IREM d'Aquitaine
Institut de Recherche sur l'Enseignement des Mathématiques
40, rue Lamartine, 33400 Talence

Deviner un nombre

Fiche élève

On souhaite produire un algorithme permettant de faire deviner un nombre entier, choisi aléatoirement par la machine, dans un intervalle choisi par l'utilisateur.

Travail préparatoire

Algobox offre une fonction (nommée `ALGOBOX_ALEA_ENT`) qui permet d'obtenir un nombre choisi aléatoirement dans un intervalle donné. Ainsi, l'instruction :

```
n PREND_LA_VALEUR ALGOBOX_ALEA_ENT(minimum,maximum)
```

donne à la variable `n` une valeur prise aléatoirement dans l'intervalle [`minimum,maximum`], avec `minimum < maximum`. Les deux extrémités de l'intervalle, `minimum` et `maximum`, doivent être des entiers positifs.

1. Créer un algorithme (Algobox) qui lit les deux extrémités (minimum et maximum) d'un intervalle d'entiers et qui affiche un nombre secret (choisi aléatoirement par la machine) appartenant à cet intervalle.

Conception de l'algorithme

2. Modifier l'algorithme précédent afin qu'il n'affiche plus le nombre secret mais qui puisse répondre si un nombre choisi par l'utilisateur est égal au nombre secret.

3. Modifier cet algorithme afin qu'il affiche un message indiquant si le nombre secret est égal, plus grand ou plus petit que le nombre proposé.

4. Modifier cet algorithme afin qu'il redemande un nombre tant que l'utilisateur ne trouve pas le nombre secret.

IREM d'Aquitaine
Institut de Recherche sur l'Enseignement des Mathématiques
40, rue Lamartine, 33400 Talence

Deviner un nombre

Fiche enseignant

Objectifs. Introduction de la structure répétitive `TANT-QUE`. Utilisation de la fonction `ALGOBOX_ALEA_ENT` d'Algobox (choix aléatoire d'un entier dans un intervalle).

Prérequis. Notions de base d'algorithmique (notion de variable, entées-sorties, structure conditionnelle `SI-ALORS-SINON`).

On souhaite produire un algorithme permettant de faire deviner un nombre entier, choisi aléatoirement par la machine, dans un intervalle choisi par l'utilisateur.

Travail préparatoire

Algobox offre une fonction (nommée `ALGOBOX_ALEA_ENT`) qui permet d'obtenir un nombre choisi aléatoirement dans un intervalle donné. Ainsi, l'instruction :

```
n PREND_LA_VALEUR ALGOBOX_ALEA_ENT(minimum,maximum)
```

donne à la variable `n` une valeur prise aléatoirement dans l'intervalle [`minimum,maximum`], avec `minimum < maximum`. Les deux extrémités de l'intervalle, `minimum` et `maximum`, doivent être des entiers positifs.

1. Créer un algorithme (Algobox) qui lit les deux extrémités (minimum et maximum) d'un intervalle d'entiers et qui affiche un nombre secret (choisi aléatoirement par la machine) appartenant à cet intervalle.

Il s'agit essentiellement ici de tester l'utilisation de la fonction `ALGOBOX_ALEA_ENT`*. L'algorithme est le suivant :*

```
AfficherSecret  -  26.01.2015
****************************************
Cet algorithme affiche un nombre (positif) choisi aléatoirement dans un
intervalle dont les extrémités sont fournies par l'utilisateur.
****************************************
1   VARIABLES
2     secret EST_DU_TYPE NOMBRE
3     minimum EST_DU_TYPE NOMBRE
4     maximum EST_DU_TYPE NOMBRE
5   DEBUT_ALGORITHME
6     AFFICHER "Donnez les extrémités de l'intervalle"
7     LIRE minimum
8     LIRE maximum
9     secret PREND_LA_VALEUR ALGOBOX_ALEA_ENT(minimum,maximum)
10    AFFICHER secret
11  FIN_ALGORITHME
```

Conception de l'algorithme

2. Modifier l'algorithme précédent afin qu'il n'affiche plus le nombre secret mais qui puisse répondre si un nombre choisi par l'utilisateur est égal au nombre secret.

On complète l'algorithme précédent en utilisant une variable supplémentaire `nombre_propose` *qui permet de récupérer la valeur proposée par l'utilisateur. La structure conditionnelle* `SI-ALORS-SINON` *permet ensuite d'afficher le bon message…*

L'algorithme est le suivant :

```
TesterSecret  -  26.01.2015
****************************************
Cet algorithme choisit un nombre (positif) aléatoirement dans un intervalle
dont les extrémités sont fournies par l'utilisateur
et propose à l'utilisateur de deviner ce nombre (un seul essai).
****************************************
1   VARIABLES
2     secret EST_DU_TYPE NOMBRE
3     minimum EST_DU_TYPE NOMBRE
4     maximum EST_DU_TYPE NOMBRE
5     nombre_propose EST_DU_TYPE NOMBRE
6   DEBUT_ALGORITHME
7     AFFICHER "Donnez les extrémités de l'intervalle"
8     LIRE minimum
9     LIRE maximum
10    secret PREND_LA_VALEUR ALGOBOX_ALEA_ENT(minimum,maximum)
11    AFFICHER "Entrez votre nombre"
12    LIRE nombre_propose
13    SI (nombre_propose == secret) ALORS
14      DEBUT_SI
15      AFFICHER "Bravo ! Vous avez deviné le nombre secret."
16      FIN_SI
17      SINON
18        DEBUT_SINON
19        AFFICHER "Désolé, vous avez perdu : le nombre secret était "
20        AFFICHER secret
21        AFFICHER "..."
22        FIN_SINON
23  FIN_ALGORITHME
```

3. Modifiez cet algorithme afin qu'il affiche un message indiquant si le nombre choisi est égal, plus grand ou plus petit que le nombre secret.

Il suffit effectivement de modifier la partie « affichage du message résultat » de la façon suivante :

```
13    SI (nombre_propose == secret) ALORS
14      DEBUT_SI
15      AFFICHER "Bravo ! Vous avez deviné le nombre secret."
16      FIN_SI
17      SINON
18        DEBUT_SINON
19        SI (secret < nombre_propose) ALORS
20          DEBUT_SI
21          AFFICHER "Désolé, le nombre que vous avez proposé était trop
grand..."
```

```
22          FIN_SI
23          SINON
24            DEBUT_SINON
25            AFFICHER "Désolé, le nombre que vous avez proposé était trop
petit..."
26            FIN_SINON
27        FIN_SINON
```

4. Modifier cet algorithme afin qu'il redemande un nombre tant que l'utilisateur ne trouve pas le nombre secret.

C'est ici que nous allons devoir utiliser la structure répétitive `TANT-QUE`*…*

Il va falloir cependant réorganiser légèrement l'algorithme. En effet, la répétition du corps de boucle doit s'arrêter dès que l'utilisateur trouve le nombre secret. Le message « gagné » sera donc affiché en dehors du corps de boucle.

Il faut également penser à demander la proposition de l'utilisateur avant de placer la structure répétitive `TANT-QUE`*… En effet, le traitement à répéter est « donner une chance supplémentaire à l'utilisateur en cas d'échec » ; si l'utilisateur trouve le nombre secret au 1er essai, ce traitement n'a pas lieu d'être réalisé…*

***Attention :** il faut bien penser à redemander une proposition de nombre dans le corps de boucle ! En effet, dans le cas contraire, si on rentre dans le corps de boucle on n'en sortira jamais car les valeurs des variables restent les mêmes…*

La partie « traitement » de notre algorithme devient ainsi :

```
6   DEBUT_ALGORITHME
7     AFFICHER "Donnez les extrémités de l'intervalle"
8     LIRE minimum
9     LIRE maximum
10    secret PREND_LA_VALEUR ALGOBOX_ALEA_ENT(minimum,maximum)
11    AFFICHER "Entrez votre nombre"
12    LIRE nombre_propose
13    TANT_QUE (nombre_propose != secret) FAIRE
14      DEBUT_TANT_QUE
15      SI (secret < nombre_propose) ALORS
16        DEBUT_SI
17        AFFICHER "Désolé, le nombre que vous avez proposé était trop
grand..."
18        FIN_SI
19        SINON
20          DEBUT_SINON
21          AFFICHER "Désolé, le nombre que vous avez proposé était trop
petit..."
22          FIN_SINON
23      LIRE nombre_propose
24      FIN_TANT_QUE
25    AFFICHER "Bravo ! Vous avez deviné le nombre secret."
26  FIN_ALGORITHME
```