TD 1 : Notion d'algorithme

Les exercices non marqués d'un ♣ sont considérés comme acquis d'une semaine sur l'autre.

Notes pour les enseignants : Ne vous cassez pas la tête à introduire la notion d'algorithme. L'objectif est justement de les faire manipuler pour leur donner une idée informelle de ce que c'est.

Notes pour les enseignants : Les deux exercices suivants sont l'occasion de faire découvrir intuitivement par la manipulation les notions suivantes :

- Langage utilisé : quelles sont les instructions ? quelle est leur sémantique ?
- Instructions, programme ;
- État du système ;
- Représentation (mentale) de cet état et de son évolution ;
- Vérification qu'un algorithme est correct par exécution pas-à-pas. Est-ce une preuve ?
- Étendre le langage en ajoutant de nouvelles instructions.

En général, beaucoup d'étudiants auront déjà fait ces exercices. L'intérêt de la séance est d'introduire et nommer explicitement les concepts sous-jacents.

Exercice 1 (Modélisation d'un problème et notion d'algorithme)**.**
Un passeur doit faire passer de l'autre côté d'une rivière un loup, une chèvre et un chou. Pour cela, il ne peut transporter qu'un seul d'entre eux à la fois dans sa barque. On ne sait pas trop ce qui arriverait si le loup venait à rester seul en présence de la chèvre, ou si la chèvre se retrouvait seule en présence du chou.

(1) Décrire une suite d'instructions simples (un premier algorithme !) à effectuer scrupuleusement par le passeur garantissant que tout le monde traverse la rivière sain et sauf. Les instructions disponibles sont :
   - `charge(objet)` (exemple : `charge(chèvre)`)
   - `décharge(objet)`
   - `traverse()`

(2) Identifier une séquence d'instructions qui se répète et proposer une nouvelle instruction les combinant – d'un niveau d'abstraction plus élevé – permettant de simplifier l'écriture de l'algorithme précédent.

Correction :

(1)
```
charge(chèvre)
traverse()
décharge(chèvre)
traverse()
charge(loup)
traverse()
décharge(loup)
charge(chèvre)
traverse()
décharge(chèvre)
```

```
charge(chou)
traverse()
décharge(chou)
traverse()
charge(chèvre)
traverse()
décharge(chèvre)
```

Remarque : comme le loup et le chou ont des rôles symétriques, on peut trouver une autre solution en échangeant loup et chou ci-dessus.

(2) La séquence `charge(objet) traverse() décharge(objet)` est présente cinq fois dans la suite d'instructions donnée en solution à la questions précédente. En créant une nouvelle instruction `déplace(objet)` correspondant à `charge(objet) traverse() décharge(objet)` on peut écrire une nouvelle solution qui fait sept lignes au lieu de dix-sept et qui est beaucoup plus lisible et facile à exécuter pas-à-pas pour la vérifier :

```
déplace(chèvre)
traverse()
déplace(loup)
déplace(chèvre)
déplace(chou)
traverse()
déplace(chèvre)
```

Exercice 2 (Modélisation d'un problème et notion d'algorithme)**.**
Soient deux cruches de capacités respectives 5 et 7 litres. Ces cruches ne sont pas graduées. Le but de ce problème est que l'une des deux cruches contienne 4 litres. Vous avez accès à un robinet et à une évacuation d'eau, mais vous ne disposez de rien d'autre que les deux cruches.

(1) Formalisez le problème en termes de données et actions, puis donnez une suite d'actions permettant d'atteindre une solution.

(2) Peut-on faire une suite d'actions de façon que l'une des deux cruches contienne 1 litre ? 2 litres ? 3 litres ? 4 litres ? 5 litres ? 6 litres ? 7 litres ?

Correction :

(1) Données : C5, C7 (cruches de 5 et 7 litres respectivement)
Actions :
— `remplit(objet)` : remplit au maximum l'objet (au robinet)
— `vide(objet)` : vide entièrement l'objet (dans l'évacuation)
— `transvase(objet1, objet2)` : verse de l'eau de `objet1` dans `objet2` jusqu'à ce que soit `objet2` soit plein, soit `objet1` soit vide.
Suite d'instructions permettant d'obtenir 4 litres :

```
remplit(C7)
transvase(C7,C5)
vide(C5)
transvase(C7,C5)
remplit(C7)
transvase(C7,C5)
```

(2) Oui (se prouve en décrivant la suite d'actions). Il est possible de faire
1 L : 8 actions

2 L : 2 actions
3 L : 4 actions
4 L : 6 actions
5 L : 1 action
6 L : 10 actions
7 L : 1 action.

Notes pour les enseignants : Représenter au tableau l'état du système après chaque action (ou demander aux élèves de le faire sur leur feuille) afin de les familiariser avec la notion d'exécution pas-à-pas.
Si vous avez du temps et un bon groupe, vous pouvez en profiter pour leur demander comment faire pour prouver que leur méthode est la plus efficace (par exemple prouver qu'on ne peut pas obtenir 4 litres en moins de 6 actions). (Peut se faire en dessinant le graphe (l'automate en fait) de toutes les possibilités en partant de l'état (0, 0) qui représente les deux cruches vides).

Exercice 3.
Écrire un algorithme qui, étant donné un nombre entier N, décide si N est ou non le carré d'un autre entier.

Par exemple, si l'entrée est 16, la réponse est oui. Si l'entrée est 42, la réponse est non.

Votre algorithme doit être assez simple pour que vous puissiez l'appliquer vous-même uniquement à l'aide d'un papier et d'un crayon pour poser les opérations (pas de calculatrice).

Notes pour les enseignants : Il y a toujours des élèves qui veulent utiliser `racine(N)`. Demandez-leur s'ils sont vraiment capables de calculer à la main la racine de n'importe quel entier (par exemple 235).

Correction : Entrée : un entier N
Sortie : un booléen (vrai ou faux)
Algorithme :
Pour M nombre entier allant de 1 à N faire :
Si $M * M$ est égal à N alors répondre vrai (et s'arrêter).
Répondre faux (si on n'a pas déjà répondu).
Autre solution :
M ← 1
Tant que M*M < N faire M ← M + 1
Si M*M est égal à N répondre vrai, sinon répondre faux.

Exercice 4 (Nombre mystère)**.**
On a fabriqué un robot « intelligent » capable de
- poser des questions dont la réponse est oui ou non ;
- effectuer des opérations de base sur les nombres (addition, soustraction, multiplication, division)

Voici un exemple d'algorithme pour le robot :

```
Reponse = Demande("Êtes-vous né avant le 1er janvier 2000 ?")
Si Réponse = Oui alors
    Dire("Vous avez plus de 22 ans")
Sinon
    Dire("Vous avez 22 ans ou moins")
```

(1) Décrivez un algorithme qui permet au robot de deviner un nombre choisi entre 0 et 100 par l'utilisateur.

(2) Décrivez un algorithme qui permet au robot de donner l'âge exact de l'utilisateur.

(3) Quelles difficultés avez-vous rencontrées pour écrire les algorithmes ?

Notes pour les enseignants : La rédaction de l'algorithme pose des questions de syntaxe (comment écrire la boucle, comment afficher les nombres, comment récupérer le nombre entré par l'utilisateur, comment réutiliser l'algorithme précédent). C'est intéressant de voir si les étudiants se posent ces questions et comment on peut y répondre. Comme pour l'instant on n'a pas donné de syntaxe, on peut choisir la variante qu'on veut, tant qu'elle est logique. En particulier on peut laisser les étudiants exprimer la boucle de la façon qui est naturelle pour eux, à condition que l'idée de boucle transparaisse bien dans leur solution.

Correction :

(1)

```
m = 0
M = 100
N = 50
Faire
    Réponse = Demande("Votre nombre est-il inférieur ou égal à ", N)
    Si Réponse == Oui alors M = N
    Sinon m = N+1
    N = (m + M) / 2      # division entière
Tant que m différent de M

Dire("Votre nombre est", m)
```

(2)

```
m = 1890  # date assez ancienne pour une annee de naissance
M = 2022  # annee actuelle
N = (m + M) / 2
Faire
    Réponse = Demande("Votre année de naissance est-elle au plus ", N)
    Si Réponse == Oui alors M = N
    Sinon m = N + 1
    N = (m + M) / 2      # division entière
Tant que m différent de M

Réponse = Demande("Votre anniversaire a-t-il eu lieu cette année ?")
Si Réponse == oui Dire("Vous avez ", M - m, "ans")
Sinon Dire("Vous avez ", M - m - 1, "ans")
```

(3) Syntaxe : comment exprimer les boucles ? Les variables ? Comment afficher un nombre ?

Efficacité : tous les algorithmes sont-ils équivalents ? On peut aborder ces différentes questions en fonction des réponses des étudiants.

Exercice 5.

Notes pour les enseignants : Les élèves n'ont pas du tout entendu parler de logique ou de booléens en amphi pour l'instant. Cet exo est à introduire en douceur, ou à réserver à ceux qui ont déjà fait de l'informatique (pour les occuper s'ils s'ennuient).

Cet exercice porte sur les notions de base de la logique (booléens).

(1) Construire les tables de vérité correspondant aux opérations booléennes suivantes :
— `op1(a, b) : (a ET (NON b)) OU ((NON a) ET b)`
— `op2(a, b) : (a OU b) ET ((NON a) OU (NON b))`.

(2) Que constatez-vous ?

(3) `op1(a, b)` est en fait une opération très courante en informatique, elle a même un nom. Savez-vous comment on l'appelle ?

(4) Écrire une opération `op3(a, b)` dont le résultat est vrai si et seulement si le résultat de `op1(a, b)` est faux. Utiliser au plus 5 mots `OU`, `NON` et `ET` dans l'écriture de `op3(a, b)`.

(5) Expliquez pourquoi cette opération permet de déterminer si deux mots booléens sont identiques.

Correction : (1) Voir (2).

(2) Les deux opérations ont la même table de vérité :

a	b	op(a, b)
vrai	vrai	faux
vrai	faux	vrai
faux	vrai	vrai
faux	faux	faux

(3) On l'appelle `XOR` ce qui est l'abréviation en anglais de `OU exclusif` (exclusive or).

(4) S'il n'y avait pas la limitation sur le nombre de mots, on pourrait simplement mettre un `NON` en facteur de l'expression de `op1` ou de `op2`, mais cela donne 6 mots. En 5 mots : `op3(a, b) : (a ET b) OU ((NON a) ET (NON b))`

Exercice ♣ 6 (Notion de test de programme)**.**
Une autre personne s'est occupée de coder un programme `pointDeChute` qui calcule l'abscisse à laquelle retombe un projectile lancé en $x = 0$ avec une vitesse v suivant un angle α (en degrés). Vous voulez tester si les résultats de ce programme sont cohérents.

(1) Spécifier le programme sans écrire l'algorithme : que va-t-il prendre en entrée ? renvoyer en sortie ?

(2) Donner une série de tests dont vous connaissez le résultat sans calculs : situations extrêmes (par ex : tirer verticalement), symétries, ...

Correction :

(1) Deux entrées : vitesse et angle. Sortie : l'abscisse de retombée.

(2) Pour toute vitesse v, `pointDeChute(v, 0) = 0` et `pointDeChute(v, 90) = 0`
Pour tout angle α, `pointDeChute(0, α) = 0`
Pour toute vitesse v et tout angle α,
`pointDeChute(v, 180 - α) = -pointDeChute(v, α)`.