Mini-projet 1 (Raffinages)

1 Cahier des charges

On souhaite écrire un programme qui aide à **réviser les tables de multiplication**. Voici les attentes concernant ce programme.

1. L'interface avec l'utilisateur sera en mode texte (console) et devra respecter l'exemple donné au listing 1 et expliqué ci-après.

```
Table à réviser : 7

(M1) 7 * 1 ? 7
Bravo !

(M2) 7 * 3 ? 24
Mauvaise réponse.

...

(M10) 7 * 5 ? 35
Bravo !

4 erreurs.  Il faut encore travailler la table de 7.

Des hésitations sur la table de 8 : 6.405475000 secondes contre 1.798116600
en moyenne.  Il faut certainement la réviser.

On continue ? _
```

Listing 1 – Exemple d'exécution du programme de révision des multiplications

2. Le principe général du programme est le suivant. L'utilisateur commence par choisir la table à réviser (entre 1 à 10), le programme lui pose 10 multiplications dont le nombre de gauche est la table à réviser et le nombre de droite est un nombre choisi au hasard (entre 1 à 10 aussi). Le programme demande à l'utilisateur le résultat de cette multiplication et lui indique s'il a bien répondu ou pas (« Bravo ! » ou « Mauvaise réponse. »). À la fin de la série de multiplications, un message est affiché en fonction du nombre d'erreurs commises :

— « Aucune erreur. Excellent !» si toutes les réponses sont justes,

— « Une seule erreur. Très bien. » s'il n'a commis qu'une seule erreur,

— « Tout est faux ! Volontaire ? » si toutes les réponses sont fausses.

— Le nombre de bonnes réponses et une incitation à apprendre cette table s'il y a moins de la moitié de bonnes réponses. Par exemple : « Seulement 2 bonnes réponses. Il faut apprendre la table de 7 ! ».

— Le nombre d'erreurs et une incitation à retravailler cette table dans les autres cas. Par exemple « 4 erreurs. Il faut encore travailler la table de 7. ».

3. La table à réviser saisie par l'utlisateur doit être entre 1 et 10 sinon on lui redemande.

4. À la fin d'une série de multiplications, on conseillera de réviser la table correspondant au nombre de droite de la multiplication pour laquelle l'utilisateur a mis le plus de temps pour répondre, à condition que ce temps soit supérieur d'une seconde au temps moyen.

5. À la fin d'une série de multiplications, on demandera à l'utilisateur s'il veut continuer à s'entraîner ou s'il arrête.

Indication : En Ada, on utilisera le paquetage `Alea` fourni (fichiers `alea.ads` et `alea.adb` dont il n'est pas utile de regarder le contenu). Le fichier `exemple_alea.adb` (listing 2) est un exemple d'utilisation du paquetage `Alea`. `Alea` est un paquetage générique [1] paramétré par deux entiers correspondant aux bornes de l'intervalle dans lequel les nombres aléatoires seront tirés. Pour l'utiliser, le `use Alea;` en début de fichier n'est pas suffisant. Il faut aussi instancier le paquetage pour donner une valeur aux bornes. Ceci se fait dans la partie déclarations :

```
1      package Mon_Alea is new Alea (5, 15);
2          -- Les nombres aléatoires seront dans [5..15]
3      use Mon_Alea;  -- on peut alors utiliser Get_Random_Number
```

Indication : Le programme `mesure_temps.adb` (listing 3) montre comment on peut se servir du module `Calendar` pour obtenir l'heure actuelle et mesurer la durée utilisateur d'une opération.

2 Contraintes

1. Le programme doit fonctionner sur les salles d'enseignement de l'ENSEEIHT.
2. **Il est interdit de définir des sous-programmes et d'utiliser des tableaux.**

3 Documents à rendre

Les **documents à rendre**, dits *livrables*, sont :

— le source du programme : `multiplications.adb`,

— le fichier LISEZ-MOI.txt complété (il contient les raffinages).

4 Principales dates

— lundi 23 septembre 2019 : publication du sujet (Moodle) et des fichiers fournis (SVN),

— fin première séance de TP (séance TP 2) : rendu des raffinages réalisés en TP,

— vendredi 27 septembre 2019 : rendu des raffinages (version finale),

— fin deuxième séance de TP (séance TP 4) : rendu du code source écrit en TP,

— jeudi 10 octobre 2019 : date limite pour rendre la version finale des livrables.

1. Cette notion sera vue plus tard...

```
1  with Ada.Text_IO;           use Ada.Text_IO;
2  with Ada.Integer_Text_IO;   use Ada.Integer_Text_IO;
3  with Alea;
4
5  -- Procédure qui illustre l'utilisation du paquetage Alea.
6  procedure Exemple_Alea is
7
8      package Mon_Alea is
9          new Alea (5, 15);  -- générateur de nombre dans l'intervalle [5, 15]
10     use Mon_Alea;
11
12     Nombre: Integer;
13 begin
14     -- Afficher 10 nombres aléatoires
15     Put_Line ("Quelques nombres aléatoires : ");
16     for I in 1..10 loop
17         Get_Random_Number (Nombre);
18         Put (Nombre);
19         New_Line;
20     end loop;
21 end Exemple_Alea;
```

Listing 2 – Le fichier `exemple_alea.adb`

5 Critères de notation

Voici les principaux critères qui seront pris en compte lors de la correction du projet :

- le respect du cahier des charges,
- la manière dont le programme a été testé,
- la qualité des raffinages,
- la facilité à comprendre la solution proposée,
- l'absence de code redondant,
- le choix des identifiants,
- la cohérence entre les commentaires et les raffinages,
- la bonne utilisation des commentaires,
- le respect de la solution algorithmique (raffinages) dans le programme,
- la présentation du code (le programme doit être facile à lire et à comprendre),
- l'utilisation des structures de contrôle adéquates,
- la validité du programme,
- la robustesse du programme.

```ada
with Ada.Text_IO;           use Ada.Text_IO;
with Ada.Integer_Text_IO;   use Ada.Integer_Text_IO;
with Ada.Calendar;          use Ada.Calendar;

procedure Mesure_Temps is
    N: Integer;          -- un entier lu au clavier
    Debut: Time;         -- heure de début de l'opération
    Fin: Time;           -- heure de fin de l'opération
    Delai : Duration;    -- durée de l'opération
begin
    -- récupérer l'heure (heure de début)
    Debut := Clock;

    -- réaliser l'opération
    Put_Line ("Début");
    Put ("Valeur : ");
    Get (N);
    Put_Line ("Fin");

    -- récupérer l'heure (heure de fin)
    Fin := Clock;

    -- calculer la durée de l'opération
    Delai := Fin - Debut;

    -- Afficher la durée de opération
    Put ("Durée :" & Duration'Image(Delai));
end Mesure_Temps;
```

Listing 3 – Le fichier `mesure_temps.adb`