PIM : Mini-projet 1

Auteur : Maxime Moshfeghi

Temps passé sur les raffinages : 2 h 00
Temps passé sur la programmation : 4 h 00
Temps passé sur la mise au point : ? h …

Raffinages

-- On définit une fonction Aléatoire(a,b) prenant en entrée deux entiers a et b tels que a < b. Cette fonction renvoie un nombre aléatoire compris entre a et b (pouvant être égale à a ou à b).

R0 : Faire réviser les tables de multiplication

R1 : **Comment** « Faire réviser les tables de multiplication » ?
    -- Introduction d'un booléen qui permet de recommencer l'entraînement
    Répéter
        Demander un entier k dont on veut faire réviser la table de multiplication
k : **out entier**
        { (k >= 0) et (k <= 10) -- l'entier est compris entre 0 et 10 }
        Proposer dix multiplications de ce chiffre par dix entiers xi k : **in entier**
        { Pour tout entier i compris entre 0 et 10, (xi >= 0) et (xi <= 10) -- l'entier est compris entre 0 et 10 }
        -- Afficher pour chaque multiplication si la réponse est bonne ou mauvaise
        Afficher un message personnalisé selon les réponses de l'utilisateur
        Proposer de continuer veut_travailler : **out**
    JusquÀ Non veut_travailler veut_travailler : **in out**

R2 : **Comment** « Demander un entier k dont on veut faire réviser la table de multiplication » ?
    OK <- False OK : **out booléen** -- variable booléenne qui vérifie si k est entre 0 et 10
    TantQue Non OK **Faire** -- On demande à l'utilisateur une table à réviser et on affiche un message d'erreur si cet entier n'est pas entre 0 et 10
        Ecrire (“Table à réviser : ”)
        Lire (k)
        Si k >= 0 Et k <= 10 **Faire**
            OK <- True
        Sinon Faire
            Ecrire ("Impossible. La table doit être comprise entre 0 et 10.")
        Fin Si
    FinTQ

R2 : **Comment** « Proposer dix multiplications de ce chiffre par dix entiers xi » ?
    compt <- 0 compt : **out entier**
    -- On introduit un compteur qui va compter le nombre de bonnes réponse
    Pour i De 1 À 10 **Faire**

```
        xi <- Aléatoire(0,10)          xi : out entier
        Ecrire ("(M")
        Ecrire (i)
        Ecrire (") ")
        Ecrire (k)
        Ecrire (" * ")
        Ecrire (xi)
        Ecrire (" ? ")
        Lire (rep)                rep : out entier
        Incrémenter ou pas le compteur de bonnes réponses selon la réponse donnée
    FinPour
```

R2 : **Comment** « Afficher un message personnalisé selon les réponses de l'utilisateur » ?

```
    Selon compt Dans
        10 => Ecrire ("Aucune erreur. Excellent !")
        9 => Ecrire ("Une seule erreur. Très bien.")
        0 => Ecrire ("Tout est faux ! Volontaire ?")
        1..5 =>     Ecrire ("Seulement ")
                    Ecrire (compt)
                    Ecrire ("bonnes réponses. Il faut apprendre la table de ")
                    Ecrire (k)
        Autres =>   Ecrire (compt)
                    Ecrire (" erreurs. Il faut retravailler la table de ")
                    Ecrire (k)
    FinSelon
```

R2 : **Comment** « Proposer de continuer » ?

```
    Ecrire ("On continue (o/n) ?")
    Lire (car)              car : out caractère
    Selon car Dans
        'O', 'o' => veut_travailler <- True          veut_travailler : out booléen
        Autres => veut_travailler <- False           veut_travailler : in out booléen
    FinSelon
    -- le caractère "car" contient la réponse de l'utilisateur
```

R3 : **Comment** « Incrémenter ou pas le compteur de bonnes réponses selon la réponse donnée » ?

```
    Si k * xi == rep Alors      -- On vérifie si la réponse est bonne
        Ecrire ("Bravo !")
        temp_compt <- compt              temp_compt : out ; compt : in
        -- On introduit une variable temporaire pour pouvoir ensuite incrémenter
        compt <- temp_compt + 1          compt, temp_compt : in out entier
    Sinon
```

```
        Ecrire(“Mauvaise réponse.”)
    FinSi

...
```

Evaluation des raffinages

		Evaluation Etudiant (I/P/A/+)	**Justification / commentaire**	**Evaluation Enseignant (I/P/A/+)**
Forme (D-21)	Respect de la syntaxe Ri : Comment "... une action complexe ..." ? des actions combinées avec des structures de controle Rj : ...	+		P
	Verbe à l'infinitif pour les actions complexes	+		A
	Nom ou équivalent pour expressions complexes	?		A
	Tous les Ri sont écrits contre la marge et espacés	+		A
	Les flots de données sont définis	A		A
	Une seule décision ou répétition par raffinage	A		A
	Pas trop d'actions dans un raffinage (moins de 6)	P		A
	Bonne présentation des structures de contrôle	A		P
Fond (D21-D22)	Le vocabulaire est précis	A		P
	Le raffinage d'une action décrit complètement cette action			A
	Le raffinage d'une action ne décrit que cette action	A		A
	Les flots de données sont cohérents	A		A
	Pas de structure de contrôle déguisée	P		A
	Qualité des actions complexes	A		A

Evaluation du code

		Consigne : Mettre O (oui) ou N (non) dans la colonne Etudiant suivant que la règle a été respectée ou non. Une justification peut être ajoutée dans la colonne “commentaire”.	
Commentaire	**Etudiant (O/N)**	**Règle**	**Enseignant (O/N)**
	O	Le programme ne doit pas contenir d'erreur de compilation.	
	O	Le programme doit compiler sans messages d’avertissement.	
	O	Le code doit être bien indenté.	
	O	Les règles de programmation du cours doivent être respectées : toujours un Sinon pour un Si, pas de sortie au milieu d'une répétition...	
	O	Pas de code redondant.	
	O	On doit utiliser les structures de contrôle adaptées (Si/Selon/TantQue/Répéter/Pour)	
	O	Utiliser des constantes nommées plutôt que des constantes littérales.	
	O	Les raffinages doivent être respectés dans le programme.	
	O	Les actions complexes doivent apparaître sous forme de commentaires placés AVANT les instructions correspondantes.	
		Une ligne blanche doit séparer les principales actions complexes	
	O	Le rôle des variables doit être explicité à leur déclaration (commentaire).	

Prise en compte d’évolutions possibles

Répondre de manière concise et précise aux questions posées. Ces évolutions ne doivent pas être implantées dans votre programme.

Question 1 : Au lieu de poser 10 questions, on veut en poser 15. Comment faire ?

Réponse : Si on veut poser 15 questions, il suffit de changer les bornes de la boucle “for” (ligne 102), en changeant la borne supérieur par 15 au lieu de 10. On fait aussi attention à changer le “10 - compt” de la ligne 137 en “15 - compt”.

Question 2 : On veut afficher "Bien" si l'utilisateur n'a commis que 2 ou 3 erreurs. Comment modifier le programme ?

Réponse : Il suffit de rajouter un cas dans la structure de contrôle “Selon” qui sert à afficher le message bilan des performances de l’utilisateur (“case” ligne 132), englobant donc le cas où le compteur vaut 8 ou 7.

Question 3 : On veut donner la possibilité à l'utilisateur d'abandonner le programme en tapant -1 quand on lui demande le résultat d'une multiplication. Quelles modifications faut-il alors faire au programme ?

Réponse : Il suffit de rajouter un “Sinon si” (“elsif”) dans la structure de contrôle “Si” permettant de vérifier si la réponse est la bonne (“if” ligne 114).

Question 4 : On veut faire réviser les tables de 0 à 20. Comment modifier le programme ?

Réponse : Il y a plusieurs choses à changer :

- dans un premier temps, il faut changer la borne supérieure du générateur d’entiers aléatoires (ligne 77)
- la condition de la structure de contrôle “Si” qui permet d’attester que l’utilisateur a bien entré un entier entre 0 et 10 est donc à changer (“if” ligne 90)
- on change également la valeur initial de la variable tampon “xi_temp” à 21 (ligne 100)

Remarque : dans le cas on l’on veut facilement changer ce paramètre dans le programme, il aurait été judicieux d’introduire une constante de la valeur de la table maximale que l’on veut faire réviser.

Question 5 : À la fin d'une série de questions, on veut proposer à l'utilisateur de réviser la table pour laquelle l'utilisateur a commis le plus d'erreurs. Par exemple, s'il se trompe pour 3 * 5, on compte une erreur pour 5 mais pas pour 3. Comment faire ?

Réponse : On peut créer 11 variables entières relatives aux entiers de 0 à 10 qui stockent le nombre d'erreurs commises avec chacun d'eux. Il faudrait alors imbriquer un "Selon" dans le "Sinon" de la structure "Si" qui vérifie si la réponse est bonne (après le "else" de la ligne 117).

Question 6 : Si l'utilisateur fait 4 erreurs, on arrête de poser les multiplications en disant qu'il faut aller apprendre la table. Comment modifier le programme ?

Réponse : On peut choisir d'opter pour une structure de contrôle "TantQue" au lieu de la boucle "Pour" qui permet de faire les 10 multiplications. Il faut alors introduire une variable entière que l'on incrémente à chaque passage et qui ne doit pas excéder 10 (première condition) et on fait aussi attention que l'écart entre cet entier et le compteur de bonnes réponses n'excède pas strictement 3 (deuxième condition).

Question 7 : Comment l'extension 1 a été prise en compte (pas deux fois de suite la même multiplication) ?

Réponse : J'ai d'abord initialisé une variable entière "xi_temp" à 11. Celle-ci est censée être une variable temporaire conservant le dernier entier aléatoire par lequel on multiplie l'entier dont on veut réviser la table. J'ai ensuite fait une petite boucle "TantQue" qui redéfinit l'entier aléatoire généré "xi" tant que celui-ci est égal à l'entier précédent "xi_temp".
Remarque : il n'y a pas de problème au premier passage dans la boucle "TantQue" car la variable "xi_temp" est initialisée à 11, donc forcément différente de la première valeur que va prendre "xi" (entre 0 et 10).

Question 8 : Comment l'extension 2 a été prise en compte (proposer de réviser la table avec la plus grande hésitation) ?

Réponse : J'ai réalisé un algorithme de candidat. L'idée est d'évaluer à chaque multiplication le temps que met l'utilisateur à répondre. À chaque multiplication proposée, si l'utilisateur met plus de temps pour répondre, la variable "Candidat" de type "Duration" est écrasée, ainsi que la variable entière "xi_Candidat" qui sert à conserver l'entier pour lequel la multiplication a été la plus longue à résoudre.
Remarque : La variable "Candidat" est initialisé à 0.0. Ainsi, en premier passage de boucle, Candidat prend obligatoirement la valeur du temps mis par l'utilisateur pour répondre à cette

première multiplication, et la variable entière “xi_Candidat” prend obligatoirement la valeur du premier entier tiré aléatoirement.

Difficultés rencontrées

TODO : Indiquer ici les difficultés rencontrées dans la réalisation de ce mini-projet.
(cette partie n’est pas prise en compte dans la notation)

…

Informations complémentaires

TODO : Indiquer ici ce qui est utile à l’enseignant pour comprendre les raffinages et/ou le programme correspondant. Cette partie peut être vide.

Bilan

TODO : Dire quel bilan vous tirez de ce mini-projet Cette partie n’est pas prise en compte dans la notation !