Un exemple d'algorithme : le problème des anniversaires.

*Jean-Marie Thomassin
jeanmarie.thomassin@laposte.net*

Résumé : Présentation d'un algorithme écrit dans le langage Algobox ayant pour but de construire aléatoirement une classe de 36 élèves puis de déterminer si, dans cette classe, plusieurs élèves fêteront leur anniversaire le même jour.

Mots clés : Anniversaire ; algorithme ; Algobox ; classement ; dénombrement ; simulation ; tri ; tableau.

1. Le problème abordé :

Dans un précédent numéro de la Feuille de Vigne, notre collègue Michel Plathey a proposé une étude approfondie du problème des anniversaires. Rappelons qu'il s'agissait d'étudier si, dans une classe de 36 élèves, le fait que deux d'entre eux aient le même jour anniversaire est fréquent ou pas. Après avoir abordé le premier problème des « doubles anniversaires », il étudiait aussi celui des « triples anniversaires », celui des « doubles doubles-anniversaires »... Seule la recherche de l'éventualité d'un « double anniversaire » sera abordée dans ce texte.

L'algorithme proposé ci-dessous en est une illustration simplifiée. On considère une classe de 36 élèves ; on adopte la règle suivante : le jour précis de son anniversaire (dimanche, jour de fête ou de vacance compris), chaque élève amène un gros gâteau que l'on se partagera. La question est alors la suivante : combien de jours de l'année, y aura-t-il du gâteau à manger ?

Pour répondre à cette question, on considère une année ordinaire de 365 jours ; on peut raisonnablement estimer que la répartition des jours anniversaires est uniforme dans l'année ; ainsi, pour chaque élève, le jour anniversaire sera représenté par un nombre entier pris au hasard entre 1 et 365.

Dans ce qui suit, on propose une description des différentes étapes de la construction de l'algorithme proposé ; celui-ci met en œuvre plusieurs notions élémentaires d'algorithmique et peut donc être proposé à des élèves, soit, en une seule fois, comme illustration d'un enchaînement de plusieurs outils, soit en plusieurs étapes construites

progressivement ; on peut imaginer aussi des travaux de groupes, chacun devant traiter une partie du programme avec, ensuite, une partie de mise en commun (et donc la définition d'un « cahier des charges » à respecter...).

2. Le langage :

Le langage choisi est Algobox, logiciel gratuit et couramment utilisé dans les classes ; la version utilisée est numérotée : 0.7 ; c'était la plus récente disponible sur Internet lorsque Michel Plathey écrivait son article. Par rapport à la version initiale, des fonctions aléatoires plus sophistiquées et une fonction donnant le minimum dans une liste ont été introduites ; en complément, on peut éventuellement proposer à des élèves motivés de les recréer à partir d'outils plus rustiques.

3. Construction d'une classe :

Pour chaque élève, il nous suffit d'en connaître la date anniversaire, c'est-à-dire le quantième de ce jour dans l'année. Une classe sera donc définie par la donnée des numéros de ces 36 jours de l'année ; ce sera donc une liste de 36 nombres entiers choisis au hasard, avec répétitions possibles, dans l'ensemble des nombres entiers compris entre 1 et 365. Mathématiquement, à partir de la classe de 1^{ère}, on peut utiliser la notion de suite. Avec un tableur ou certains langages informatiques, on utilisera un tableau d'une ligne et de 36 colonnes ou d'une colonne et de 36 lignes. Plus simplement, on peut considérer qu'il s'agit de définir une fonction de l'ensemble des entiers allant de 1 à 36 dans l'ensemble des quantième de l'année ; il s'agit donc d'une liste (ordonnée) de nombres que l'on numérote. Informatiquement, on utilise donc la notion de liste ; l'objet à construire, appelé dans cet algorithme « classe » est déclaré à la ligne 3 dans la catégorie « liste » et, pour désigner chaque élément de cette liste, on utilise la notation fonctionnelle. On obtient ainsi :

Déclarations à faire :

```
1 VARIABLES  
2 I EST_DU_TYPE NOMBRE  
3 Classe EST_DU_TYPE LISTE
```

Exemples d'utilisations :

Classe[5] désigne le jour de naissance de l'élève n°5.
Classe[I] désigne le jour de naissance de l'élève n°I.

Pour construire cette liste, il faudra donc procéder successivement à 36 tirages au sort dans l'ensemble des entiers de 1 à 365. La version du langage Algobox utilisée possède une fonction permettant de réaliser simplement ce tirage. En effet, la fonction : ALGOBOX_ALEA_ENT(n,p) renvoie un entier choisi au hasard entre n et p, bornes comprises. On utilise donc la boucle suivante :

```
19 POUR I ALLANT_DE 1 A 36
20 DEBUT_POUR
21 Classe[I] PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,365)
22 FIN_POUR
```

On peut proposer éventuellement aux élèves les plus motivés de reconstruire cette fonction à partir d'une procédure élémentaire donnant un nombre réel aléatoire choisi dans l'intervalle [0 ; 1[.

Dans une première étape de construction de l'algorithme, à titre de vérification, on fera afficher la classe ainsi construite, on pourra alors observer « à la main » la répartition des jours anniversaires. Une simple boucle suffit :

```
1 POUR I ALLANT_DE 1 A 36
2 DEBUT_POUR
3 AFFICHER Classe[I]
4.....AFFICHER " ; "
5 FIN_POUR
```

On obtient alors la liste des 36 nombres choisis au hasard ; ils se présenteront tous sur la même ligne, ce qui est plus facile à lire ; pour pouvoir conserver cet affichage pendant l'élaboration de l'étape suivante et donc vérifier que le tri à réaliser est correct, on placera à la suite de ces instructions, c'est à dire en 6^{ème} ligne de cette liste d'instructions, un affichage « vide » (ou contenant un « blanc ») avec, cochée, l'option « retour à la ligne ».

4. Un tri

On dispose maintenant de la liste des jours où on va fêter un anniversaire ; la deuxième étape consiste à trier cette liste par ordre croissant. On va donc ranger les éléments de cette liste par ordre croissant ; puisque la liste initiale ne sera plus, par la suite, utilisée, on peut utiliser ce même objet informatique pour stocker la liste ordonnée. Le procédé utilisé pour réaliser ce tri est le suivant :

1. Chercher le plus petit élément de la liste
2. Le placer en premier en permutant cet élément avec le premier de la liste
3. Recommencer avec la liste privée de son premier élément.

A chaque étape, le nombre d'éléments à trier diminue d'une unité donc le processus s'arrêtera.

La version utilisée du logiciel Algobox possède deux fonctions très utiles :

- La fonction ALGOBOX_MINIMUM(Liste,I,N) renvoie le plus petit des éléments de la partie de la liste comprise entre le rang n^oI et le rang n^oN.

- La fonction `ALGOBOX_POS_MINIMUM(Liste,I,N)` renvoie le rang du nombre obtenu en utilisant la fonction précédente.

Pour réaliser la permutation entre deux nombres d'une même liste, afin de n'en écraser aucun, on est obligé d'utiliser une mémoire auxiliaire, couramment appelée mémoire-tampon, d'où le nom choisi pour cette variable numérique : « Tampon ».

On construit donc la boucle suivante :

```

23 POUR I ALLANT_DE 1 A 36
24 DEBUT_POUR
25 Tampon PREND_LA_VALEUR ALGOBOX_MINIMUM(Classe,I,36)
26 J PREND_LA_VALEUR ALGOBOX_POS_MINIMUM(Classe,I,36)
27 Classe[J] PREND_LA_VALEUR Classe[I]
28 Classe[I] PREND_LA_VALEUR Tampon
29 FIN_POUR

```

Si le logiciel utilisé ne propose pas de fonction permettant de rechercher un minimum dans une liste, il faudra alors la créer. Le principe en est le suivant : on définit un variable « minimum » ; on affecte à priori à cette variable la première valeur de la liste puis on compare avec la valeur suivante : si cette dernière est plus petite, « minimum » prend cette nouvelle valeur sinon elle est inchangée ; ensuite on compare la valeur de « minimum » ainsi obtenue à l'élément suivant de la liste etc. Une boucle est donc nécessaire ; il faudra aussi stocker le rang de cette valeur. Cette construction peut être un excellent exercice.

A titre de contrôle, lors de l'élaboration de l'algorithme, on pourra faire afficher la liste obtenue ; pour cela, il suffit de recopier (avec l'outil « copier-coller ») l'algorithme d'affichage proposé dans le paragraphe précédent ; si on a conservé l'affichage proposé lors de la construction initiale de la première liste, on obtient une vérification rapide. Pour des raisons de clarté, on a fait disparaître ces affichages du programme définitif proposé ci-dessous.

5. Décompte des jours où on mange du gâteau.

Il s'agit ici de compter combien, dans la liste « classe » y a-t-il de nombres différents. Puisque, maintenant, la liste obtenue contient des nombres rangés par ordre croissant, il suffit de comparer chaque nombre à son successeur.

On utilise un « compteur », la variable « NbreJourFete ». On l'initialise à la valeur 1 puis on parcourt le tableau en comparant chaque élément de la liste à son successeur immédiat ; à chaque fois que les deux nombres sont différents, on ajoute 1 au compteur. `Classe[I+1]` devant rester un élément de la liste, on arrête la boucle lorsque I vaut 35.

« NbreJourFete » donne alors le nombre de jours où on mangera du gâteau ; on peut facilement le comparer à 36, le nombre d'élèves de la classe. On peut faire tourner plusieurs fois le programme et observer les divers résultats obtenus.

36-NbreJourFete est le nombre d'élèves qui apporteront un gâteau alors qu'il y en a déjà à manger. On n'a pas cherché ici à dénombrer les cas où il y aurait exactement deux anniversaires le même jour (respectivement exactement trois...)

6. Compléments :

Dans le programme proposé, on a ajouté une boucle supplémentaire proposant de réaliser 100 fois la simulation et de calculer la moyenne des nombres de jours de fête obtenu pour ces 100 simulations.

7. L'algorithme complet :

ClasseNbrJourFeteRepetition100fois

Cet algorithme détermine les âges des 36 élèves de la classe puis range ces âges par ordre croissant. IL donne ensuite le nombre de jours où il y aura du gâteau à manger. Il répète 100 fois le processus et calcule la moyenne.

```
1 VARIABLES
2 I EST_DU_TYPE NOMBRE
3 Classe EST_DU_TYPE LISTE
4 Tampon EST_DU_TYPE NOMBRE
5 J EST_DU_TYPE NOMBRE
6 RangMin EST_DU_TYPE NOMBRE
7 NbreJourFete EST_DU_TYPE NOMBRE
8 NbANMultiple EST_DU_TYPE NOMBRE
9 CumulNBjourFete EST_DU_TYPE NOMBRE
10 CumulNbmultiple EST_DU_TYPE NOMBRE
11 MoyenneNbjourFete EST_DU_TYPE NOMBRE
12 MoyenneNbMultiple EST_DU_TYPE NOMBRE
13 k EST_DU_TYPE NOMBRE
14 DEBUT_ALGORITHME
15 CumulNBjourFete PREND_LA_VALEUR 0
16 CumulNbmultiple PREND_LA_VALEUR 0
17 POUR k ALLANT_DE 1 A 100
18 DEBUT_POUR
19 POUR I ALLANT_DE 1 A 36
20 DEBUT_POUR
21 Classe[I] PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,365)
22 FIN_POUR
```

```

23 POUR I ALLANT_DE 1 A 36
24 DEBUT_POUR
25 Tampon PREND_LA_VALEUR ALGOBOX_MINIMUM(Classe,I,36)
26 J PREND_LA_VALEUR ALGOBOX_POS_MINIMUM(Classe,I,36)
27 Classe[J] PREND_LA_VALEUR Classe[I]
28 Classe[I] PREND_LA_VALEUR Tampon
29 FIN_POUR
30 NbreJourFete PREND_LA_VALEUR 1
31 POUR I ALLANT_DE 1 A 35
32 DEBUT_POUR
33 SI (Classe[I]!=Classe[I+1]) ALORS
34 DEBUT_SI
35 NbreJourFete PREND_LA_VALEUR NbreJourFete+1
36 FIN_SI
37 FIN_POUR
38 AFFICHER " "
39 AFFICHER "Le nombre de jours où on mangera du gâteau d'anniversaire est : "
40 AFFICHER NbreJourFete
41 AFFICHER "Le nombre d'élèves amenant du gâteau lorsqu'il y en a déjà à manger
est : "
42 NbANMultiple PREND_LA_VALEUR 36-NbreJourFete
43 AFFICHER NbANMultiple
44 CumulNBjourFete PREND_LA_VALEUR CumulNBjourFete+NbreJourFete
45 CumulNbmultiple PREND_LA_VALEUR CumulNbmultiple+NbANMultiple
46
47 FIN_POUR
48 MoyenneNbjourFete PREND_LA_VALEUR CumulNBjourFete/100
49 MoyenneNbMultiple PREND_LA_VALEUR CumulNbmultiple/100
50 AFFICHER " "
51 AFFICHER "Nombre moyen de jours de fête : "
52 AFFICHER MoyenneNbjourFete
53 AFFICHER "Moyenne Nombre jours Fêtes multiples : "
54 AFFICHER MoyenneNbMultiple
55
56 FIN_ALGORITHME

```