on fait cet exercice de rappel (1) chaque soir pour les savoirs appris durant la journée, (2) chaque soir pour les savoirs appris la veille, (3) chaque lundi pour les savoir appris vendredi, jeudi pour les savoirs appris lundi, vendredi pour les savoirs appris mardi, samedi pour les savoirs appris mercredi, dimanche pour les savoirs appris jeudi, (4) chaque jour pour les savoirs appris le même jour la semaine dernière, (5) chaque jour pour les savoirs appris il y a deux semaines, cela fait 5 x 10 mn = 50 minutes tous les soirs avant de s'endormir, alors on n'oublie plus rien.

Pour mémoriser il y a trois étapes : (1) encoder ce que l'on a entendu et compris, (2) stocker l'information en réalisant des associations d'idées, en mémorisant le chemin qui a conduit à l'information ou le contexte, et (3) retrouver l'information en faisant l'effort de se souvenir (aussi fréquemment que nécessaire pour s'en rappeler vraiment).

Faire travailler notre mémoire permet d'alléger notre charge cognitive[3], ce qui nous permet d'automatiser une part du travail à faire et libère notre cerveau que l'on va pouvoir utiliser pour faire autre chose et en particulier résoudre le problème posé ou bien faire plusieurs autres choses en même temps. Une écoute active permet de faire des liens avec ce que l'on connaît déjà pour mieux le mémoriser, ou bien permet de tenter de deviner l'intention de l'interlocuteur, ce qui aide aussi à retrouver le chemin par le biais du fil de la pensée…

[3] **Chanquoy**, L. **Tricot**, A. **Sweller**, J. (2007). *La charge cognitive. Théorie et application.* Paris : Armand Colin.

Être attentif.

La méditation, pratiquée en classe[4], aide à la concentration des élèves. L'usage d'un tétra-aide[5] facilite la gestion de classe.

Mais pourquoi faut-il faire attention à son attention ? Parce qu'un des mécanismes de notre cerveau est de faire le nettoyage de ce qui ne lui semble pas très utile et d'oublier. En focalisant notre attention, on fait passer l'information de la mémoire sensorielle à la mémoire court-terme et l'on dit explicitement à notre cerveau que ce sur quoi on est concentré est important. Et le bon moyen pour s'en souvenir est de s'en souvenir ! C'est-à-dire se poser explicitement la question qu'est-ce que j'ai appris ou qu'est-ce que je dois me rappeler et est-ce que je m'en souviens ? C'est le bon moyen pour faire passer l'information de la mémoire court terme à la mémoire long terme.

Qu'est-ce qui détourne notre attention ? Un smartphone susceptible de vibrer dans notre poche, une nuit trop courte car passée à jouer en ligne au lieu de se reposer, le bruit et les conversations ambiantes : voilà trois facteurs sur lesquels il est assez simple d'agir pour augmenter son attention.

Il y a une grande différence entre entendre et écouter, voir et regarder. Pour être attentif, il faut être centré sur soi, ici et maintenant. Si une pensée parasite vient détourner notre attention, ne pas stresser (c'est normal, c'est le fonctionnement de notre cerveau qui est comme ça) : il faut juste ne pas la suivre, la laisser passer, et se focaliser sur ce qui se passe ici et maintenant.

---

[4] Happy, la Méditation à l'école : https://www.jupiter-films.com/film-happy-la-meditation-a-l-ecole-100.php
[5] http://bdemauge.free.fr/tetraaide2.pdf

Il ne suffit donc pas d'être en silence pour apprendre, mais il faut chercher à décoder ce qui est dit, mémoriser ce qui est vu, créer des liens avec ce que l'on connaît déjà, se demander où en veut venir le prof ou quelles sont les autres situations ou ce que l'on m'explique pourrait s'appliquer. C'est cette activité mentale qui fait la différence entre un élève attentif qui réussit ses études et un élève distrait qui doit, ensuite, travailler beaucoup plus pour arriver au même résultat que l'élève attentif. Finalement, c'est donc un objectif long terme : réussir à comprendre et mémoriser sans travailler qui doit permettre au cerveau d'accepter d'aller à l'encontre d'un objectif court terme : se faire plaisir en pensant à autre chose que ce qui se passe d'important en classe.

Comprendre ce qui est attendu

Décrypter les attentes du prof, c'est faire des liens avec ce que l'on sait déjà et tenter d'anticiper les questions qui pourraient être posées lors de l'examen. C'est aussi construire un lien émotionnel et affectif avec ce qui se passe. Le circuit de la récompense s'active lorsque l'on découvre, avant que ce soit dit, là où le prof souhaite aller. Puis lorsque le but reste flou, on active le lien social : on demande aux autres ce qu'ils ont compris, ou bien on va voir le prof à la fin du cours pour essayer de mieux comprendre son propos. En essayant de comprendre les intentions du prof, on cherche à se projeter plus loin dans l'avenir, ce qui met le présent en perspective et nous donne un projet.

Check list métacognitive :

Avant : *Que faire pour pouvoir me mettre au travail ?*

- Qu'est-ce que je vois ?
- Qu'est-ce que je dois faire ?
- Qu'est-ce qui est utile et qu'est-ce qui ne l'est pas ?
- Est-ce que j'arrive à faire des liens avec des choses que je connais ou que l'on a déjà vu en classe ?

Pendant : *Comment réfléchir, comment me débloquer ?*

- Si je fais ça, que se passera-t-il ?
- Quelles sont les prochaines étapes ?
- Est-ce que je prends bien le temps de vérifier ma réponse avant de répondre ?
- Est-ce que j'ai vraiment exploité toutes les données et exploré toutes les possibilités ?

Après : *Que vaut ma solution ? Qu'est-ce que j'ai compris ?*

- Mon résultat est-il précis ? Les autres ont-ils compris ?
- Qu'est-ce que j'ai appris et que dois-je retenir ?
- Un mot qui rassemble plusieurs choses ?
- Une situation semblable ou je pourrais réutiliser ce que j'ai compris ?

Img. 1 - Image libre de droits. Source : https://www.hiclipart.com/free-transparent-background-png-clipart-yttwp

10. Quadriplicateur. Octoplicateur.

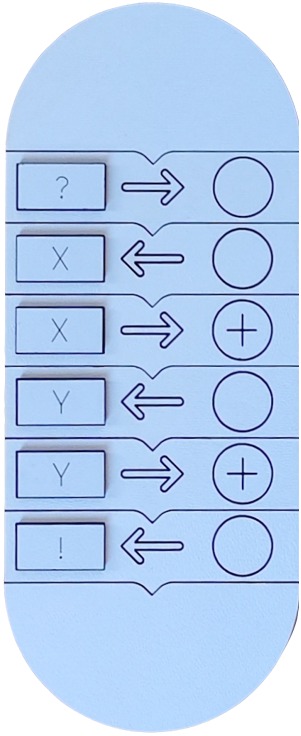
01110. Quadriplicateur.

Multiplie par quatre la valeur entrée.
Affiche le résultat sur la sortie.
Challenge : maximum 2 additions.

Indice : 1+1=2 et 2+2=4

solution

```
BEGIN
   READ [?]
   WRITE[X]
   ADD  [X]
   WRITE[Y]
   ADD  [Y]
   WRITE[!]
END
```



Explicitation du processus de réflexion

- Lire l'énoncé.
- Faire le lien avec l'exercice *triplicateur* précédent.
- Ajouter une addition.
- Lire le challenge de n'utiliser que deux additions.
- Penser que (1+1+1+1)=((1+1)+(1+1)) et que de mémoriser la somme intermédiaire pour l'additionner à elle-même, permet de multiplier par quatre avec une addition de moins.

#FAIRE_DES_LIENS
#RAISONNEMENT_LOGIQUE

01111. Octoplicateur.

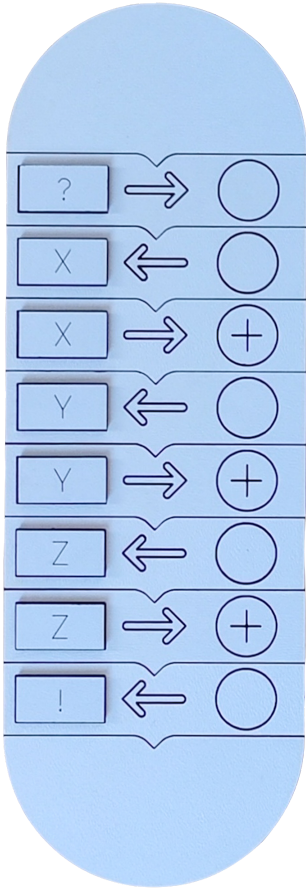
Multiplie par huit la valeur entrée.
Affiche le résultat sur la sortie.
Challenge : maximum 3 additions.

Déjà fini ? Essaye maintenant de résoudre ce problème en respectant le challenge proposé.

Indice : 8 est une puissance de 2. Pour obtenir 8 à partir de 1, ajouter 1 à lui-même, puis ajouter le résultat à lui-même, et recommencer encore une fois. En effet : 1+1=2, 2+2=4, 4+4=8.

une solution possible

```
BEGIN
   READ [?]
   WRITE[X]
   ADD  [X]
   WRITE[Y]
   ADD  [Y]
   WRITE[Z]
   ADD  [Z]
   WRITE[!]
END
```



Explicitation du processus de réflexion

- Lire l'énoncé.
- Faire le lien avec l'exercice *quadriplicateur* précédent.
- Raisonner de la même manière : utiliser de la mémoire pour éviter des additions.
- Écouter l'indice de l'enseignant si bloqué.
- Stocker le résultat fois 4 dans une mémoire et l'ajouter à elle même pour obtenir fois 8.

#FAIRE_DES_LIENS
#RAISONNEMENT_LOGIQUE

11. Apprendre à résoudre

Schéma global des processus d'apprentissage et de résolution

La figure 12 présente les opérations mentales réalisées en étapes successives lors des actions d'apprentissage et de résolution. Pour résoudre, il faut d'abord apprendre quel est le problème à résoudre, et donc récolter des données, faire des liens, de l'assemblage, avec des choses connues, définir un plan d'action de la pensée, qui va mener à une réflexion ou permettre l'élaboration d'une idée, que l'on va ensuite vérifier et, si l'idée ou la réflexion ne mène pas au résultat souhaité, on recommence. Sinon, si l'idée semble bonne, on l'exprime et plus tard on analyse ce qui nous a fait trouver la réponse pour apprendre.

Les flèches pleines représentent les transitions les plus probables entre les phases du processus cognitif, les flèches en pointillés les chemins alternatifs possibles. Chaque case représente un groupe d'actions mentales mobilisant des fonctions cognitives[1] qui sont détaillées en fin de chapitre.

En développant ces fonctions cognitives, tu vas développer tes capacités cérébrales et devenir plus intelligent, ce qui va te permettre de résoudre un nombre plus varié de problèmes. Développer une nouvelle fonction cognitive c'est comme apprendre une nouvelle instruction pour l'ordinateur : ça aide à réaliser plus simplement des choses plus complexes.

[1] Cette liste est une compilation des travaux de Reuven Feuerstein revisités par Christine Mayer, Todd Lubart et Pierre Vianin. Références en fin de chapitre.

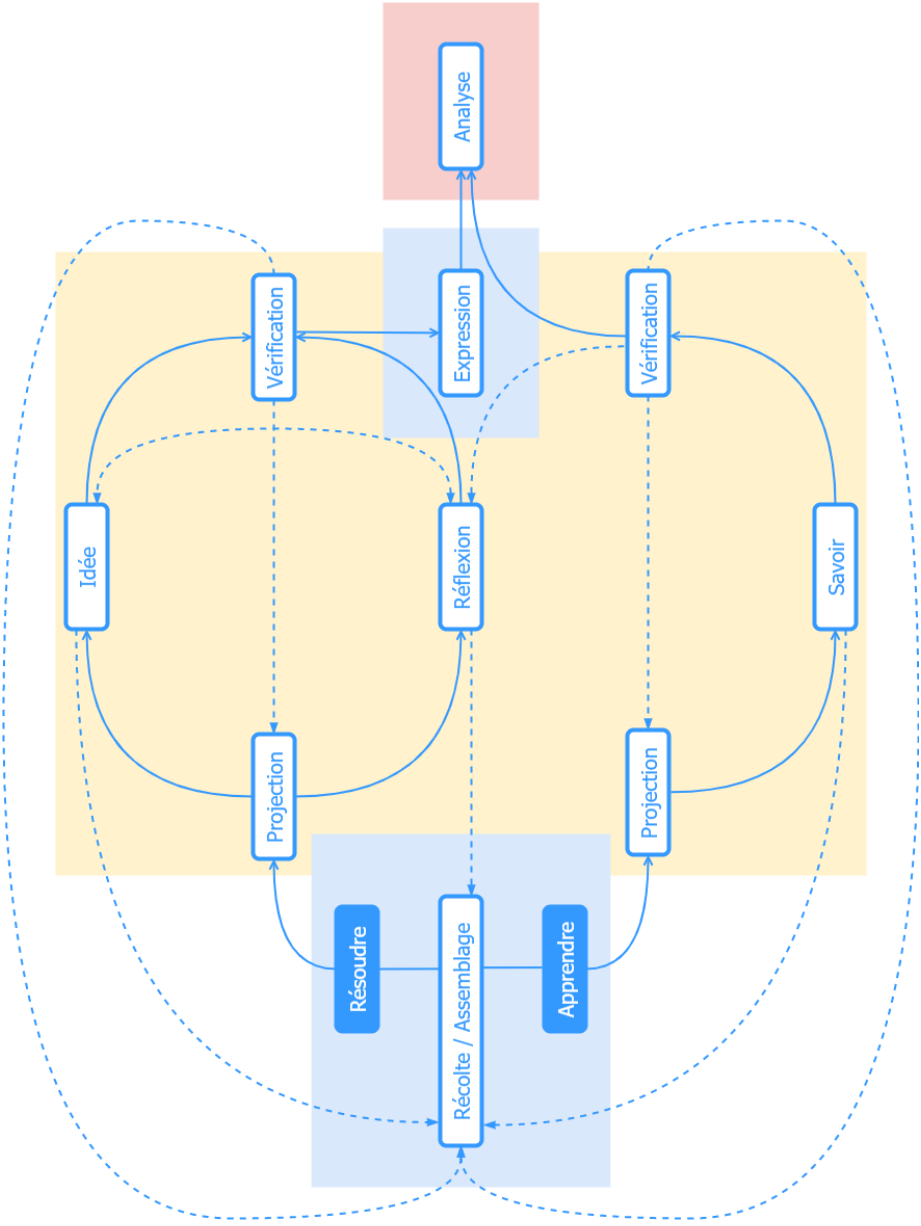


Fig. 12 - Schéma global des processus d'apprentissage et de résolution

La récolte des données :

1. **attention** : Orienter ses sens délibérément et intentionnellement vers une source d'information. Focalise son attention sur ce que je vois et ce que j'entends.
2. **exploration** : Effectuer une observation systématique de la tâche.
3. **méthode** : Examiner tout. Explorer systématiquement les informations disponibles pour ne pas sauter ou oublier quelque chose d'important ou pour ne pas répéter une information.
4. **rigueur** : Être précis et exact dans le rassemblement des données.
5. **identification** : Identifier le type de problème. Déterminer les caractéristiques et les attributs d'un objet en identifiant les ressemblances. Faire des liens avec les expériences passées.
6. **discrimination** : Distinguer des autres objets ou des autres tâches en identifiant les différences.
7. **gestion** : Prendre plusieurs sources d'informations à la fois. Être attentif à plusieurs choses en même temps.

L'assemblage des données :

8. **évocation** : Donner un sens à l'information perçue en élaborant une représentation mentale.
9. **connexion** : Établir des liens, des relations, des associations entre les objets, les événements, les expériences. Avoir une compréhension de la réalité non fragmentée.
10. **sériation** : Classer les éléments dans l'ordre chronologique (séquence temporelle).
11. **sélection** : Trier les informations pertinentes et élimine les informations inutiles.
12. **catégorisation** : Extraire les caractéristiques communes à un ensemble d'objets et constitue des classes sur la base de ces similitudes.
13. **comparaison** : Distinguer les objets entre eux. Déterminer leurs ressemblances et leurs différences.
14. **organisation** : Synthétiser les informations en les organisant en un tout cohérent et en créant des unités de sens.

La projection d’une stratégie de résolution :

15. **reformulation** : Décrire, expliquer, résumer ce qui a été compris.
16. **délimitation** : Choisir le cadre, le contexte et les moyens à disposition pour résoudre le problème.
17. **fluidité** : Ouverture d’esprit. Tolérance à l'ambiguïté.
18. **problématisation** : Définir le problème. Ce que l'on me demande. Ce que j'ai besoin de faire.
19. **anticipation** : Fixer un but et envisager le déroulement et les éventuelles difficultés qui pourraient se présenter.
20. **structuration** : Structurer sa tâche en étapes et en sous-étapes.
21. **planification** : Orienter l'action vers le but fixé et organiser la chronologie des étapes.
22. **modélisation** : Se construire un modèle mental de la chose sur laquelle on est en train de travailler.

La projection d’une stratégie d’apprentissage :

23. **conceptualisation** : Constituer des classes en créant des liens. Extraire les invariants. Négliger les différences pour regrouper les choses. Trouver des similitudes.
24. **récupération** : Récupérer de la mémoire long terme, les connaissances nécessaires à la réalisation de la tâche actuelle.
25. **classification** : Associer des objets semblables ou présentant certaines caractéristiques ou critères communs.
26. **cohérence** : Vérifier que les nouvelles connaissances ne sont pas en contradiction avec les anciennes et que le tout (anciens savoirs et nouveaux savoirs) reste compatible.
27. **réorganisation** : Synthétiser les informations en sélectionnant les éléments importants et en les réorganisant en un tout cohérent.
28. **généralisation** : Construire une superclasse permettant d’englober les nouveaux savoirs aux anciens, en identifiant les éléments de similitudes.

L’élaboration d’une idée :

29. **flexibilité** : Ne pas se bloquer. Ne pas paniquer. Chercher spontanément une autre manière de faire.
30. **désinhibition** : Accepter la prise de risque. Propension à oser.
31. **pensée latérale** : Approcher le problème sous plusieurs angles au lieu de se concentrer sur une approche unique, lorsque celle-ci ne semble pas donner de résultats satisfaisants.
32. **pensée convergente synthétique** : Réaliser des combinaisons sélectives. Synthétiser différents éléments en un seul et même système par combinaison de deux structures complexes.
33. **pensée divergente** : Lorsque la pensée convergente est en échec, penser "out of the box" permet d'envisager de nouvelles alternatives et, parfois, de trouver des solutions originales.
34. **pensée analogique** : Utiliser des comparaisons sélectives par le biais des analogies et des métaphores pour trouver de nouvelles idées.
35. **pensée intuitive** : S'appuyer sur ses propres expériences, ses propres émotions et sentiments.

L’élaboration d’une réflexion :

36. **induction** : Tirer des conclusions générales. Remonte des faits aux lois ou aux règles.
37. **abduction** : Formuler des hypothèses intuitives. Envisage d'autres possibilités tout en cherchant une conclusion concordante aux observations et en éliminant les solutions improbables.
38. **inférence** : Déduire logiquement une proposition en réalisant des liens avec des propositions préalables tenues pour vraies.
39. **déduction** : Tirer une conclusion de la réflexion. Rapport de cause à effet. Si... alors... Chercher les causes, les conséquences.
40. **rétention** : Retenir toutes les informations utiles dans sa tête pour pouvoir les utiliser simultanément. A une vision mentale large.
41. **intériorisation** : Travailler dans sa tête uniquement. A une bonne image mentale de ce qu'il s'agit de chercher ou de faire. Déplace les choses dans sa tête ou avec ses yeux.
42. **concentration** : Faire abstraction des pensées parasites et des perturbations externes (notifications téléphones, bavardages, ...).

L’élaboration d’un savoir :

43. **encodage** : Simplifier l'information apprise pour la mémoriser et la retrouver ultérieurement avec un moindre effort.
44. **boucle phonologique** : Se répéter les mots dans sa tête jusqu’à ce qu’ils transitent en mémoire moyen terme.
45. **bloc notes visuo-spatial** : Utiliser la mémoire visuo-spatiale en invoquant un tableau blanc pour y dessiner mentalement les informations à retenir.
46. **mémoire sémantique** : Enrichir de sens les données à mémoriser. Les contextualiser.
47. **mémoire associative** : Solliciter des associations d’idées pour construire un chemin permettant de retrouver le savoir.
48. **mémoire émotionnelle** : Associer sciemment une émotion forte à l’information désirant être mémorisée.

La vérification d’une réponse :

49. **récapitulation** : Vérifier, compter, inventorier ou résumer, tout ce à quoi j'ai pensé. Faire la synthèse avant de répondre.
50. **justification** : Utiliser un raisonnement logique pour prouver les choses ou pour vérifier leur exactitude.
51. **précision** : Faire attention à l'exactitude des éléments de réponse données.
52. **circonspection** : Ne pas être impulsif. Ne pas se précipiter. Ne pas dire la première chose qui lui passe par la tête.
53. **monitoring** : Surveillance de l'action. Contrôler l'efficacité des stratégies choisies et garder le but en tête.

La vérification d’un savoir :

54. **stabilisation** : Renforcer le chemin d’accès aux données mémorisées.
55. **rémanence** : Vérifier que les données ont bien transité de la mémoire court-terme à la mémoire moyen terme.

L’expression d’une réponse :

56. **régulation** : Ajuster ou réorienter les stratégies utilisées.
57. **décentration** : Traduire la réflexion dans un langage écrit ou oral compréhensible par autrui. Se mettre à la place de l'autre pour être sûr d'être compris. Communiquer de manière non égocentrique.
58. **dénomination** : Connaître et utiliser le bon nom des choses. Avoir le vocabulaire pour décrire les expériences et pour mieux les retenir.
59. **autocontrôle** : Comparer le résultat obtenu avec le but recherché.

L’analyse réflexive :

60. **introspection** : Faire l’effort d’analyser a posteriori l’action réalisée pour en tirer des leçons et maximiser ses apprentissages.
61. **tactique** : Utiliser une combinaison de fonctions cognitives pertinentes, adaptées à la résolution du problème et si possible optimales.
62. **autoévaluation** : Qu'est-ce qui a bien fonctionné ? Ou est-ce que j'ai eu des difficultés ?
63. **transferts** : Envisager l'application à d'autres contextes des apprentissages réalisés.
64. **mémorisation** : Déposer en mémoire long terme les connaissances élaborées durant l'activité.
65. **apprentissages** : Invoquer l’expérience vécue en mémoire de manière espacée dans le temps pour rafraîchir les savoirs et inverser la courbe de l'oubli.

Références :

Lubart, T. **Mouchiroud**, C. **Tordjman**, S. **Zenasni**, F. (2015) *Psychologie de la créativité.* Paris : Armand Colin.

Lubart, T. **Zenasni**, F. **Barbot**, B. (2016). Le potentiel créatif : de la mesure à son développement. Dans I. Capron Puozzo, *La créativité en éducation et formation* (pp. 65-78). Bruxelles : De Boeck Supérieur.

Mayer, C. (2005). *Manuel de métapédagogie.* Menoncourt : Upbraining. En ligne: https://www.upbraining.net/produit/manuel-de-metapedagogie

Vianin, P. (2009). *L'aide stratégique aux élèves en difficulté scolaire – Comment donner à l'élève les clés de sa réussite ?* Bruxelles : De Boeck.

Vianin, P. (2010). *Neurosciences cognitives et pédagogie spécialisée : un exemple d'évaluation diagnostique des processus cognitifs.* Berne : Centre Suisse de Pédagogie Spécialisée. En ligne: https://edudoc.ch/record/102518

12. Histo-ram. Décaplicateur

Nous attaquons la deuxième partie de cette formation. Les exercices vont devenir un peu plus complexes. Pour aider les élèves à déboguer leur code eux-mêmes, nous introduisons un nouvel artefact central pour le débogage des algorithmes : l'histo-ram (Fig. 13). C'est une sorte d'histogramme des valeurs en mémoire (RAM) présentant, dans 6 colonnes, les valeurs manipulées par le code : sortie, mémoire z, y, x, entrée, accumulateur.

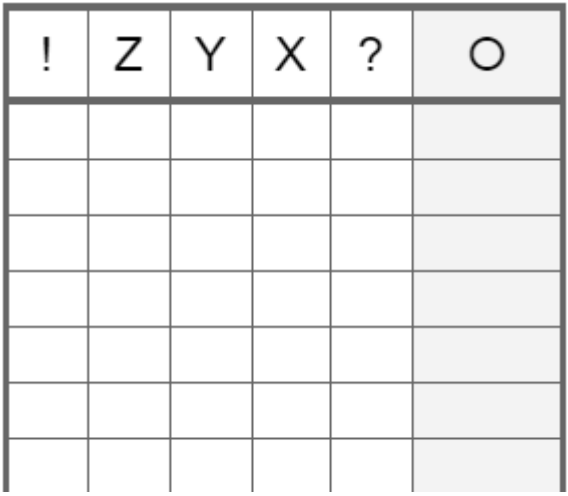


Fig. 13 - Histo-ram pour l'évolution des valeurs en mémoire

Des fiches A4 ou au format des boites peuvent être imprimées en utilisant ces liens : bit.ly/histo-ram-a4 et bit.ly/histo-ram-box

Il faut utiliser une ligne par instruction dans le code. Les valeurs lues sur l'entrée sont notées dans la colonne "?". Comme elles sont recopiées vers l'accumulateur, on dessine une flèche de la valeur inscrite dans "?" vers la colonne "O" en écrivant le nombre deux fois : dans la colonne de départ et d'arrivée. Si la valeur de l'accumulateur est recopiée vers la mémoire "X", on dessine une flèche de la colonne "O" vers la colonne "X" dans laquelle on écrira la valeur recopiée. En cas d'opération (addition ou soustraction) on écrit l'opération dans la colonne "O" et son résultat toujours dans la colonne "O" mais une ligne au-dessous.

Ces règles d'écriture ne sont pas rigides. Il faut les considérer comme des suggestions pour réussir à conserver un histo-ram facile à lire. Il faut donc laisser les élèves les adapter à leur propre manière de voir les choses. Cet outil est une aide et ne doit pas alourdir leur charge cognitive. Si un élève trouve une manière plus logique et élégante de noter l'historique des valeurs en mémoire, il pourra la présenter au reste de la classe pour que nous adoptions les mêmes conventions.

Cet artéfact aide les élèves à élaborer dans leur tête une représentation mentale des différents états et valeurs présentes dans le processeur. La construction d'une image mentale de l'architecture du système est, en effet, indispensable pour réussir à penser le code. C'est un prérequis à la pensée algorithmique. De plus, il leur donne la possibilité, moyennant un petit effort, d'obtenir une rétroaction sur leur prédiction, tout en apportant une explication dans un délai court. La combinaison de tous ces paramètres : savoir, juste après avoir écrit son algorithme, si le code imaginé marche ou pas (et s'il ne marche pas, voir pourquoi) permet au cerveau de l'élève de minimiser son énergie libre (au sens de Friston). L'artefact histo-ram va produire dans l'environnement de l'élève et moyennant un petit peu d'effort de

sa part, une preuve que le modèle mental de l'ordinateur qu'il aura mobilisé lors de la conception de l'algorithme est efficient ou pas : c'est à dire s'il permet de prédire ce qui se passe réellement dans le processeur de l'ordinateur. Si le résultat obtenu en sortie est faux, c'est qu'il y a eu, dans les étapes précédentes, un déphasage entre ce que l'élève avait imaginé des états de la mémoire et ce qui se passe réellement. La surprise ainsi créée va, par renforcement négatif, modifier le système de représentation mentale du processeur chez l'élève et devrait le conduire à affiner sa compréhension du modèle et à s'auto-corriger.

Le fait de pouvoir prendre conscience, seul, de l'inexactitude de sa propre représentation mentale utilisée pour prédire l'état du système, va, par effet d'autonomisation, renforcer chez l'élève sa conviction dans sa capacité à réussir à se corriger et donc à apprendre.

Ainsi, les bénéfices de l'histo-ram sont multiples :

- permet de savoir si la solution proposée marche ou pas.
- voir d'où vient l'erreur en cas d'erreur et d'observer la différence entre ce qui a été prévu et ce qui se passe réellement en apportant une preuve et une explication.
- apporte une explication qui va permettre à l'élève de modifier sa représentation mentale et diminuer les prochaines mauvaises surprises.
- génère un état de plaisir dans la prise de conscience de sa capacité à améliorer son processus de prédiction de ce que le processeur va faire en fonction du code fournis.
- engendre un renforcement positif lorsque la prédiction est réussie.

10000. Décaplicateur.

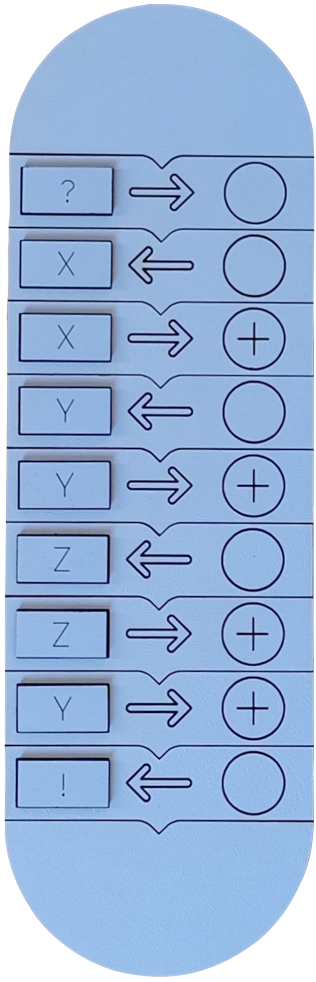
Multiplie par dix la valeur entrée.
Affiche le résultat sur la sortie.
Challenge : maximum 4 additions.

Indice : 10 = 8 + 2.

une solution possible

```
BEGIN
   READ [?]
   WRITE[X]
   ADD  [X]
   WRITE[Y]
   ADD  [Y]
   WRITE[Z]
   ADD  [Z]
   ADD  [Y]
   WRITE[!]
END
```

!	Z	Y	X	?	O
				2 >>	>> 2
			2 <<		<< 2
			2 >>		>>2+2
		4 <<			<< 4
		4 >>			>>4+4
	8 <<				<< 8
	8 >>				>>8+8
		4 >>			>>16+4
20 <<					<< 20



Explicitation du processus de réflexion

- Lire l'énoncé.
- Faire le lien avec l'exercice *octoplicateur* précédent.
- Doubler fois 8 fait du fois 16, or nous voulons du fois 10.
- Penser à conserver le résultat fois 2 intermédiaire pour l'additionner au fois 8 de sorte que 2+8=10.

#FAIRE_DES_LIENS
#RAISONNEMENT_LOGIQUE
#PENSEE_CREATIVE

13. Les 10 commandements du parfait codeur

Quelques conseils pour réussir.

Programmer ne s'improvise pas. Ce n'est pas quelque chose d'inné et comme beaucoup d'autres choses, cela s'apprend. Voici 10 conseils pour t'apprendre à devenir un bon programmeur.

1. **Observe bien comme il faut.**
2. **Cherche à faire des connexions entre les choses.**
3. **Pose toi des questions et fait des hypothèses.**
4. **Travaille avec méthode et rigueur.**
5. **Visualise ce qu'il faut faire avant de le faire.**
6. **Cherche plusieurs manières de résoudre le problème.**
7. **Varie les techniques et les stratégies.**
8. **Si tu es bloqué, pense autrement, hors des sentiers.**
9. **Persévère malgré les échecs.**
10. **Cherche des idées en groupe.**

Nous développerons quelques-unes de ces stratégies par la suite et nous en donnerons d'autres. Il faut garder à l'esprit que résoudre des problèmes algorithmiques pour ensuite pouvoir les programmer n'est pas quelque chose de simple. Les solutions ne sont pas évidentes, mais avec un peu de bon sens, d'astuce et de patience, tout se construit petit à petit pour ceux qui en font l'effort.

human processor!

Manuel Enseignant

LIVRET 2

Problèmes

HEP Lausanne - Juin 2023

http://human-processor.xyz

Livret 2 - Problèmes

Le dispositif didactique débranché "*human processor!*" et ses exercices sont inspirés du jeu "*human resource machine*" développé en 2015 par Kyle Gabler, Allan Blomquist et Kyle Gray de la société Tomorrow Corporation.

Les plans des pièces en bois de la boîte de jeu, le présent manuel ainsi que la liste des exercices et les fiches des stratégies cognitives, sont disponibles sur :
http://human-processor.xyz

Le présent manuel propose, en alternance :

- 10 activités débranchées sur le thème de l'algorithmie qui utilisent les pièces en bois pour programmer ;
- 7 activités réflexives ayant pour but d'aider les enseignants à développer l'intelligence algorithmique[1] de leurs élèves.

Ce manuel est accessible au format numérique sur Google Doc. Il est possible de faire des commentaires directement sur le document en ligne :
https://bit.ly/manuel_enseignants

Pour toute information complémentaire :
christian.blanvillain@hepl.com

[1] Nous définissons l'intelligence algorithmique comme étant le produit des trois intelligences logico-mathématique, pratique et créative (**Robert J. Sternberg.** 2015. *Beyond I.Q.: A triarchic theory of human intelligence.* New York: Cambridge University Press.)

14. Multiplication égyptienne

CANEVAS

Algorithme

Soient deux nombres entiers J et K. Nous souhaitons calculer leur produit J*K. Supposons que J soit plus petit que K. Considérons sa représentation binaire. Écrivons là en colonne avec le premier bit (de parité) en haut. En face du premier bit, écrivons K en binaire. Puis, sur la ligne suivante, écrivons 2*K, c'est-à-dire ajoutons un 0 à la représentation binaire de K. Répétons cette opération pour l'ensemble des lignes représentées par les bits de J. Ignorons les lignes ou le bit de J est à 0. Effectuons une addition avec les nombres en face des lignes dont le bit est à un. La somme obtenue correspond au produit de J par K.

Exemple

Soit J=10 et K=14, c-à-d $J=1010_2$ et $K=1110_2$.

J_2	K_2	$J*K_2$	K_{10}	$J*K_{10}$
0	1110_2	0_2	14	0
1	11100_2	11100_2	28	28
0	111000_2	0_2	56	0
1	1110000_2	1110000_2	112	112
		10001100_2		140

Explication

2^0 = 1 **= 000001_2**
2^1 = 2 **= 000010_2**
2^2 = 2x2 = 4 **= 000100_2**
2^3 = 2x2x2 = 8 **= 001000_2**

10 = **8 + 2 = 2x2x2 + 2**
10 = **1x8 + 0x4 + 1x2 + 0x1**
10 = **1***2^3 **+ 0***2^2 **+ 1***2^1 **+ 0***2^0
10 = **1010_2 en base binaire**

1010*1110 = (1000+10)*1110 = 1000*1110+10*1110
= 1110000+11100 = 10001100

Activité

Nous avons vu, avec l'exercice 00110. Triplicateur, comment multiplier un nombre par lui-même 3 fois. Proposer aux élèves de généraliser, en commençant par écrire l'algorithme pour multiplier par 4, puis par 8. Demandez-leur comment écrire la multiplication par 8 en utilisant qu'une addition de plus que pour la multiplication par 4. Demandez-leur alors d'écrire la multiplication par 4 avec seulement deux additions. Maintenant, demandez-leur d'écrire la multiplication par 10 avec un nombre minimal d'addition. (Ces deux derniers problèmes seront retravaillés en petits groupes la semaine prochaine).

Faire remarquer que 10 s'écrit 2+8, et que 8 est une puissance de 2. Demander de multiplier par 16, puis par 15. Faire le lien avec la machine à calculer mécanique[1] (si elle a été présentée en classe)

Expliquer la représentation binaire des nombres et prendre comme exemple la numérotation des exercices. Décomposer un nombre décimal en binaire et inversement. Montrer que multiplier par deux revient à ajouter un zéro. Enfin, présenter l'algorithme de la multiplication égyptienne.

[1] https://www.youtube.com/watch?v=orzJoSVQ_Io

15. Heptaplicateur. Compteur.

10001. Heptaplicateur.

Multiplie par sept la valeur entrée.
Affiche le résultat sur la sortie.
Challenge : utilise uniquement les pièces d'une seule boite.

Indice : 7 = 8-1.

une solution possible

```
BEGIN
   READ [?]
   WRITE[X]
   ADD  [X]
   WRITE[Y]
   ADD  [Y]
   WRITE[Z]
   ADD  [Z]
   SUB  [X]
   WRITE[!]
END
```

!	Z	Y	X	?	O
				2 >>	>> 2
			2 <<		<< 2
			2 >>		>>2+2
		4 <<			<< 4
		4 >>			>>4+4
	8 <<				<< 8
	8 >>				>>8+8
			2 >>		16 - 2
14 <<					<< 14

?
X
X
Y
Y
Z
Z
X
!

Explicitation du processus de réflexion

- Lire l'énoncé.
- Faire des liens avec les exercices précédents.
- Décomposer 7 en somme de multiples de 2 = 4+2+1.
- Réaliser que le nombre de pièces du dispositif ne suffit pas pour écrire toutes ces additions.
- Relire l'énoncé et découvrir le challenge indiqué.
- Chercher une nouvelle idée.
- Réaliser que jusqu'à présent, on a fait surtout des multiples de 2.
- Penser au fait que multiplier par 8 est assez simple à obtenir.
- Penser au fait que nous disposons de l'opération de soustraction.
- Penser que 7 = 8 - 1.

#FAIRE_DES_LIENS
#RAISONNEMENT_LOGIQUE
#PENSEE_CREATIVE

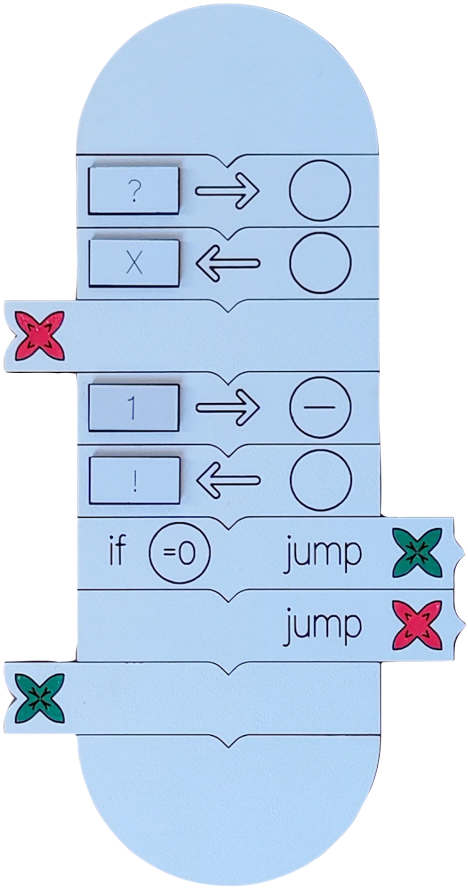
10010. Compteur

Lis un entier positif.
Affiche tous les entiers plus petits jusqu'à 0 sur la sortie.

une solution possible

```
BEGIN
   READ [?]
[0]:
   IF  [=0] JUMP [1]
   SUB  [1]
   WRITE[!]
   JUMP [0]
[1]:
END
```

!	Z	Y	X	?	O
				4 >>	>> 4
					4 - 1
3 <<					<< 3
					3 - 1
2 <<					<< 2
					2 - 1
1 <<					<< 1
					1 - 1
0 <<					<< 0


?
X
1
!
if
=0
jump
jump

Explicitation du processus de réflexion

- Lire l'énoncé.
- Lire les nombres fournis en exemple.
- Comprendre qu'il faut faire un compte à rebours.
- Penser à l'opération de soustraction de l'exercice précédent.
- Se rappeler qu'il existe une constante qui vaut 1 dans le jeu de pièces.
- Se dire qu'un jump permet de répéter l'opération de soustraction.
- Penser qu'il ne faut pas descendre en dessous de zéro.
- Se souvenir que l'on peut faire un saut si la valeur est zéro.
- Placer les bonnes étiquettes pour le saut inconditionnel et pour le saut conditionnel, au bon endroit dans le code.

#RAISONNEMENT_ALGORITHMIQUE

16. Se mettre au travail

Éloge de l’effort.

C’est un leurre de croire que le talent seul suffit pour réussir. Beaucoup de travail sont également nécessaires. Il faut constamment s’opposer à la tentation de la facilité et faire chaque jour des efforts pour vaincre notre oisiveté naturelle qui s’oppose à notre épanouissement. Résister à la tentation du plaisir immédiat, c’est prendre le contrôle de soi. Le plaisir se découvre lorsque l’on atteint un résultat qui nécessite de dépasser nos limites, preuve que nous avons un pouvoir sur notre propre avenir, un contrôle de notre destin et une autonomie qui nous permet de choisir notre vie et décider de la direction vers laquelle nous souhaitons évoluer. En enseignant aux élèves le goût de l’effort, nous leur offrons la possibilité de redéployer ce plaisir de surpassement de soi dans leur quotidien, dans leur travail. Enseigner aux élèves le contrôle de leur vie, c’est leur apprendre, par effet de bord, à contrôler la société et le monde qu’ils construiront demain. À l’école, il est primordial d’apprendre à apprendre car l’effort seul ne suffit pas, il faut en plus avoir les bonnes stratégies pour pouvoir devenir intelligent. Mais ce qui est encore plus important, c’est le plaisir de l’accomplissement de soi au travers de son travail. Voici donc, pour moi, l’objectif principal que devrait se donner tout enseignant.

Débattre avec les élèves.

Une stratégie cognitive est “une procédure ou un ensemble de procédures utilisées pour accomplir un but” (Patrick Lemaire et Lynne Reder[1], 1999, p. 35). “C’est une suite d’opérations mentales que met en œuvre un individu pour accomplir une tâche cognitive“ (Patrick Lemaire et André Didierjean[2], 2018, p. 303). Nous présentons 14 stratégies cognitives qui nous semblent pouvoir aider les élèves à développer leur aptitude à résoudre des problèmes. Ces stratégies, tout comme la liste des 64 fonctions cognitives du chapitre 8, n’ont pas pour objectif d’être présentées de manière magistrale aux élèves : cela ne servirait à rien. Elles sont là plutôt pour l’enseignant, de manière à ce qu’il puisse savoir comment aider un élève, lorsqu’il détecte une lacune soit au niveau d’une fonction cognitive (qu’il faudra alors renforcer par le biais d’exercices plus ciblés), soit au niveau de la démarche de résolution de problèmes (et une stratégie viendrait alors supposément débloquer l’élève). En effet, ce n’est que si l’élève trouve, au travers de l’aide apporté par l’enseignant, le moyen de se débloquer et de penser, par lui-même, à la solution au problème posé, qu’il assimilera la stratégie utilisée et pourra effectuer des transferts à d’autres situations similaires ou d’autres disciplines.

La liste des stratégies n’est pas figée. C’est au groupe classe d’en identifier de nouvelles en situation, au travers de discussions et de partages. Souvent les élèves qui ont trouvé une solution ne parviennent pas à exprimer le chemin qu’ils ont pris pour y arriver. Notre rôle d’enseignant consiste alors à tenter de leur faire repenser aux actes réalisés au moment de la résolution, aux pensées qui sont apparues et d’analyser, grâce à cette approche

[1] **Lemaire**, P. **Reder**, L. (1999). *What affects strategy selection in arithmetic?* Memory & Cognition, 27(2), 364-382.
[2] **Lemaire**, P. **Didierjean**, A**.** (2018). *Introduction à la psychologie cognitive.* Bruxelles : De Boeck Supérieur.

explicitative (au sens de Pierre Vermersch), les gestes mentaux effectués. Au début de l'apprentissage, durant la phase de découverte du dispositif au travers des exercices tutoriaux, il convient de ne pas trop varier le contexte d'apprentissage pour faciliter le renforcement des liens, encore faibles, qui se tissent dans les esprits des élèves. À ce stade de la formation nous avons déjà presque trois mois de pratique. Nous pouvons donc nous permettre de varier les contextes d'apprentissage et de décontextualiser les exercices afin de permettre les transferts des acquis. L'expression des stratégies cognitives peut donc se faire en élargissant le champ des exemples considérés durant les discussions de groupe ou bien en revenant sur les premiers exercices afin de faire des répétitions et de consolider les apprentissages.

Voici quatre stratégies susceptibles d'aider les élèves à se préparer à travailler et pour se mettre à réfléchir :
CAPABLE, FOCUS, PENSE, PAUSÉ.

Je suis capable de le faire.

Je me concentre pour réussir.

Je résiste à donner la première réponse sans vérifier.

Je pense avant d'agir.

CAPABLE

Je suis capable de le faire.

Qu'est-ce que je dois faire ? Est-ce que j'ai bien toutes les données ? C'est ok de ne pas y arriver du premier coup ou de faire des erreurs : garde confiance en toi. Si c'était trop facile, tu n'apprendrais rien.

Img. 2 - Image non libre de droits. Source : https://www.pinterest.co.uk/pin/412360909626776258/

Pose-toi la question de savoir si tu as bien toutes les connaissances utiles pour résoudre le problème. Essaye de l'expliquer à quelqu'un d'autre pour t'assurer que tu as bien compris, ou même de te l'expliquer à toi-même en reformulant le problème avec tes mots. Si tu as des difficultés, peut-être que le prof a utilisé des mots que tu ne connais pas ou bien peut-être que tu as mal écouté en n'était pas super attentif tout le temps, ce qui est tout à fait normal. Ne reste surtout pas sans rien oser faire. N'hésite pas à demander qu'on te ré-explique les choses.

FOCUS

Je me concentre pour réussir.

Est-ce que tu as besoin qu'on te résume ce qu'il faut faire ? Focalise ton attention sur la tâche que tu es en train d'accomplir. Tu penseras au reste plus tard. Il y a un temps pour apprendre et un temps pour s'amuser.

Img. 3 - Image non libre de droits. Source : https://www.thefearlessman.com/wp-content/uploads/2016/08/focused.jpg

Pendant que tu réfléchis, évite toute forme de distraction : un bruit, un mouvement et tu peux perdre le fil de ta pensée, oublier ce à quoi tu étais en train de réfléchir. La base de la concentration c'est donc tout simplement le calme. Et si tu ne veux pas faire le calme pour toi, fais-le pour les autres.

PAUSÉ

Je résiste à donner la première réponse sans vérifier.

Est-ce que tu as pris le temps avant de donner ta réponse ? Vérifie bien l'exactitude de ton idée. Parfois la vérité se cache ailleurs. Adopter une approche critique et réfléchie permet d'éviter les pièges.

Img. 4 - Image non libre de droits. Source : https://cdn.fioulreduc.com/blog/wp-content/uploads/2019/01/true-false.jpg

Se demander ce que la personne qui t'a posé le problème a derrière la tête, ce qu'elle attend de toi que tu fasses, c'est déjà résoudre à moitié le problème.

PENSE

Je pense avant d'agir.

Comment ne pas répondre au hasard ? Demande-toi si tu fais ça, alors qu'est-ce qui se passe ? Tu exploreras très rapidement beaucoup plus de possibilités en raisonnant dans ta tête qu'en essayant pour de vrai.

Img. 5 - Image non libre de droits. Source : https://www.emarkable.ie/wp-content/uploads/2018/12/code.png

Il ne s'agit pas de deviner les bonnes réponses. Il s'agit de construire un raisonnement, une réflexion, qui te mènera à la solution au problème.

17. Egalisateur. Distanciateur.

10011. Egalisateur.

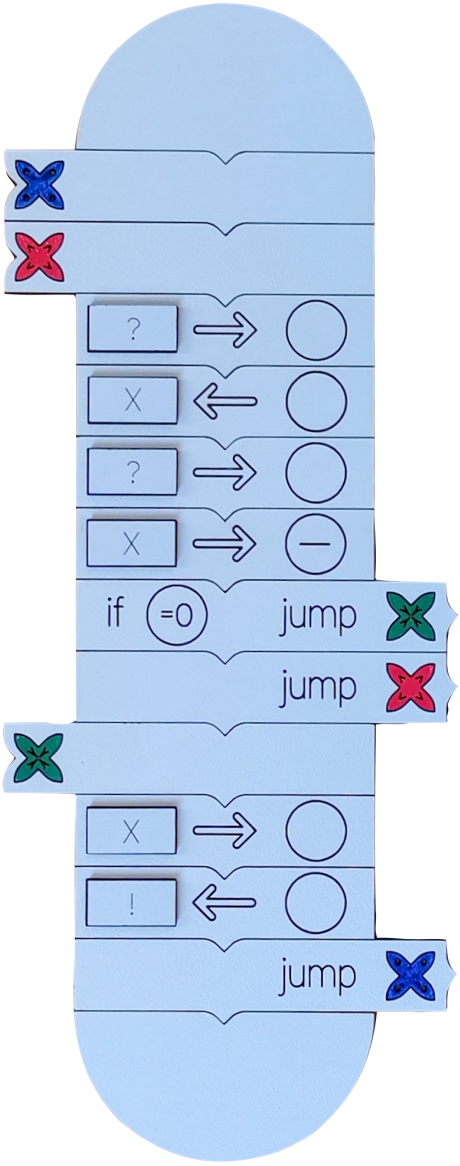
Pour chaque paire de nombres, s'ils sont différents ne fais rien, s'ils sont égaux recopie l'un d'entre-eux sur la sortie. Challenge : 8 instructions.

Indice : si deux nombres sont égaux leur différence est nulle.

une solution possible

```
BEGIN
[0]:
[1]:
   READ [?]
   WRITE[X]
   SUB  [?]
   IF   [=0] JUMP [2]
   JUMP [1]
[2]:
   READ [X]
   WRITE[!]
   JUMP [0]
END
```

!	Z	Y	X	?	O
				2 >>	>> 2
			2 <<		<< 2
				0 >>	>>2-0
				5 >>	>> 5
			5 <<		<< 5
				5 >>	>>5-5
			5 >>		>> 5
5 <<					<< 5


?
X
?
X
if =0
jump
jump
X
!
jump

Explicitation du processus de réflexion

- Lire l'énoncé.
- Lire l'exemple fourni.
- Se demander comment faire pour pouvoir comparer deux nombres.
- Essayer de trouver un lien entre la comparaison de deux nombres et les instructions disponibles.
- Les seuls branchements conditionnels utilisent =0 ou <0.
- Penser différemment.
- (1) Réfléchir à l'aspect géométrique des nombres. Se représenter un axe. Penser à calculer la distance qui sépare deux nombres. Réaliser que lorsque deux nombres sont égaux leur distance est nulle.
- (2) Réfléchir au nombres comme des poids. Se représenter une balance à plateaux. Penser à calculer la différence des poids pour savoir de quel côté la balance penche. Il y a équilibre en cas d'égalité.
- (3) Réfléchir au nombre comme des distances. Se représenter deux distances identiques. Lorsqu'elles sont au même endroit, leurs distances est nulle.
- Revenir au code.
- Stocker tous les nombres en mémoire pour pouvoir effectuer les opérations demandées.
- Lorsque le code est écrit, vérifier qu'il est juste.
- Lorsque le code est juste, lire le challenge.
- Se rendre compte qu'il n'est pas nécessaire de mémoriser les deux nombres puisque c'est seulement lorsqu'ils sont identiques que l'on doit afficher la valeur en sortie.

#RAISONNEMENT_LOGIQUE
#RAISONNEMENT_ALGORITHMIQUE (Challenge)

10100. Distanciateur.

Pour chaque paire de nombres, soustrait d'abord le second au premier, envoie le résultat sur la sortie puis fais l'inverse : soustrais le premier du second, envoie le résultat et recommence. **Challenge : 8 instructions.**

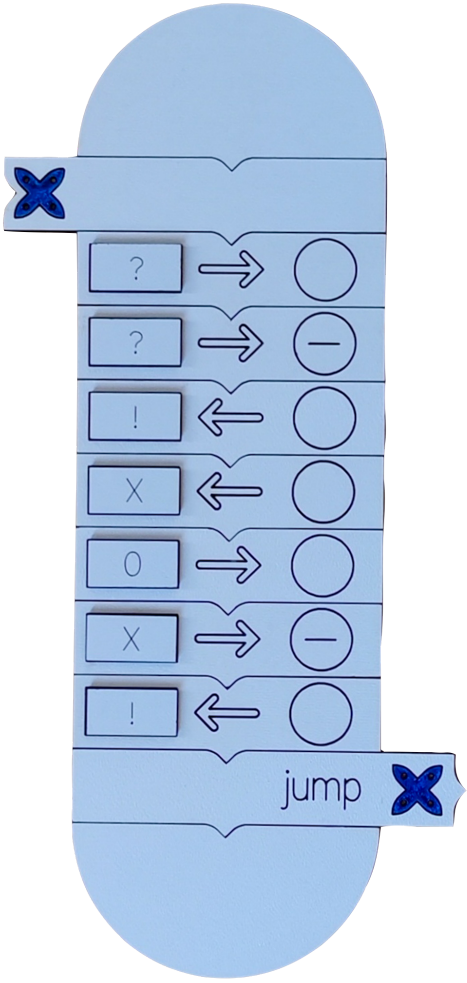
Rappel : La convention choisie est de soustraire la nouvelle valeur avec celle qui est déjà dans l'assiette.

Indice : c'est comme en géométrie lorsque l'on calcule la distance entre deux points : le signe dépend de l'ordre dans lequel on considère les points.

une solution possible

```
BEGIN
[0]:
   READ [?]
   SUB  [?]
   WRITE[!]
   WRITE[X]
   READ [0]
   SUB  [X]
   WRITE[!]
   JUMP [0]
END
```

!	Z	Y	X	?	O
				1 >>	>> 1
				9 >>	>>1-9
-8 <<					<< -8
			-8 <<		<< -8
					0
			-8 >>		0 - (-8)
8 <<					<< 8


?
?
!
X
0
X
!
jump

Explicitation du processus de réflexion

- Lire l'énoncé.
- Lire l'exemple fourni.
- Lire deux nombres et les mémoriser dans X et Y.
- Calculer la première différence X-Y et afficher le résultat.
- Calculer la deuxième différence Y-X et afficher le résultat.
- Déboguer le code.
- Lire le challenge.
- Observer le code débogué.
- Remarquer que les deux valeurs calculées sont identiques en valeur absolue et que seul leur signe change. Faire le parallélisme entre la distance géométrique de deux points sur un axe.
- Réfléchir à comment inverser le signe de la valeur calculée.
- Se souvenir que dans les pièces disponibles nous avons la constante zéro.
- Constater que zéro moins une valeur permet d'inverser le signe de la valeur.
- Lire un premier nombre.
- Soustraire le deuxième nombre.
- Afficher le premier résultat.
- Inverser le signe en soustrayant à zéro la valeur calculée.
- Afficher le deuxième résultat.

#RAISONNEMENT_LOGIQUE
#PENSEE_CREATIVE

18. Résoudre des problèmes

Voici quatre stratégies susceptibles d'aider les élèves à trouver des solutions aux problèmes proposés :
LIENS, IMAGE, MENTAL, SIMPLE.

Je fais des liens avec ce que je connais.

Je me construis une image mentale du problème.

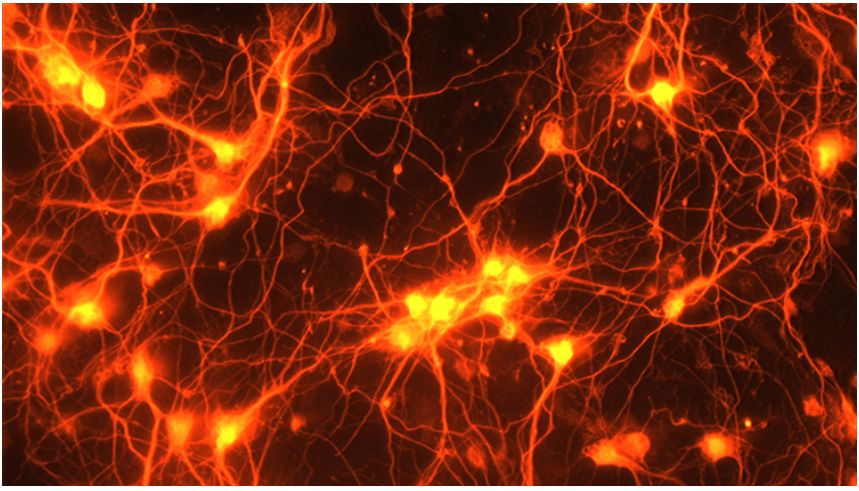
Je manipule mon image mentale.

Je cherche à simplifier le problème.

LIENS

Je fais des liens avec ce que je connais.

Qu'est-ce qui est pareil ? Qu'est-ce qui est différent ? Cherche des similitudes avec ce que tu sais déjà et fais une hypothèse "c'est comme si…". Observe les différences et essaye de voir si ton intuition est juste.



Img. 6 - Image non libre de droits. Source : https://bit.ly/2sFJNEW

Cherche à faire des liens avec ce que tu connais déjà pour mieux comprendre et contextualiser le problème, cherche des similitudes qui pourraient t'aider à avoir de nouvelles idées et à mobiliser ton intelligence pratique pour résoudre un problème qui pourrait sembler *a priori* abstrait.

IMAGE

Je me construis une image mentale.

Qu'est-ce qui est utile ? Qu'est-ce qui ne l'est pas ? Qu'est-ce que je garde dans ma tête ? Représente-toi visuellement ce qu'il faut faire avant de commencer, ça va t'aider à mieux réfléchir.

Img. 7 - Image non libre de droits. Source : https://www.pinterest.ch/pin/296117744252994211/

Dans ta tête, tu peux essayer plein de choses, vérifier si c'est juste, te tromper et recommencer, dix fois plus vite qu'en vrai. Utilise ce pouvoir pour visualiser ce que tu penses que tu pourrais faire.

Je manipule mon image mentale pour résoudre le problème.

Utilise l'image que tu as construite dans ta tête pour chercher une solution avant de l'essayer pour de vrai. Déplace les choses avec tes yeux pour vérifier tes idées.

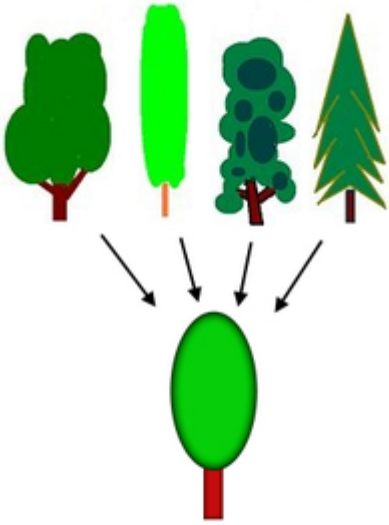
Img. 8 - Image non libre de droits. Source : https://www.emarkable.ie/wp-content/uploads/2018/12/code.png

À force de réfléchir avec les pièces devant toi, tu finiras par ne plus avoir besoin de leur aide pour réfléchir. Tu vas pouvoir, très rapidement, déplacer les valeurs dans ta tête pour prédire les prochains états du système et voir si tes idées sont bonnes ou pas.

SIMPLE

Je cherche à simplifier le problème.

Découpe le problème en plus petits problèmes ou bien résous un problème semblable, mais plus simple. Qu'est-ce qui est constant ? Qu'est-ce qui est variable ? Essaye de généraliser ta solution. Si ce n'est pas suffisant, découpe le problème en plus petits problèmes.



Img. 9 - Image libre de droits. Source : https://upload.wikimedia.org/wikipedia/commons/thumb/f/f3/Generalization_process_using_trees_PNG_version.png

Réduit le problème à quelque chose que tu sais résoudre. Cherche des problèmes similaires et étudie leurs différences. Puis quand tu as trouvé un plus simple qui marche, complexifie le, peu à peu, pour revenir au problème initial.

19. Maximisateur. Positiveur.

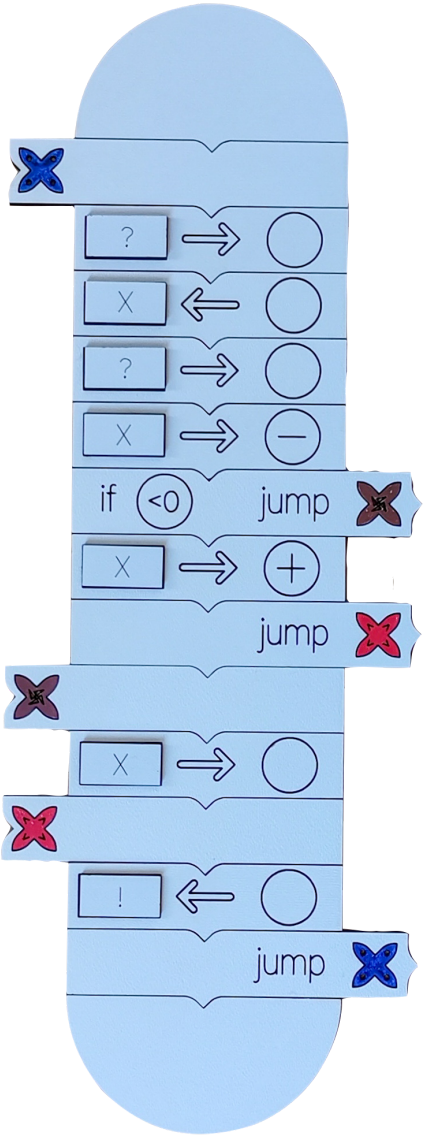
10101. Maximisateur.

Pour chaque paire de nombres, recopie le plus grand des deux sur la sortie. **Challenge : 10 instructions.**

une solution possible

```
BEGIN
[0]:
   READ [?]
   WRITE[X]
   READ [?]
   SUB  [X]
   IF   [<0] JUMP [2]
   ADD  [X]
   JUMP [1]
[2]:
   READ [X]
[1]:
   WRITE[!]
   JUMP [0]
END
```

!	Z	Y	X	?	O
				3 >>	>> 3
			3 <<		<< 3
				5 >>	>> 5
			3 >>		>>5-3
			3 >>		>>2+3
5 <<					<< 5
				-4 >>	>> -4
			-4 <<		<< -4
				-3 >>	>> -3
			-4 >>		-3 - (-4)
			-4 >>		1+(-4)
-3 <<					<< -3


?
X
?
X
if
<0
jump
X
jump
X
!
jump

Processus de résolution détaillé (lien fonctions cognitives)

[méthode - rigueur]
Pour chaque => 1 JUMP inconditionnel
Paire de valeurs => 2 instructions READ
Recopie sur la sortie => 1 instruction WRITE
Total 4 instructions. Indice : solvable en 10 instructions.

[reformulation]
Comment, en 6 instructions, identifier la plus grande valeur ? Nous n'avons pas d'opérateur comparaison mis à part celui utilisé pour savoir si l'accumulateur est négatif ou nul. Comment relier le fait que l'accumulateur est négatif ou nul, au fait que deux nombres seraient identiques ou différents ?

[structuration]
J'ai besoin de comparer deux valeurs => Je dois utiliser au moins une mémoire et l'accumulateur (voir deux mémoires). *Lire un nombre et le mémoriser. Lire un deuxième nombre et le mémoriser.*

[inférence - récupération]
Faire des liens avec les exercices précédents. Nous avons déjà utilisé le fait que deux nombres identiques ont une différence nulle dans l'exercice de l'*Egalisateur*. Penser au fait que lorsque deux nombres sont différents, leur différence X-Y est positive si X est plus grand que Y, négative si Y est plus grand que X. *Calculer la différence des deux nombres. Si le résultat est négatif, afficher le deuxième nombre. Sinon afficher le premier nombre.*

[pensée latérale]
Lire le challenge. Compter le nombre d'instructions. Nous avons une ligne de trop. Pour faire plus court, je peux peut-être n'utiliser qu'une seule mémoire et, après avoir effectué la soustraction, inverser la soustraction pour retrouver la valeur précédente si besoin.

[induction - intériorisation]
Compte tenu de l'idée et de l'inférence ci-dessous, je met la première valeur en mémoire. Je lis la seconde valeur. Puis je soustrait la première à la seconde valeur (qui est toujours dans l'accumulateur). Je teste si le résultat est négatif (je ne peux pas tester autre chose).

[monitoring]
A ce niveau, je ne sais pas quoi écrire dans mon test, car, du fait que je suis dyslexique, je n'arrive pas à visualiser qui est le plus grand des deux. Je compte les instructions déjà utilisées, il en reste 2 disponibles.

[récapitulation]
Mais une des deux instructions est (voir l'idée) d'ajouter la valeur en mémoire à l'accumulateur et l'autre est de lire ce qui avait en mémoire.

[précision]
Je code ce qui précède et je réfléchis aux trois dernières instructions qui me manquent en regardant mon code.

[autocontrôle]
Après avoir codé, je vérifie que le code fait bien ce que je pensais qu'il devait faire avec *l'historam* ou bien en relisant dans ma tête.

[introspection]
Pour rédiger cette analyse, je repasse au ralenti les différentes étapes de résolution.

[transferts]
Je me dis que cette analyse peut être étendue à tous les exercices et pourrait servir pour les élèves en difficulté à suivre mon raisonnement.

#FAIRE_DES_LIENS
#RAISONNEMENT_LOGIQUE
#PENSEE_CREATIVE (Challenge)

10110. Positiveur.

Pour chaque nombre, s'il est positif, recopie-le directement sur la sortie, sinon retire-lui d'abord son signe.
Challenge : 8 instructions.

Indice : il existe deux manières de retirer le signe négatif d'un nombre, soit en le soustrayant deux fois à lui-même, soit en le soustrayant directement à zéro.

une solution possible

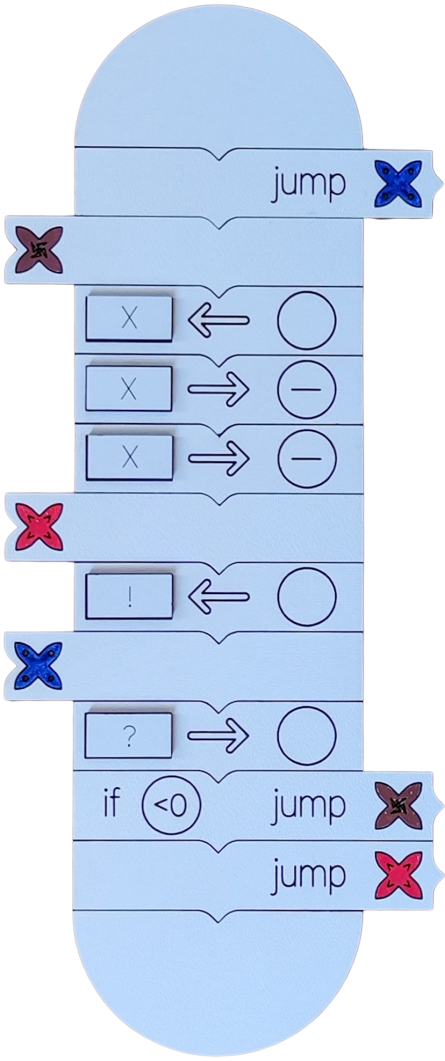
```
BEGIN
   JUMP [2]
[0]:
   WRITE[X]
   SUB  [X]
   SUB  [X]
[1]:
   WRITE[!]
[2]:
   READ [?]
   IF  [<0] JUMP [0]
   JUMP [1]
END
```

!	Z	Y	X	?	O
				5 >>	>> 5
5 <<					<< 5
				-4 >>	>> -4
			-4 <<		<< -4
			-4 >>		-4 -(-4)
			-4 >>		0 - (-4)
4 <<					<< 4

une autre solution

```
BEGIN
[2]:
   READ [?]
   IF  [<0] JUMP [0]
[1]:
   WRITE[!]
   JUMP [2]
[0]:
   WRITE[X]
   READ[+0]
   SUB  [X]
   JUMP [1]
END
```

!	Z	Y	X	?	O
				5 >>	>> 5
5 <<					<< 5
				-4 >>	>> -4
			-4 <<		<< -4
					0
			-4 >>		0 - (-4)
4 <<					<< 4


jump
X
X
X
!
?
if
<0
jump
jump

Explicitation du processus de réflexion

- Lire l'énoncé.
- Faire le lien avec l'exercice précédent Distanciateur pour inverser le signe d'un nombre, il suffit de le soustraire à zéro.
- Lire un nombre.
- S'il est négatif, mémoriser ce nombre, puis le soustraire de zéro pour inverser son signe.
- Afficher le résultat sur la sortie.
- Recommencer.
- Lire le challenge.
- Le code fait actuellement bien 8 instructions de long.

#FAIRE_DES_LIENS

20. Lorsque l’on est bloqué

Voici trois stratégies susceptibles d’aider les élèves à avoir des idées pour se débloquer : CRÉATIF, FLEXIF, GROUPE.

J’explore une autre piste.

Je change de chemin.

Je trouve de nouvelles idées.

J’utilise l’intelligence du groupe.

BACKTRACK

J’explore une autre piste.

Lorsqu’une piste de réflexion ne mène à rien, faire marche arrière et envisager d’autres variations alternatives partout où un choix a été fait.

Img. 10 - Image non libre de droits. Source : https://cdn.xxl.thumbs.canstockphoto.fr/labyrinthe-cercle-vecteur-illustration-clipart-vecteur_csp34490858.jpg

On peut avoir la bonne idée, mais avoir fait de mauvais choix au moment de la réaliser. Avant de chercher complètement autre chose et de changer de chemin, regarde bien si, tout en faisant la même chose, tu ne pourrais pas réussir à faire marcher ton idée.

Je change de chemin.

Quel est le contexte ? Quelles sont toutes les possibilités ? Si tu es bloqué, cherche à changer de tâche ou de stratégie. Demande-toi comment faire totalement autrement.

Img. 11 - Image non libre de droits. Source : https://smile.hr/wp-content/uploads/2018/07/Flexibility.png

Le travail et les efforts ne suffisent pas pour réussir : il faut aussi appliquer les bonnes stratégies. As-tu envisagé toutes les astuces et passé en revue toutes les manières de résoudre les problèmes qui ont été abordés en classe, dans les phases de réflexions ?

CRÉATIF

Je trouve de nouvelles idées.

Ferme les yeux et évacue toutes les pensées parasites en utilisant ton attention pour faire taire la petite voix dans ta tête. Des idées vont émerger d'elles-mêmes.

Img. 12 - Image non libre de droits. Source : https://static.qobuz.com/images/covers/25/00/8184190930025_600.jpg

La créativité c'est aussi être capable de faire des liens entre des choses qui, a priori n'en avaient pas. Cherche dans ton expérience du quotidien des situations similaires qui te permettraient de résoudre le problème sur lequel tu bloques actuellement.

GROUPE

J’utilise l’intelligence du groupe.

Si tu es bloqué, cherche de l’aide et lorsque tu as trouvé, regarde si tu peux à ton tour aider les autres à trouver par eux-mêmes. Demande-toi si les autres vont comprendre ce que tu as fait ou ce que tu dis.

Img. 13 - Image libre de droits. Sources : http://www.istockphoto.com/en/vector/ideas-exchange-gm535107545-56925818

En classe, tu as la chance de ne pas être tout seul. Si le prof n’est pas disponible, pense aux élèves : même s’ils n’ont pas toujours la réponse à ta question, ensemble vous êtes plus forts et les idées des autres peuvent t’aider à te débloquer.

21. Accumulateur. À rebours.

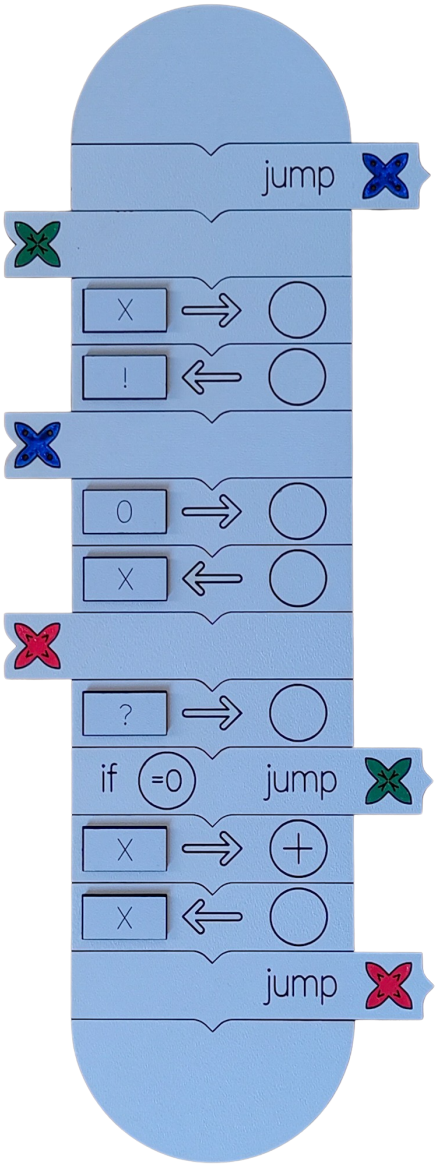
10111. Accumulateur

Pour chaque chaîne de nombres, additionne les valeurs entre elles. Dès que la fin de la chaîne est atteinte (valeur nulle) recopie le résultat, réinitialise la mémoire et recommence. Les chaînes de nombres sont séparées par une valeur nulle.
Challenge : 10 instructions.

une solution possible

```
BEGIN
   READ[+0]
   WRITE[X]
[1]:
   READ [?]
   IF  [=0] JUMP [0]
   ADD  [X]
   WRITE[X]
   JUMP [1]
[0]:
   READ [X]
   WRITE[!]
   JUMP [2]
END
```

!	Z	Y	X	?	O
					0
			0 <<		<< 0
				8 >>	>> 8
			0 >>		>>8+0
			8 <<		<< 8
				2 >>	>> 2
			8 >>		>>2+8
			10 <<		<< 10
				3 >>	>> 3
			10 >>		>>3+10
			13 <<		<< 13
				0 >>	>> 0
			13 >>		>> 13
13 <<					<< 13


jump
X
!
0
X
?
if
=0
jump
X
X
jump

Explicitation du processus de réflexion

- Lire l'énoncé.
- *Réfléchir au concept de chaîne de nombres. L'enseignant pourra faire le parallélisme avec la notion de string lorsqu'un langage de plus haut niveau sera abordé*.
- Lire un nombre
- Tester si le nombre est égal à zéro
- Additionner sa valeur à... Nous avons besoin d'une mémoire pour stocker les sommes intermédiaires calculées.
- Additionner la mémoire X au nombre dans l'accumulateur.
- Attention, la mémoire n'est peut-être pas correctement initialisée. Prévoir de mettre zéro dans la mémoire avant de commencer le programme, avant même de lire le premier nombre.
- Stocker le résultat de l'addition dans la mémoire X.
- Recommencer à lire un nombre.
- Si le nombre est égal à zéro, lire la mémoire et l'afficher sur la sortie. Puis réinitialiser la mémoire à zéro et recommencer.
- Lire le challenge.
- Le code fait actuellement bien 10 instructions de long.

#RAISONNEMENT_ALGORITHMIQUE

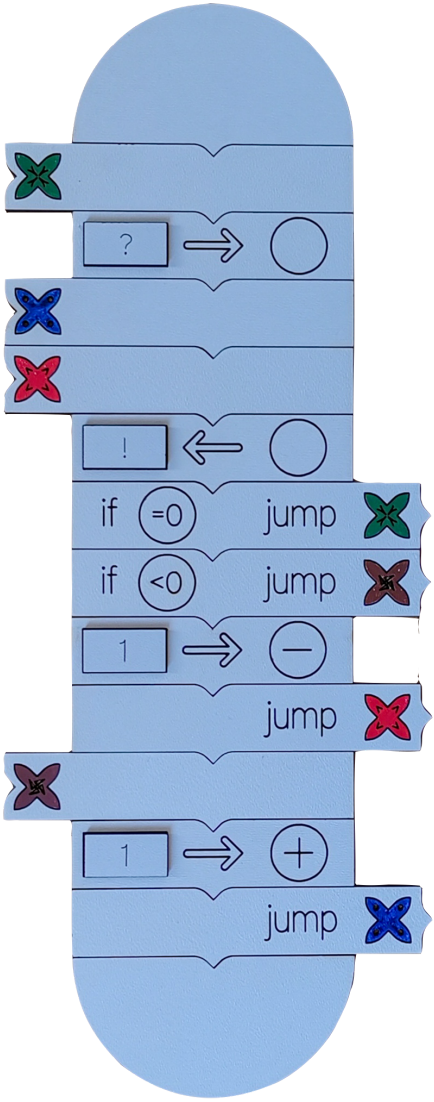
11000. À rebours.

Pour chaque entier, recopie ce nombre sur la sortie suivi de tous les nombres qui le précèdent ou qui le suivent (s'il est négatif !) jusqu'à zéro. Challenge : 8 instructions.

une solution possible

```
BEGIN
[0]:
   READ [?]
[1]:
[2]:
   WRITE[!]
   IF  [=0] JUMP [0]
   IF  [<0] JUMP [3]
   SUB [+1]
   JUMP [1]
[3]:
   ADD [+1]
   JUMP [2]
END
```

!	Z	Y	X	?	O
				3 >>	>> 3
3 <<					<< 3
					3 - 1
2 <<					<< 2
					2 - 1
1 <<					<< 1
					1 - 1
0 <<					<< 0
				-2 >>	>> -2
-2 <<					<< -2
					-2 + 1
-1 <<					<< -1
					-1 + 1
0 <<					<< 0


?
!
if =0 jump
if <0 jump
1
jump
1
jump

Explicitation du processus de réflexion

- Lire l'énoncé.
- Lire un nombre.
- Écrire le nombre sur la sortie.
- Si le nombre est égal à zéro recommencer.
- Soustraire un.
- Sauter à écrire le nombre sur la sortie.
- Déboguer le code.
- Comparer avec l'exemple donné.
- Notre code gère seulement le cas des nombres positifs.
- Relire l'énoncé.
- Ajouter un saut si le nombre est négatif avant la soustraction.
- Dans le code ajouté relatif au saut, additionner un (au lieu de soustraire).
- Sauter à écrire le nombre sur la sortie.
- Déboguer le code.
- Lire le challenge.
- Le code fait actuellement bien 8 instructions de long.

#DECOMPOSITION
#RAISONNEMENT_ALGORITHMIQUE

22. Stratégies réflexives

Voici trois stratégies susceptibles d'aider les élèves à mieux apprendre une fois l'acte de résolution terminé :
RÉFLEXIF, OPTIMUM, TRANSFERT.

Quand la tempête est passée, j'apprends à baisser les tours.

Je réfléchis aux chemins pris pour apprendre.

Je réfléchis plus pour travailler moins.

Je réfléchis aux applications futures.

CALME

Quand la tempête est passée, j'apprends à baisser les tours.

Lorsque je ne trouve pas, que je suis bloqué, je ne dois pas stresser ni paniquer. Je dois juste m'habituer à accepter le doute et l'incertitude. C'est la condition pour laisser émerger de nouvelles idées.

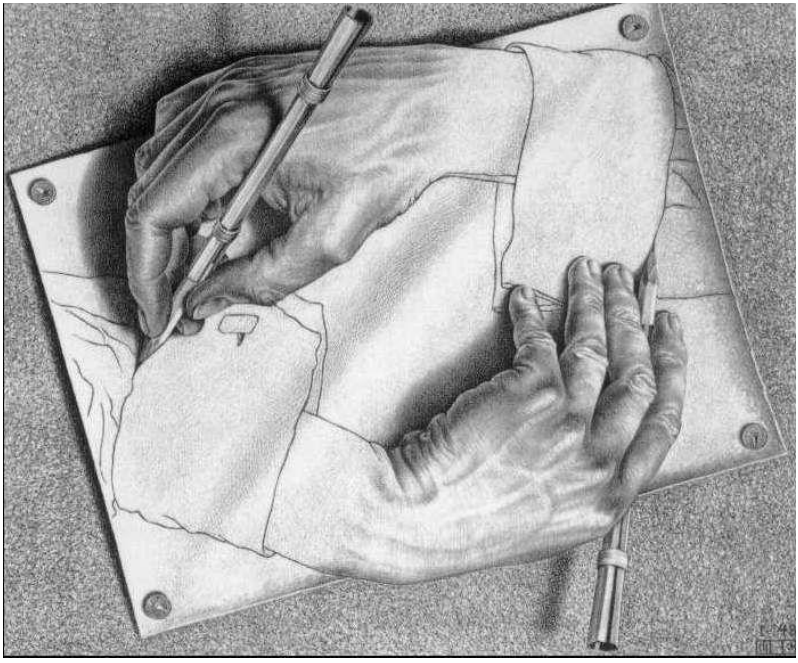
Img. 14 - Image non libre de droits. Source : https://jerem77.files.wordpress.com/2015/06/peur.png

Lorsque tu ne trouves pas, ne t'énerves pas, n'ais pas peur, calme-toi. Il est normal de ne pas savoir quoi faire ni comment faire. C'est ce qu'on apprend ici : résister aux émotions jusqu'à ce que l'on finisse par trouver quelque chose qui marche. Mais pour ça il faut d'abord maîtriser ses émotions.

RÉFLEXIF

Je réfléchis aux chemins pris pour apprendre.

Quelles sont les étapes que tu as suivies ? Juste après avoir résolu un problème, pose-toi la question de ce qui t'a permis de résoudre le problème pour t'en souvenir plus tard. Repenses-y le soir, pendant quelques minutes, puis le lendemain, puis dans trois jours, puis dans une semaine. C'est le meilleur moyen pour que tous tes apprentissages restent en toi beaucoup plus longtemps.



Img. 15 - Image non libre de droits. Source : https://www.pinterest.com.au/pin/621848661021657005/

Pose-toi des questions pour récupérer en mémoire les souvenirs de tes actions pour en prendre conscience, pour qu'ils se renforcent et puissent revenir plus rapidement une prochaine fois, dans une situation similaire. C'est la clé de l'acte d'apprendre. Plus on le fait souvent, plus ça devient facile à faire. Potvin (2011)[1] dit que << le cerveau est comme une forêt dans laquelle l'apprenant marche. Cette forêt est densément peuplée avec une végétation abondante. La marche y est donc difficile initialement. Pour se déplacer, l'apprenant doit pousser les branches en plus d'écraser l'herbe et les petits arbustes avec ses pieds. Le passage répété du marcheur crée progressivement un sentier qu'il est de plus en plus facile d'emprunter. >>. Réfléchis aux chemins pris pour apprendre en faisant non seulement des liens entre les notions apprises, mais aussi avec ce que tu connaissais avant. Repasse, en esprit, par les sentiers utilisés pour résoudre le problème, afin de renforcer tes aptitudes à le résoudre. Refait le chemin en esprit, mobilise ton intelligence pour te souvenir de tes erreurs et de tes succès : c'est aussi puissant que de refaire l'exercice. Cela renforce ta mémoire : plus tu repenses à ce que tu as appris dans la journée, la veille, il y a trois jours, il y a une semaine, mieux tu t'en souviendras. Lorsqu'il s'agit de stratégies de résolution, cela te permet de les intégrer et d'y repenser plus rapidement lorsqu'une situation similaire se présentera à nouveau.

[1] Cité p.47 dans **Masson**, S. (2020). *Activer ses neurones.* Paris: Odile Jacob.

Je réfléchis plus pour travailler moins.

Qu'est-ce que tu as vu ou entendu dans ta tête ? Comment as-tu su que c'était correct ? Dirige ton attention sur ta manière d'apprendre, pour apprendre mieux et avec moins d'efforts.

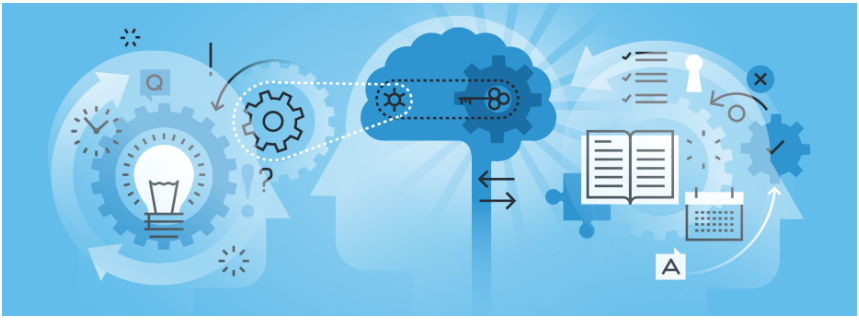
Img. 16 - Image libre de droits. Source : https://www.istockphoto.com/fr/photo/homme-daffaires-y-pose-ses-pieds-gm173606120-8186980

Chercher à t'expliquer à toi-même ce que tu as fait, comment tu l'as fait, ce que tu as compris, comment tu l'as appris, cela va te permettre de mieux structurer ces savoirs, à t'en souvenir et peut-être même à prendre conscience de liens entre les concepts qui t'auraient initialement échappés. C'est en dirigeant tes efforts sur réfléchir aux problèmes et sur comment corriger tes éventuelles erreurs que tu améliores tes stratégies et ta capacité d'apprendre.

TRANSFERT

Je réfléchis aux applications futures.

Quel principe général peux-tu dégager de cette expérience ? Comment l'appliquer ailleurs ou autrement ? Imagine comment réutiliser ce que tu apprends pour pouvoir faire des liens avec des situations futures.



Img. 17 - Image non libre de droits. Source : https://www.lean.org/LeanPost/Images/966_large.png

Ton travail, tes efforts, ta persévérance, tes stratégies d'apprentissages, n'ont qu'un seul objectif : te faire réussir plus facilement dans toutes les situations que tu rencontreras à l'école et plus tard dans la vie. Anticiper ces situations en cherchant à identifier, dans ce que tu as appris, ce qui pourra être réutilisé, ça te sert à mieux ancrer ses apprentissages, à les retrouver plus facilement en toi le jour ou tu en as besoin et surtout à réaliser plus vite ce jour-là les situations sont similaires et donc que tu sais déjà faire.

23. Comparateur.

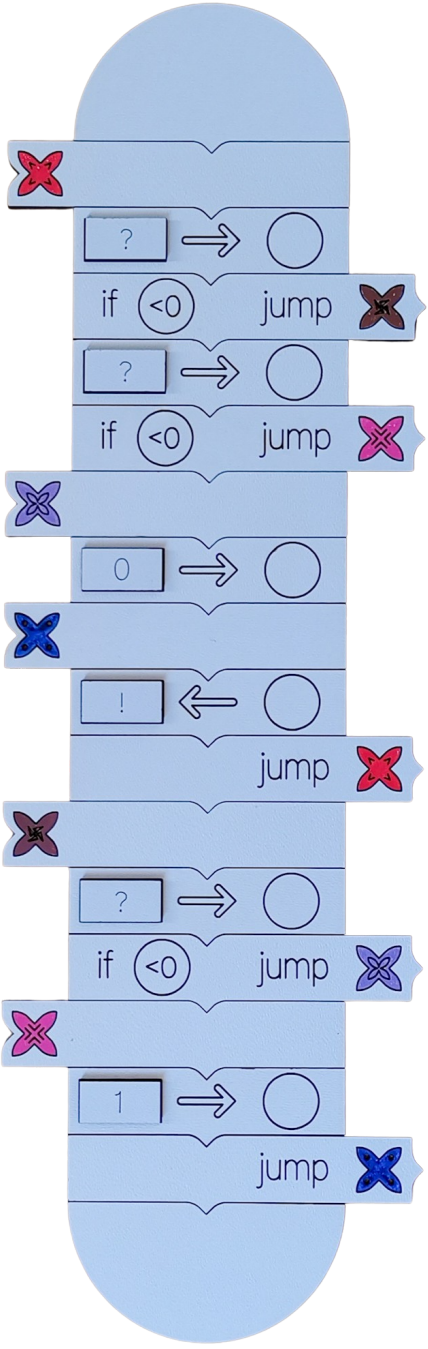
11001. Comparateur

Pour chaque paire de nombres, si les nombres sont de même signe envoie 0 sur la sortie, sinon envoie 1.
Challenge : 11 instructions.

une solution possible

```
BEGIN
[0]:
   READ [?]
   IF  [<0] JUMP [3]
   READ [?]
   IF  [<0] JUMP [4]
[1]:
   READ[+0]
[2]:
   WRITE[!]
   JUMP [0]
[3]:
   READ [?]
   IF  [<0] JUMP [1]
[4]:
   READ[+1]
   JUMP [2]
END
```

!	Z	Y	X	?	O
				5 >>	>> 5
				3 >>	>> 3
					0
0 <<					<< 0
				-2 >>	>> -2
				2 >>	>> 2
					1
1 <<					<< 1
				1 >>	>> 1
				0 >>	>> 0
					0
0 <<					<< 0


?
if <0 jump
?
if <0 jump
0
!
jump
?
if <0 jump
1
jump

Explicitation du processus de réflexion

- Lire l'énoncé.
- Lire un nombre.
- S'il est négatif, sauter vers la zone de traitement des nombres négatifs
- Lire un nombre.
- S'il est négatif, les signes sont différents : lire 1 et l'afficher sur la sortie.
- Sinon, les signes sont identiques : lire 0 et l'afficher sur la sortie, puis recommencer.
- Dans la zone de traitement des nombres négatifs, lire un nombre.
- S'il est négatif, les signes sont identiques : lire 0 et l'afficher sur la sortie.
- Sinon, les signes sont différents : lire 1 et l'afficher sur la sortie, puis recommencer.
- Déboguer le code.
- Lire le challenge.
- On peut mutualiser les lignes d'affichage de l'accumulateur sur la sortie pour économiser quelques lignes. Remplacer les lignes d'affichage sur la sortie par un saut vers le code qui fait déjà ça.
- Attention à ne pas casser le code qui fonctionne en implémentant ces simplifications.
- Conserver en mémoire une image avec l'état des variables à tout moment dans le code, pour s'assurer que les simplifications écrites ne modifient pas le comportement général.

#DECOMPOSITION
#RAISONNEMENT_ALGORITHMIQUE

24. Penser les algorithmes.

Lecture et compréhension de l’énoncé.

Pour résoudre un problème algorithmique, on commence par lire son énoncé. En première lecture, on peut facilement saturer notre mémoire de travail et ne pas comprendre le texte, ce n'est pas grave. Il ne faut pas paniquer et garder confiance en soi. Reprendre la lecture en essayant d’éliminer l’inutile et conserver les informations importantes pour identifier ce qui est demandé de faire ou de résoudre. Trier les informations fournies dans les consignes en cherchant les données de départ et les choses à produire. Cette étape est parfois nécessaire pour pouvoir être plus sélectif sur les informations pertinentes pour se représenter le travail à réaliser et les distinguer des informations qui le sont moins, qui posent juste le contexte. L’objectif est de ne conserver en mémoire de travail que les informations nécessaires pour que l’on puisse se créer une représentation mentale du problème à résoudre en écartant les informations parasites. Il faut réussir à se représenter le problème à résoudre dans son ensemble : ce que l’on sait, ce que l’on cherche. Il s’agit de faire une synthèse de ce qui est demandé afin de pouvoir émettre une hypothèse heuristique sur la stratégie à utiliser et sur la démarche qui conduira à la solution. Stratégie qu’il va falloir mentalement tester en une approche exploratoire rapide sur des données simples, pour voir si elle est efficiente ou pas. L’exploitation de la représentation mentale du problème à résoudre que l’on s’est créée en lisant l’énoncé va permettre d’explorer plusieurs possibilités en éliminant rapidement les mauvaises idées, juste en testant les situations triviales.

Parmi les hypothèses faites sur les différentes stratégies de résolution possibles, conserver celle qui nous semble être la plus simple à mettre en œuvre, même si ce n'est pas forcément la plus élégante. Attention cependant à ne pas se précipiter sur la première idée qui vient à l'esprit ni se lancer dans l'élaboration d'une réponse sans avoir pris le temps de vérifier si l'idée semble valable ou pas. Tester cette hypothèse : appliquer le plan d'action prévu sur les données mémorisées, toujours en commençant par les situations les plus simples, car on cherche encore à valider notre hypothèse heuristique de résolution. Si notre hypothèse est correcte, on a un début de solution. Continuer alors d'analyser les autres données pour résoudre la suite du problème. Si notre hypothèse bloque sur certaines données et ne permet plus d'obtenir la réponse à la question posée, revoir l'hypothèse de départ et envisager d'utiliser une stratégie moins simple, mais plus précise. Un changement d'hypothèse peut aussi avoir lieu en cours de résolution : l'hypothèse initiale peut tout à fait fonctionner pour les cas simples, mais pas pour tous les cas. Les résultats obtenus ne sont pas forcément tous à remettre en cause : il ne faut pas obligatoirement jeter les réflexions faites et les conclusions partielles auxquelles on est arrivé. Une manière de chercher de nouvelles hypothèses est de faire des liens avec ce que l'on connaît en cherchant des similitudes avec d'autres situations déjà rencontrées. Observer les ressemblances et les différences avec ce que l'on a déjà vécu par le passé peut nous aider à penser à une nouvelle manière de faire.

Lorsque l'on réussit à produire le résultat escompté dans les situations triviales, vérifier que la solution trouvée est consistante avec l'énoncé du problème. La compréhension partielle du problème va nous aider à mieux cerner ce qui est demandé et mieux comprendre ce qu'il faut faire pour pouvoir gérer les

situations non triviales. Les réflexions initiales ne sont donc pas perdues. Elles étaient nécessaires pour arriver à ce stade du raisonnement. Elles nous permettent maintenant d'affiner nos réponses, de pouvoir reprendre l'ensemble de notre approche avec une conscience plus globale de ce qu'il faut faire en prenant en considération toutes les données du problème.

Le processus peut sembler coûteux en temps, mais il n'en est rien. Chaque étape permet de se familiariser davantage avec le problème et de mieux appréhender les difficultés, les choses à faire et à ne pas faire, les informations essentielles et les informations moins cruciales. Cette expertise qui se construit en nous va nous permettre de nous approcher, mentalement et de manière itérative, vers ce qui est attendu. Ce chemin intellectuel de compréhension est nécessaire pour se construire une représentation mentale fidèle du problème à résoudre. Parfois, le problème est suffisamment simple pour qu'il n'y ait pas besoin de faire de démarche particulière : on comprend tout de suite ce qui est demandé et ce qu'il faut faire. Mais parfois, le véritable problème n'apparaît pas tout de suite et se découvre au fil du processus de lecture et de tentative de résolution.
L'établissement d'une stratégie de résolution s'effectue conjointement au processus de compréhension du problème. Si l'on ne réussit pas à trouver une stratégie de résolution satisfaisante, c'est peut-être que l'on n'a pas assez bien exploré les données du problème. Une boucle se crée alors, entre la relecture pour une meilleure compréhension du problème, la recherche d'une nouvelle hypothèse heuristique et la vérification de l'hypothèse retenue. Nous appelons cette boucle le cycle d'élaboration d'une image mentale du problème (figure 14).

Résolution du problème

La phase de résolution commence lorsque l'on met en œuvre la stratégie de résolution imaginée sur l'ensemble des données du problème en envisageant les cas plus complexes. Concrètement, nous allons manipuler les informations disponibles (données du problème) pour déplacer avec les yeux les données en faisant attention à ne pas oublier les nouveaux états du système que nous sommes en train de produire au travers de nos manipulations mentales.

À ce stade de la réflexion, il est très important de rester concentré sur la tâche : écarter tout événement extérieur perturbateur, pour ne pas altérer notre état mental et la représentation que nous nous faisons du système. Notre mémoire de travail est fragile et éphémère. Si une perturbation que nous ne réussissons pas à éviter à lieu (quelqu'un qui nous pose directement une question par exemple), ne pas hésiter à abandonner le raisonnement dans son ensemble et à recommencer depuis le début. Le nouvel état mental du système doit être fiable et dénué de toute erreur pour que nous puissions l'utiliser dans la suite du raisonnement, pour calculer les prochains états du système de manière itérative jusqu'à ce que nous atteignons la fin de notre algorithme de résolution associé à notre stratégie et à nos hypothèses. Ces opérations mentales sont nécessaires et importantes pour que nous puissions nous assurer d'avancer, dans le raisonnement, vers une solution au problème posé. De temps en temps, si l'image mentale devient floue ou imprécise, reprendre rapidement les dernières étapes pour rafraîchir notre mémoire de travail et éliminer les risques d'erreur de raisonnement dus aux imprécisions inhérentes à notre mémoire de travail.

Concrètement, réfléchir au code à écrire ou bien aux opérations à décrire pour élaborer un algorithme solution au problème posé, consiste à visualiser dans son esprit les différents états du modèle mental du système en cours de conception, puis de décrire les opérations qui vont faire évoluer cet état pour obtenir un état objectif qui est soit l'objectif final, soit une étape intermédiaire que l'on imagine nous rapprocher de l'objectif final. La représentation des états internes du modèle du système que nous programmons mentalement se fait en utilisant une image de l'algorithme en cours de construction. Cette image permet de visualiser les valeurs et leurs évolutions. Nous pouvons nous représenter les différentes opérations à effectuer comme si c'était nous qui devions physiquement réaliser ces opérations. En nous projetant à la place du système que nous sommes en train de programmer, nous mobilisons notre intelligence pratique : il suffit de faire appel au bon sens et de réfléchir à ce que nous devrions faire et pouvons faire pour obtenir ce que l'on souhaite, puis une fois que ces opérations sont conscientisées, de les décrire en langage naturel.

Souvent, une bonne stratégie de résolution consiste à découper le problème en étapes plus simples à résoudre et à se concentrer sur chacune des étapes pour trouver une solution sans se préoccuper pour le moment des autres étapes. Mais lorsque l'on ne sait plus quoi faire et que l'on est bloqué, il faut observer avec un peu de recul l'état global du système dans notre tête, vérifier que nous avons bien exploité toutes les données à disposition, que nous avons bien exploré tout ce qu'il est possible de faire avec ces données, y compris les configurations de données improbables lorsque le problème est corsé. On peut donc faire un inventaire des données déjà utilisées et traitées, des résultats de réflexion intermédiaires obtenus et des données qu'il reste à traiter pour atteindre l'objectif qui nous est demandé. Si l'on ne

sait vraiment plus quoi faire après avoir fait cet inventaire, on peut repasser en revue toutes les informations de l'énoncé du problème au lieu de se contenter de parcourir uniquement les données conservées dans notre mémoire de travail. La recherche d'une éventuelle information oubliée dans l'énoncé risque de nuire à notre représentation avancée de l'état du système et parfois, cet exercice nous obligera à reprendre notre raisonnement depuis le début pour intégrer la nouvelle information glanée dans l'énoncé. Cette gymnastique n'est pas aussi coûteuse en temps que ce que l'on pourrait croire, car refaire mentalement le chemin que nous venons juste de parcourir est assez rapide. Notre mémoire de travail va nous aider à aller bien plus vite la deuxième fois.

Si l'on est toujours bloqué dans l'étape de résolution du problème, avant d'abandonner notre hypothèse de travail, de tout jeter et de repartir sur une autre hypothèse plus complexe pour reprendre nos réflexions à partir de zéro, il convient de faire une pause. Relâcher la tension intellectuelle intérieure, calmer son esprit, laisser le cerveau se reposer quelques secondes en ne pensant à rien : une nouvelle idée pourrait alors jaillir spontanément nous permettant de nous débloquer. À ce stade du raisonnement, il est hyper important de ne pas se laisser distraire par les évènements extérieurs, ou par nos émotions intérieures : stress, doute, peur de ne pas y arriver, peur de se tromper, peur d'être en échec, autant d'émotions qui viennent perturber notre tranquillité et notre raisonnement. Toutes ces émotions doivent être mises en sourdine, pour laisser au cerveau la possibilité d'activer sa pensée créatrice et nous montrer une alternative que nous n'avions pas jusqu'alors envisagée. Mais pour que la magie opère, il faut accepter de ne pas savoir quoi faire, de ne pas avoir de réponse, d'être perdu et rempli de doute, sans sourciller, sans paniquer, sans arrêter de faire l'effort de chercher à faire des associations d'idées pour en générer de nouvelles ou bien de ne

penser à rien pour laisser émerger spontanément de nouvelles idées.

Si toujours rien ne se passe, nous pouvons passer en revue des techniques de résolution connues, vues précédemment en nous demandant comment faire autrement, en tentant résoudre le problème en lui soumettant des principes déjà vus, des approches apprises, jusqu'à ce qu'une piste de solution apparaisse et que nous puissions repartir dans une nouvelle hypothèse de résolution, avec une stratégie adaptée. Nous pouvons également, à ce stade de la réflexion, nous aider de l'intelligence du groupe et demander aux autres élèves de partager leurs idées avec nous. Une idée fait émerger d'autres idées. Parfois, le chemin qu'emprunte la solution peut être assez détourné et c'est une idée à priori anodine qui pourrait nous mettre sur la voie d'une solution possible.

Expression de la solution

Lorsque l'on a terminé le processus de réflexion, que l'on a construit une solution à proposer, il faut vérifier qu'elle est bien juste. C'est une vérification plus lente qui s'opère par rapport à la vérification rapide de l'heuristique initiale, car ici on travaille avec une conscience complète de toutes les données du problème. Si la solution envisagée ne satisfait pas toutes les données du problème, chercher une nouvelle manière possible de résoudre le problème, adapter la stratégie de résolution et reprendre nos réflexions. Nous appelons cette boucle le cycle d'élaboration d'une image mentale de la solution (figure 14).

On peut être amené à élaborer des explications plus ou moins complexes pour présenter nos idées et notre résultat à d'autres personnes. Lorsque l'on exprime notre solution, bien s’assurer que nos idées soient structurées, concises et compréhensibles par nos interlocuteurs. Enfin, lorsque tout est fini, repenser à ce qui a été réalisé. Se remémorer la stratégie qui nous a conduits à la résolution du problème posé et essayer alors d’identifier des situations similaires où cette stratégie serait à même de fonctionner pour favoriser les transferts, ou bien des situations déjà vécues dans lesquelles nous aurions déjà exploité cette stratégie de résolution. Ce faisant, nous créons des liens entre des expériences distinctes. Ces prises de conscience viennent alors enrichir notre propre expérience et nous préparent à mieux affronter des situations inédites.

Sources :
http://biaa.eu/-upload/articleno117.pdf
https://orfee.hepl.ch/handle/20.500.12162/5377

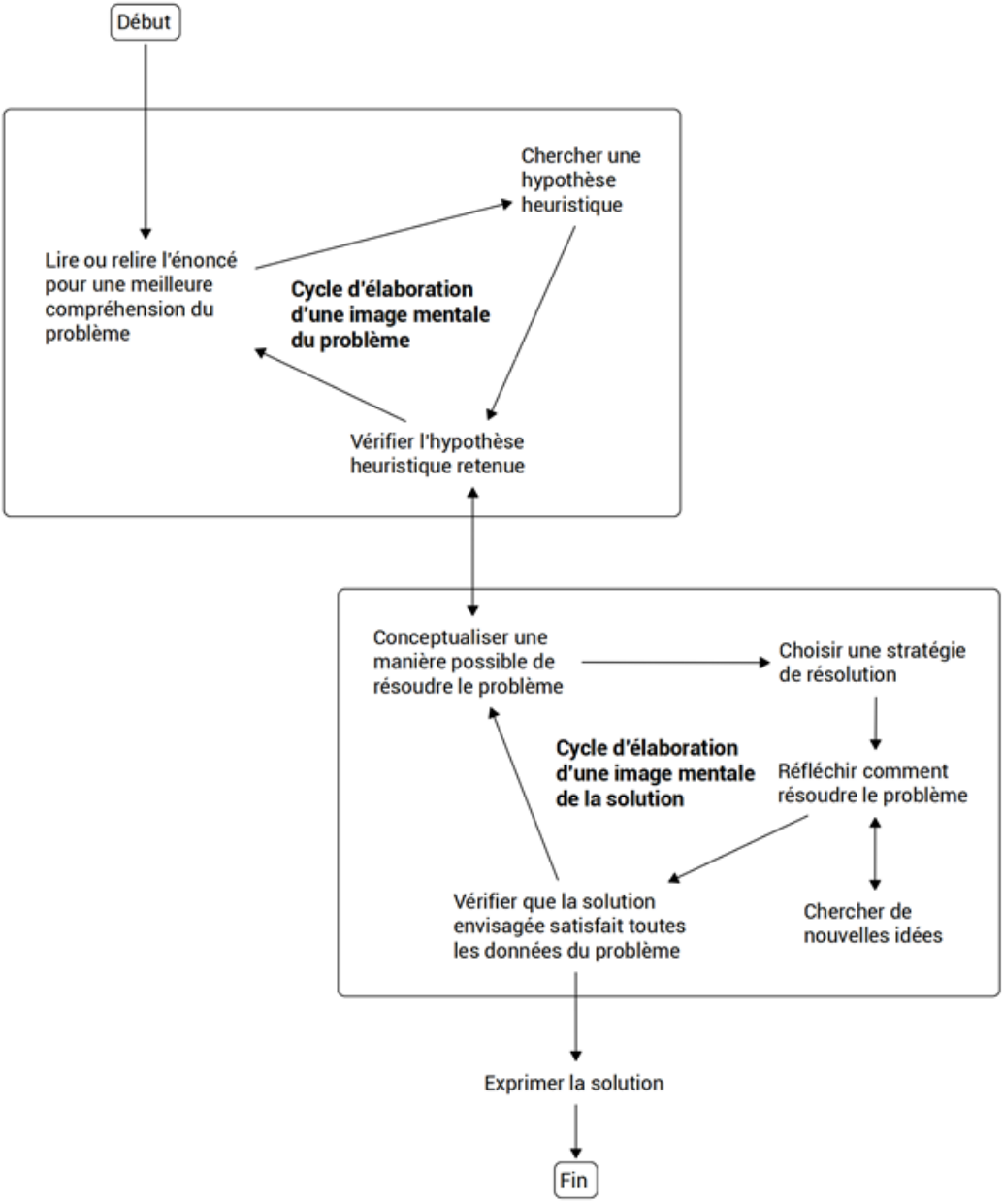


Fig. 14 - Les cycles d'élaboration d'images mentales

25. Multiplicateur.

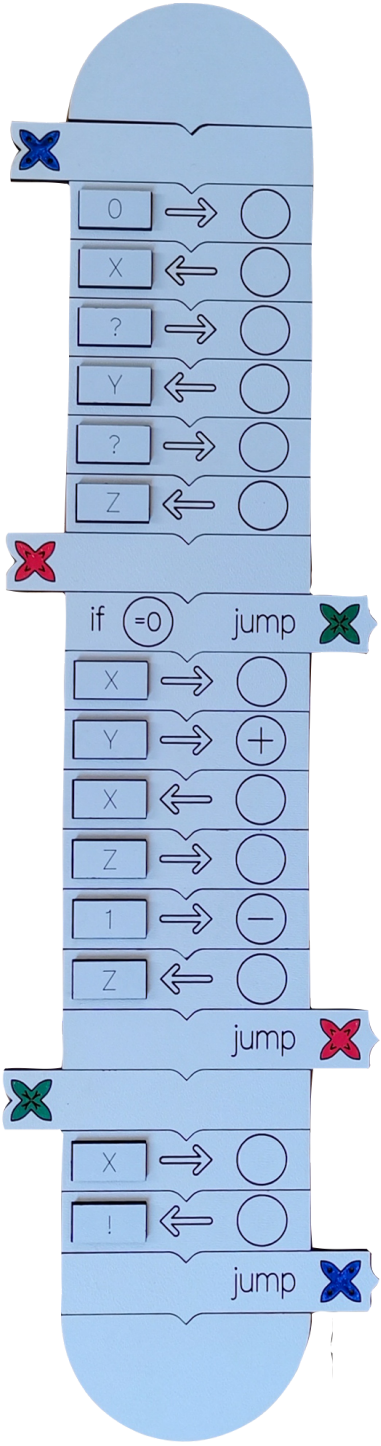
11010. Multiplicateur.

Pour chaque paire d'entiers positifs, multiplie les valeurs entre elles et recopie le résultat sur la sortie. Ne te soucie pas des nombres négatifs pour l'instant. Challenge : 17 instructions.

une solution possible

```
BEGIN
[0]:
    READ[+0]
    WRITE[X]
    READ [?]
    WRITE[Y]
    READ [?]
    WRITE[Z]
 [1]:
    IF  [=0] JUMP [2]
    READ [X]
    ADD  [Y]
    WRITE[X]
    READ [Z]
    SUB [+1]
    WRITE[Z]
    JUMP [1]
 [2]:
    READ [X]
    WRITE[!]
    JUMP [0]
 END
```

!	Z	Y	X	?	O
					0
	0 <<				<< 0
				6 >>	>> 6
			6 <<		<< 6
				4 >>	>> 4
		4 <<			<< 4
		4 >>			>> 4
					4 - 1
		3 <<			<< 3
	0 >>				>> 0
			6 >>		>>0+6
	6 <<				<< 6
		3 >>			>> 3
					3 - 1
		2 <<			<< 2
	6 >>				>> 6
			6 >>		>>6+6
	12 <<				<< 12
		2 >>			>> 2
					2 - 1
		1 <<			<< 1



	12 >>				>> 12
			6 >>		>>12+6
	18 <<				<< 18
		1 >>			>> 1
					1 - 1
		0 <<			<< 0
	18 >>				>> 18
					>>18+6
					24
	24 <<				<< 24
		0 >>			>> 0
	24 >>				>> 24
24 <<					<< 24

Explicitation du processus de réflexion

- Lire l'énoncé.
- Faire des liens avec les exercices précédents, et en particulier le *Triplicateur*.
- Il va falloir effectuer X additions de Y pour calculer le produit X*Y (ou l'inverse).
- Décrémenter X à chaque itération jusqu'à ce qu'il soit nul, pour compter le nombre d'additions effectuées.
- Lire un nombre.
- Le mémoriser dans X.
- Lire un nombre.
- Le mémoriser dans Y.
- Il va nous falloir une mémoire résultat, Z, qu'il faut initialiser à zéro avant de commencer le code.
- Si X est nul, lire Z, l'afficher sur la sortie et recommencer (aller à l'initialisation de la mémoire résultat Z).
- Sinon, lire la mémoire résultat Z, ajouter Y et écrire le résultat dans la mémoire Z.
- Lire X, soustraire 1, écrire le résultat dans X.
- Recommencer.
- Déboguer.
- Le lire challenge.
- Le code fait bien 17 instructions.

#FAIRE_DES_LIENS
#RAISONNEMENT_ALGORITHMIQUE

A propos des hashtags utilisés dans ce manuel

Concepts de la pensée computationnelle abordés avec le dispositif didactique *hp!* :

#FAIRE_DES_LIENS
#RAISONNEMENT_LOGIQUE
#RAISONNEMENT_ALGORITHMIQUE
#DECOMPOSITION
#PENSEE_CREATIVE

Concepts de la pensée computationnelle non abordés avec le dispositif didactique *hp!* (liste non exhaustive) :

#STRUCTURATION
#ABSTRACTION
#RAISONNEMENT_GEOMETRIQUES
#RECONNAISSANCE_DE_PATTERN
#REPRESENTATION_DES_DONNEES
#RECURSIVITE

26. Introduction et conclusion

Ce dernier chapitre pourrait être scindé en deux : une introduction et une conclusion. Il n'explique pas comment marche le dispositif didactique *hp!* mais se concentre plutôt sur la justification des choix didactiques qui ont conduit à la conception du dispositif. Il fait également, en guise de conclusion, un bilan des apprentissages effectués au cours de cette formation et de son utilité pour nos élèves.

Quelles sont les difficultés habituellement rencontrées par les élèves débutants en programmation ? Quelles sont les réponses apportées par le dispositif didactique ou l'approche pédagogique proposée ?

Les deux choses jugées comme étant les plus difficiles[1] dans l'apprentissage de la programmation sont de concevoir l'algorithme qui va permettre d'écrire le code pour résoudre le problème donné puis, une fois le code écrit, de le déboguer. Les trois concepts jugés les plus simples sont **la notion de variable, les tests et les boucles**. Nous avons conçu un dispositif qui ne se base que sur ces trois concepts et nous les avons encore simplifiés d'avantage. Ce faisant, nous pouvons nous concentrer sur les actions de l'élève mobilisant les compétences les plus

[1] **Lahtinen**, E. **Ala-Mutka**, K. **Järvinen**, H.-M. (2005). *A study of the difficulties of novice programmers.* Conference Paper in ACM SIGCSE Bulletin. September 2005. DOI: 10.1145/1067445.1067453

complexes tout en réduisant au maximum la charge liée à l'apprentissage d'un langage de programmation.
Les simplifications apportées concernent chacun des trois concepts : les variables typées avec l'affectation ont été remplacées par des mémoires non typées dans lesquelles il n'est possible que de lire ou d'écrire (les flèches) ; les structures de code et les différents types de boucles ont été remplacés par des sauts (jump) ; les tests avec expressions conditionnelles et gestion du cas "sinon" n'existent pas et sont remplacés par la notion de saut conditionnel (if jump), fusionnant ainsi ce concept avec le précédent. C'est ainsi que nous obtenons un langage minimaliste n'utilisant que deux notions : déplacer une information et sauter (de manière conditionnelle ou pas) vers une autre instruction que la suivante. Les efforts des élèves peuvent ainsi s'orienter sur les stratégies de résolution de problème sans trop subir d'obstacles liés à l'apprentissage d'une syntaxe exigeante ou d'un environnement de programmation complexe.

Les aspects les plus intéressants de la programmation, à savoir apprendre à penser le code, maîtriser les processus qui permettent de l'écrire et de le déboguer, sont abordés de manière collective les semaines paires, lorsqu'il n'y a pas d'activité programmée. Le choix d'alterner les séances d'expérimentation avec les séances d'analyse réflexive, permet d'allouer du temps pour la discussion et la comparaison des stratégies de chacun. Ce temps d'analyse est également utilisé pour que les élèves puissent montrer leur solution et en discuter en groupe. En plus de donner les solutions, on discute et on analyse les moyens mis en œuvre pour les trouver. C'est un espace pour que les élèves en difficulté puissent comprendre les stratégies utilisées par les autres qui auraient pu les conduire à la solution recherchée.

Si la motivation des élèves les plus en difficulté baisse, il est envisageable de les faire passer du jeu débranché au jeu sur tablette. Inconvénients : la version sur tablette est un peu plus complexe et ils ne bénéficient plus de l'aide du groupe (ils se retrouvent seuls sur leur tablette). Avantages : l'aspect ludique du jeu numérique peut éveiller leur motivation et leur permettre éventuellement de raccrocher. Une alternance tablette / activité débranchée est également possible.

Nous avons constaté que les élèves les plus en difficultés sont repérables dès les premiers instants de la formation. En effet, ils considèrent peut-être qu'ils réussiront à apprendre à programmer en utilisant des stratégies qui ont fonctionné dans d'autres disciplines scolaires et adoptent une attitude passive facilement repérable. Or, contrairement à bien d'autres disciplines, pour apprendre à programmer il ne suffit pas de lire et de comprendre le code : il faut s'entraîner à penser le code pour se construire une représentation mentale cohérente et efficiente, de manière à pouvoir développer sa pensée algorithmique. **Pour apprendre à programmer, il faut mobiliser son intellect dans l'acte de concevoir des algorithmes.** De la même manière, pour devenir intelligent, il faut entraîner son cerveau à résoudre des problèmes de plus en plus complexes. Penser les algorithmes et écrire le code est une condition nécessaire pour apprendre à programmer, mais pas suffisante. Il faut en plus savoir résoudre des problèmes complexes et donc savoir mobiliser les bonnes stratégies cognitives. C'est sur quoi nous travaillons dans cette formation.

Pour finir, voici un bilan des objectifs d'apprentissages de cette formation sous forme d'une présentation Google Slide : bit.ly/hp-BILAN

La première partie de la présentation Google Slide établit des parallèles entre les trois concepts de *hp!* et leur équivalent en Python, avec un rappel des 14 stratégies cognitives étudiées. La seconde partie retrace l'historique de la conception des premiers ordinateurs et de leur architecture et se termine par une ouverture vers le jeu "*human resource machine*" et son successeur "*7 Billion humans*". Un dernier slide tente de faire le point sur l'utilité de ces apprentissages pour tous à l'école.

Nos enfants vont être amenés à évoluer dans un monde entouré d'algorithmes. Comprendre le fonctionnement interne des processeurs, cerveaux électroniques de nos machines, c'est comprendre qu'il n'y a pas de magie dans l'intelligence artificielle ou dans les ordinateurs quantiques. Toute la puissance du code est liée aux idées algorithmiques qui les gouvernent. C'est l'usage de ces technologies en citoyen responsable qui est en jeu. Pour gérer sa vie, son quotidien, sa cité, en connaissance de cause, on ne peut pas faire l'économie d'en comprendre les rouages et les principes.

Ce cours d'introduction apporte toutes les bases utiles pour un apprentissage autonome plus poussé de l'algorithmie et de la programmation qui peut se poursuivre, au-delà de la formation, au travers des deux jeux "*human resource machine*" et "*7 Billion humans*" de la société Tomorrow Corporation. Grâce au dispositif didactique débranché, les proto-concepts sont posés et des liens pourront être tissés lors de l'apprentissage d'un prochain langage de plus haut niveau, tel que Python.

Les compétences transversales hautement réutilisables, telles que l'entraide, la compréhension des mécanismes de résolution de problème, de compréhension d'un énoncé, d'analyse de corrélations entre nouvelles et anciennes notions, de mémorisation, de créativité, alimentent toutes les discussions et réflexions qui entourent et alternent avec les séances sur les exercices proposés. Ces enseignements, à haute valeur ajoutée, constituent la raison d'être première de ce dispositif didactique débranché qui a été conçu, avant tout, pour aider les élèves à développer leurs intelligences logico-mathématique, pratique et créative : c'est-à-dire leur intelligence algorithmique.

T.00000. Photocopieur
Recopie deux valeurs de l'entrée sur la sortie.

T.00001. Inverseur
Recopie deux valeurs sur la sortie, mais en changeant l'ordre.

T.00010. Additionneur
Lis deux entiers. Calcule leur somme.
Affiche le résultat sur la sortie.

T.00011. Additionneur infini
Additionne toutes les valeurs reçues jusqu'à l'infini.
Affiche la somme sur la sortie après chaque calcul.

T.00100. Comptable
Additionne toutes les valeurs positives et négatives.
Affiche la somme sur la sortie après chaque calcul.
Arrête toi dès que le total est nul.

(T.00101. Soustracteur)
Lis deux entiers. Calcule leur différence.
Affiche le résultat sur la sortie.

(T.00110. Exterminateur de zéro infini)
Recopie tous les entiers non nuls sur la sortie.

(T.00111. Exterminateur de négatifs)
Ne recopie un entier sur la sortie que s'il est positif.

(T.01000. Exterminateur de négatifs ou nuls)
Ne recopie un entier sur la sortie que s'il est strictement positif (positif et non nul).

T.01001. Exterminateur de positifs
Ne recopie un entier sur la sortie que s'il est négatif.

(T.01010. Exterminateur de positifs infini)
Recopie tous les entiers négatifs sur la sortie.

01011. Conservateur de zéro infini
Recopie uniquement les zéros sur la sortie.

(01100. Doubleur)
Multiplie par deux la valeur entrée. Affiche le résultat sur la sortie.

01101. Triplicateur
Multiplie par trois la valeur entrée. Affiche le résultat sur la sortie.

01110. Quadriplicateur
Multiplie par quatre la valeur entrée. Affiche le résultat sur la sortie. *Challenge : maximum 2 additions.*

(01111. Octoplicateur)
Multiplie par huit la valeur entrée. Affiche le résultat sur la sortie. *Challenge : maximum 3 additions.*

10000. Décaplicateur
Multiplie par dix la valeur entrée. Affiche le résultat sur la sortie. *Challenge : maximum 4 additions.*

10001. Heptaplicateur
Multiplie par sept la valeur entrée. Affiche le résultat sur la sortie. *Challenge : utilise uniquement les pièces d'une seule boite.*

10010. Compteur
Lis un entier positif. Affiche tous les entiers plus petits jusqu'à 0 sur la sortie.

10011. Egalisateur
Pour chaque paire de nombres, s'ils sont différents ne fais rien, s'ils sont égaux recopie l'un d'entre-eux sur la sortie.
Challenge : 8 instructions.

10100. Distanciateur
Pour chaque paire de nombres, soustrait d'abord le second au premier, envoie le résultat sur la sortie puis fais l'inverse : soustrais le premier du second, envoie le résultat et recommence.
Challenge : 8 instructions.

10101. Maximisateur
Pour chaque paire de nombres, recopie le plus grand des deux sur la sortie. *Challenge : 10 instructions.*

10110. Positiveur
Pour chaque nombre, s'il est positif, recopie-le directement sur la sortie, sinon retire-lui d'abord son signe. *Challenge : 8 instructions.*

10111. Accumulateur
Pour chaque chaîne de nombres, additionne les valeurs entre elles. Dès que la fin de la chaîne est atteinte (valeur nulle) recopie le résultat, réinitialise la mémoire et recommence. Les chaînes de nombres sont séparées par une valeur nulle.
Challenge : 10 instructions.

11000. A rebours
Pour chaque valeur, recopie ce nombre sur la sortie, suivi de tous les nombres qui le précèdent s'il est positif, ou qui le suivent s'il est négatif, jusqu'à zéro. *Challenge : 8 instructions.*

11001. Comparateur
Pour chaque paire de nombres, si les nombres sont de même signe envoie 0 sur la sortie, sinon envoie 1. *Challenge : 11 instructions.*

11010. Multiplicateur
Pour chaque paire d'entiers positifs, multiplie les valeurs entre elles et recopie le résultat sur la sortie. *Challenge : 17 instructions.*

Remerciements

Je remercie Marie De Calignon, enseignante à l'École Active de Malagnou, pour m'avoir ouvert sa classe deux années consécutives et m'avoir permis de récolter toutes ces données qui ont été si précieuses pour mes recherches.

Je remercie Luc-André Fontaine, doyen de l'ESIG, pour m'avoir permis d'organiser durant deux années consécutives des cours de soutien durant lesquels j'ai pu expérimenter et mettre au point mon dispositif didactique et toute la théorie qui l'accompagne.

Je remercie mes collègues enseignants de l'ESIG, Eric Batar, Mirko Steinle, Thomas Servettaz et Clément Vogt, d'avoir accepté d'introduire ce dispositif lors des premières semaines de cours d'algorithmique et de programmation.

Je remercie enfin tous les élèves de l'École Active de Malagnou et de l'ESIG, des volées 2019-20 et 2020-21, qui ont bien voulu rentrer dans cet univers de la pensée algorithmique et relever les défis que je leur proposais en classe.