A la rencontre de quelques structures

Activité 1 — Exemples de structures "séquentielles"

A partir d'un sujet du brevet *(Nouvelle Calédonie mars 2009)*

Programme de calcul :
- choisir un nombre de départ,
- ajouter 1,
- calculer le carré du résultat obtenu,
- lui soustraire le carré du nombre de départ,
- écrire le résultat final.

1. ***a)*** Vérifier que lorsque le nombre de départ est 1, on obtient 3 au résultat final.
b) Lorsque le nombre de départ est 2, quel résultat final obtient-on?
c) Le nombre de départ étant x, exprimer le résultat final en fonction de x.

2. On considère l'expression $P = (x+1)^2 - x^2$. Développer puis réduire l'expression P.

3. Quel nombre de départ doit-on choisir pour obtenir un résultat final égal à 15 ?

→ Répondre aux questions 1.*a* et 1.*b* en exécutant ce programme de calcul à la main.

→ Pour répondre à la question 1.*c* :
- Ecrire l'algorithme correspondant à ce "programme de calcul".
- Programmer cet algorithme sur votre calculatrice et sur Algobox.
- Tester votre programme avec différents nombres de départ.

→ Terminer l'exercice.

→ Si on choisit des entiers positifs comme nombres de départ, que peut-on dire des résultats obtenus par ce programme de calcul ?

→ Modifier le programme de calcul précédant afin d'obtenir l'expression $P = (x+1)^2 + 2x + 3$.
- Vérifier que lorsque le nombre de départ est 1, on obtient 9 au résultat final.
- Lorsque le nombre de départ est 2, quel résultat final obtient-on?
- Quel nombre de départ doit-on choisir pour obtenir un résultat final égal à 25 ?
- Que peut-on dire des résultats obtenus par ce programme de calcul ?

Prolongement possible

Que se passe-t-il si l'on ajoute 1 au produit de 4 entiers consécutifs ?

Exercice

Que fait chacun de ces deux algorithmes ?

```
Variables
        a, b, c
Entrées
        Saisir a, b
Traitement
        c prend la valeur a
        a prend la valeur b
        b prend la valeur c
Sorties
        Afficher a, b
```

```
Variables
        a, b
Entrées
        Saisir a, b
Traitement
        a prend la valeur a–b
        b prend la valeur a+b
        a prend la valeur b–a
Sorties
        Afficher a, b
```

Le problème

On forme une suite de nombres en procédant ainsi :
- Choisir a
- Choisir b
- Calculer le nombre suivant c, en faisant la différence entre le dernier nombre obtenu et le précédent.
- Recommencer en remplaçant a par b et b par c.
- Afficher le $30^{ème}$ nombre de cette suite.

1. Ecrire l'algorithme correspondant à ce "programme de calcul".
Programmer cet algorithme sur votre calculatrice et sur Algobox.

2. Observer le nombre obtenu en sortie pour différentes valeurs de a et b.

3. Quel affichage devrait-on avoir pour a=12 et b=15 ? Vérifier avec le programme.
Émettre une conjecture quant à la valeur du $30^{ème}$ nombre de la suite.

4. Modifier alors le programme pour qu'il affiche les 30 premiers nombres.
Cela permet-il de justifier la conjecture ?

5. Démontrer la conjecture observée.

Fonctionnement d'une instruction POUR

*Une **structure répétitive** permet d'exécuter plusieurs fois de suite le même traitement c'est à dire la même série d'instructions.*
*La plus simple à comprendre est la « **structure itérative** » qui consiste à répéter un certain traitement un nombre N de fois fixé à l'avance.*
Dans la plupart des langages (y compris ceux des calculatrices) on utilise un compteur I (c'est une variable) pour contrôler le nombre (entier) de tours. À chaque « tour » le compteur I augmente automatiquement de 1.

Nous formulerons cela :

Pour I de 1 jusqu'à N faire
| *Traitement à répéter*
└ **Fin pour**

Prolongement possible

- Modifier le programme précédant afin d'obtenir les 30 premiers termes de la suite de Fibonnacci.
- Modifier le programme précédant afin d'obtenir aussi le quotient de deux termes consécutifs.

Exercice

Que fait cet algorithme ?

```
Variables
        n, p, i
Entrées
        Saisir n
Traitement
        p prend la valeur n
        Pour i de 1 jusqu'à 3 faire
                p prend la valeur p*(n+i)
                Fin pour
        p prend la valeur p+1
Sorties
        Afficher p
```

Le problème

> On lance un dé cubique parfait autant de fois qu'il le faut pour obtenir un six.
> Combien de fois faut-il lancer le dé en moyenne pour atteindre le premier 6 ?

1. A l'aide de la calculatrice (ou d'un logiciel) réaliser quelques simulations de l'expérience consistant à lancer un dé autant de fois qu'il le faut pour obtenir un six.
Noter à chaque fois le nombre de lancers nécessaires.

2. Ecrire l'algorithme correspondant à cette simulation, qui affiche à la sortie le nombre de lancers nécessaires à l'obtention du premier 6.
Programmer cet algorithme sur votre calculatrice et sur Algobox.

3. Observer le nombre obtenu en sortie pour différentes simulations.

4. Modifier l'algorithme précédant afin de réaliser non plus une, mais plusieurs expériences. Le nombre d'expériences pouvant être choisi par l'utilisateur au moment de l'exécution de l'algorithme.
A la fin, faire afficher le nombre moyen de lancers nécessaires pour atteindre le premier 6.

5. D'après vous, combien de fois faut-il lancer le dé en moyenne pour atteindre le premier 6 ?

Fonctionnement d'une instruction TANT QUE

Une autre structure répétitive est celle qui consiste à répéter traitement tant qu'une certaine condition reste valide (on ne souhaite évidemment pas que la répétition se fasse une infinité de fois).

Nous formulerons cette structure ainsi :

Tant que *condition* **faire**
  Traitement à répéter
Fin tant que

Le nombre de répétitions de la boucle dépendra de la condition.

Lorsque l'algorithme rencontre l'instruction TANT QUE, il teste la condition.
- *Si la condition est vraie, il entre dans la boucle en exécutant son contenu et recommence tant que la condition reste vraie.*
- *Si la condition n'est pas vérifiée, il ne rentre pas dans la boucle et saute "traitement à répéter" qui ne sera pas exécuté.*

Pour que l'algorithme soit correct, il est nécessaire que la condition cesse d'être vérifiée au bout d'un nombre fini de répétitions. Sinon le Traitement sera exécuté indéfiniment et l'utilisateur sera obligé d'arrêter le programme. On dit que le programme "boucle indéfiniment".
Il est parfois difficile d'avoir l'assurance du fait que le programme ne bouclera jamais indéfiniment.

Exercice : *Calcul de* 10 !

1. Contrôler le bon fonctionnement des algorithmes suivants.

2. Proposer, si nécessaire, une correction pour chacun d'entre eux.

```
début
  Donner à res la valeur 1
  Donner à i la valeur 1
  tant que i ≤ 10 faire
    Donner à res la valeur res*i
  fin
  Afficher res
fin
```

```
début
  Donner à res la valeur 1
  tant que i ≤ 10 faire
    Donner à res la valeur res*i
    Donner à i la valeur i + 1
  fin
  Afficher res
fin
```

```
début
    Donner à res la valeur 1
    Donner à i la valeur 10
    tant que i > 1 faire
        Donner à i la valeur i – 1
        Donner à res la valeur res*i
    fin
    Afficher res
fin
```

```
début
    Donner à res la valeur 1
    tant que i ≤ 10 faire
        Donner à i la valeur 1
        Donner à res la valeur res*i
        Donner à i la valeur i + 1
    fin
    Afficher res
fin
```

Activité 4 — Exemples de structure "alternative"

On considère l'algorithme :

Entrée : Trois couples de nombres réels (xA, yA), (xB, yB), (xC, yC).

Début : Affecter à m_1 la valeur $\dfrac{yB - yA}{xB - xA}$

Affecter à m_2 la valeur $\dfrac{yC - yA}{xC - xA}$

Si $m_1 = m_2$ **alors**
    Afficher "oui"
Sinon
    Afficher "non"

Fin

1. Quelle sera la sortie de cet algorithme avec les entrées (0; 0), (2;3) ; (4;6) ?

2. Quelle sera la sortie de cet algorithme avec les entrées (0; 0), (2;3) ; (4;7) ?

3. Décrire de façon générale le rôle de cet algorithme.

4. Traduire l'algorithme pour une machine.

5. Avec quelles valeurs pour les entrées l'algorithme donnera-t-il une erreur ?

6. Compléter l'algorithme pour tenir compte de toutes les entrées possibles.
On utilisera des instructions SI pour tester les cas qui posent problème dans la version précédente.

7. Comparer les sorties de l'algorithme ci-dessous avec les sorties obtenues par le programme précédemment complété puis expliquer le fonctionnement de ce nouveau programme.

Entrée : Trois couples de nombres réels (xA, yA), (xB, yB), (xC, yC).

Début : **Si** $(yB - yA)(xC - xA) = (yC - yA)(xB - xA)$ **alors**
    Afficher "oui"
Sinon
    Afficher "non"

Fin

Fonctionnement d'une instruction SI ALORS SINON:

Avec une "structure alternative", selon qu'une certaine condition est vérifiée ou non, on fera un certain traitement ou un autre. Par "traitement", on entend une ou plusieurs instructions en séquence.

On a choisi de traduire la structure alternative par l'instruction :

Si *condition* **alors**
  Traitement si condition vraie
Sinon
  Traitement si condition fausse
Fin si-alors-sinon

Pour ce qui concerne la "condition", c'est une expression logique (simple ou complexe) prenant l'une des deux valeurs VRAI ou FAUX.

Il se peut que, si la condition n'est pas vérifiée, il n'y ait pas de traitement à effectuer.
La série d'instruction du "Traitement si condition fausse" est alors vide.

Dans ces conditions, on écrira l'instruction :

Si *condition* **alors**
  Traitement si condition vraie
Fin si-alors

Prolongement possible : *le jeu du lièvre et de la tortue.*

Règle du jeu :
On lance un dé.
- Si le dé tombe sur 6, le lièvre gagne la partie.
- Sinon la tortue avance d'une case.
  La tortue doit franchir 4 cases pour gagner la partie.

Question : le jeu est-il à l'avantage du lièvre ou de la tortue ?

Exercice

Que fait cet algorithme ?

```
Variables
    N nombre choisi par l'utilisateur
Initialisation
    S, un nombre entier au hasard entre 10 et 100
    essai prend la valeur 1
Traitement
    Tant que essai est inférieur ou égal à 6
        Saisir N
        Si N est supérieur à S alors
            Affiche « c'est moins »

        Si N est inférieur à S
            Affiche « c'est plus »

        Si n=S alors
            Affiche « gagné »
            fin de programme

        essai prend la valeur essai+1
Sortie
    Affiche « perdu ».
```

A l'ancienne

```
VARIABLES
    a EST_DU_TYPE NOMBRE
    b EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE a
    LIRE b
    TANT_QUE (b>=a) FAIRE
        DEBUT_TANT_QUE
        b PREND_LA_VALEUR b-a
        FIN_TANT_QUE
    AFFICHER "Le résultat cherché est "
    AFFICHER b
FIN_ALGORITHME
```

Interpréter l'algorithme ci-contre.
Quel autre nombre serait-il bon d'obtenir en sortie ?
Modifier l'algorithme en conséquence.

En géométrie

Interpréter chacun des algorithmes ci-dessous.
Compléter ces algorithmes afin de placer chacun des points dans un repère et de tracer un objet mathématique particulier.

```
VARIABLES
    x EST_DU_TYPE NOMBRE
    y EST_DU_TYPE NOMBRE
    x2 EST_DU_TYPE NOMBRE
    y2 EST_DU_TYPE NOMBRE
    x0 EST_DU_TYPE NOMBRE
    y0 EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    x0 PREND_LA_VALEUR 3
    y0 PREND_LA_VALEUR -2
    AFFICHER "Entrez les coordonnées du point M"
    LIRE x
    LIRE y
    AFFICHER "les coordonnées de M' sont :"
    x2 PREND_LA_VALEUR 2*x0-x
    y2 PREND_LA_VALEUR 2*y0-y
    AFFICHER "M'("
    AFFICHER x2
    AFFICHER " ; "
    AFFICHER y2
    AFFICHER ")"
FIN_ALGORITHME
```

```
VARIABLES
    x EST_DU_TYPE NOMBRE
    y EST_DU_TYPE NOMBRE
    x2 EST_DU_TYPE NOMBRE
    y2 EST_DU_TYPE NOMBRE
    m EST_DU_TYPE NOMBRE
    p EST_DU_TYPE NOMBRE
    xH EST_DU_TYPE NOMBRE
    yH EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    m PREND_LA_VALEUR 1
    p PREND_LA_VALEUR 2
    AFFICHER "Entrez les coordonnées du point M"
    LIRE x
    LIRE y
    xH PREND_LA_VALEUR (x+m*y-m*p)/(1+m*m)
    yH PREND_LA_VALEUR m*xH+p
    x2 PREND_LA_VALEUR 2*xH-x
    y2 PREND_LA_VALEUR 2*yH-y
    AFFICHER "les coordonnées de M' sont :"
    AFFICHER "M'("
    AFFICHER x2
    AFFICHER " ; "
    AFFICHER y2
    AFFICHER ")"
FIN_ALGORITHME
```

A partir d'un sujet du BAC L *(Réunion 2008)*

Dans un lycée, un code d'accès à la photocopieuse est attribué à chaque professeur.

Ce code est un nombre à quatre chiffres choisis dans la liste {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, chaque chiffre pouvant être répété à l'intérieur d'un même code. Par exemple 0027 et 5855 sont des codes possibles.

1. Combien de codes peut-on ainsi former ?

2. Ce code permet aussi de définir un identifiant pour l'accès au réseau informatique. L'identifiant est constitué du code à quatre chiffres suivi d'une clé calculée à l'aide de l'algorithme suivant :

Entrée :	N est le code à quatre chiffres.
Initialisation :	Affecter à P la valeur de N ; Affecter à S la valeur 0 ; Affecter à K la valeur 1.
Traitement :	Tant que $K \leq 4$: Affecter à U le chiffre des unités de P ; Affecter à K la valeur $K + 1$; Affecter à S la valeur $S + K \times U$; Affecter à P la valeur $(P - U) / 10$; Affecter à R le reste dans la division euclidienne de S par 7 ; Affecter à C la valeur $7 - R$.
Sortie « la clé » :	Afficher C.

a) Faire fonctionner l'algorithme avec N = 2 282 et vérifier que la clé qui lui correspond est 3.
On prendra soin de faire apparaître les différentes étapes du déroulement de l'algorithme (on pourra par exemple faire un tableau.).

b) Un professeur s'identifie sur le réseau informatique en entrant le code 4 732 suivi de la clé 7.
L'accès au réseau lui est refusé. Le professeur est sûr des trois derniers chiffres du code et de la clé, l'erreur porte sur le premier chiffre du code. Quel est ce premier chiffre ?

Activité 6 — Avec un tableur

Le nombre de KAPREKAR

> Soit un nombre de 4 chiffres quelconque, par exemple 1946.
> Rangeons ses chiffres dans l'ordre décroissant, on obtient le nombre GRAND 9641
> Rangeons ses chiffres dans l'ordre croissant, on obtient le nombre PETIT 1469.
> La différence entre ces deux nombres est 8172.
>
> En répétant la même opération sur le dernier nombre obtenu, on parvient toujours au même nombre, appelé nombre de Kaprekar !

1. A l'aide d'un tableur :
- ***a)*** Faire afficher les chiffres du nombre choisi.
- ***b)*** Faire calculer les deux nombres GRAND et PETIT ainsi que leur différence.
- ***c)*** Conjecturer le résultat obtenu au bout d'un grand nombre d'itérations.

2. Que ce passe-t-il avec des nombres de 3 chiffres ?

3. Et pour des nombres de 2 chiffres ?

Quelques instructions utiles

- **= MOD(A ; B)** renvoie le reste de la division de A par B.
- **= GRANDE.VALEUR(plage ; k)** renvoie le $k^{ième}$ plus grand nombre présent dans la plage.
- Les fonctions **MAX** et **MIN** sont disponibles.