PIM: Mini-projet 1

Auteur: SCHLÖGEL Benjamin

Temps passé sur les raffinages : 4 h 30 Temps passé sur la programmation : 6 h Temps passé sur la mise au point : 2 h

Raffinages	2
Evaluation des raffinages	3
Evaluation du code	4
Prise en compte d'évolutions possibles	5
Difficultés rencontrées	7
Informations complémentaires	8
Bilan	9

Raffinages

TODO: écrire ici les raffinages.

On ne donnera pas d'exemple car ils vont prendre beaucoup de place. Il est cependant utile d'en prendre pour vérifier la bonne compréhension de l'énoncé.

R0 : Faire réviser les tables de multiplications

R1: Comment << Faire réviser les tables de multiplications >> ? Recommencer ← "o"

Recommencer : in Chaîne de caractères

TANT QUE (Recommencer = "o") OU (Recommencer = "O") FAIRE

{ Variant : Recommencer }

Permettre à l'utilisateur de réviser une table de multiplication désirée

FIN TANT QUE

R2 : Comment << Permettre à l'utilisateur de réviser une table de multiplication désirée >> ?

Demander la table à réviser (au clavier)

Table : out Entier ; Départ : in out Booléen

{ 0 <= Table <= 10 }

Proposer dix calculs sans réponse avec la table saisie et permettre à l'utilisateur de donner les réponses, on en retient les erreurs commises.

Erreurs, Nombres_calculs , Réponse : in out Entier

{ 0 <= Erreurs <= 10 }

Afficher le nombre d'erreurs et conseiller l'utilisateur en fonction de ce dernier { Afficher Erreurs et un conseil }

Erreurs: in

Demander à l'utilisateur de réviser à nouveau une table (au clavier)

Recommencer: out

{ Recommencer permet au programme de revenir au début, ou pas }

```
R3: Comment << Demander la table à réviser >> ?
{ Variant : Table, Départ }
Table ← 0
Départ ← Faux
TANT QUE NonDépart Faire
      Écrire (Table à réviser :)
      Lire (Table)
      Contrôler la valeur de Table et adapter la valeur de Départ
FIN TANT QUE
R3 : Comment << Proposer dix calculs sans réponse avec la table saisie et permettre à
l'utilisateur de donner les réponses >> ?
{ Variant : Erreurs , Nombre_calculs, Réponse }
{ Invariant : Table }
Erreurs ← 0
Nombre_calculs \leftarrow 0
Réponse ← 0
TANT QUE Nombre_calculs < 10 FAIRE BOUCLE FOR
      Nombres_calculs ← Nombres_calculs + 1
      Écrire (Table + " *" + Nombre_aléatoire + "?")
      Lire (Réponse)
      Contrôler Réponse et adapter la valeur d'Erreurs en avertissant l'utilisateur
FIN TANT QUE
```

R3 : Comment << Afficher le nombre d'erreurs et conseiller l'utilisateur en fonction de ce dernier >> ?

```
{ Invariant : Bonnes réponses, Erreurs }
Bonnes réponses ← 10 - Erreurs
SI Erreurs = 0 FAIRE
      Écrire ("Aucune erreur. Excellent!")
SINON SI Erreurs = 1 FAIRE
      Écrire ("Une seule erreur. Très bien.")
SINON SI Erreurs = 10 FAIRE
      Écrire ("Tout est faux ! Volontaire ?")
SINON SI 6 <= Erreurs <= 9 FAIRE
      Écrire ("Seulement" + Bonnes réponses + "bonnes réponses. Il faut
apprendre la table de " + Table + "!")
SINON 2 <= Erreurs <= 5 FAIRE
      Écrire (Erreurs + " erreurs. Il faut encore travailler la table de " + Table + ".")
FIN SI
R3: Comment << Demander à l'utilisateur de réviser à nouveau une table >> ?
Écrire ("On continue (o/n)?")
Lire (Recommencer)
R4 : Comment << Contrôler la valeur de Table et adapter la valeur de Départ >> ?
{ Variant : Départ }
{ Invariant : Table }
SI (0 <= Table <= 10) ET (Table est un Entier) Faire
      Départ ← Vrai
SINON
      Écrire (Impossible. La table doit être entre 0 et 10.)
FIN SI
```

```
{ Variant : Erreurs, Nombre_aléatoire }
{ Invariant : Table, Réponse }
SI Réponse /= Table * Nombre_aléatoire FAIRE
Écrire ("Mauvaise Réponse")
Erreurs ← Erreurs + 1
SINON FAIRE
Écrire ("Bravo!")
```

FIN SI

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 :	A		A
	Verbe à l'infinitif pour les actions complexes	Α		A
	Nom ou équivalent pour expressions complexes	Р		Р
	Tous les Ri sont écrits contre la marge et espacés	А		A
	Les flots de données sont définis	А		А
	Une seule décision ou répétition par raffinage	А		Р
	Pas trop d'actions dans un raffinage (moins de 6)	Р		Р
	Bonne présentation des structures de contrôle	Р		Р
Fond (D21-D 22)	Le vocabulaire est précis	Р		Р
	Le raffinage d'une action décrit complètement cette action	А		Р
	Le raffinage d'une action ne décrit que cette action	А		A
	Les flots de données sont cohérents	Р		Р
	Pas de structure de contrôle déguisée	Р		Р
	Qualité des actions complexes	Р		Р

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)
	0	Le programme ne doit pas contenir d'erreur de compilation.	
	0	Le programme doit compiler sans messages d'avertissement.	
	0	Le code doit être bien indenté.	
		Les règles de programmation du cours doivent être respectées : toujours un Sinon pour un Si, pas de sortie au milieu d'une répétition	
On retrouve plusieurs fois New_Line, mais cela pour respecter l'exemple de l'énoncé.	O	Pas de code redondant.	
	0	On doit utiliser les structures de contrôle adaptées (Si/Selon/TantQue/Répéter/Pour)	
	0	Utiliser des constantes nommées plutôt que des constantes littérales.	
Il faut malgré tout prendre en considération les modifications apportées par les extensions qui ne sont pas présentes dans le raffinage.	0	Les raffinages doivent être respectés dans le programme.	

0	Les actions complexes doivent apparaître sous forme de commentaires placés AVANT les instructions correspondantes.	
О	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 : Il suffit de modifier la valeur de Nb_Questions à 15 dans la déclaration des variables. Cela va augmenter la répétition de la boucle for pour les calculs posés, mais également l'intervalle d'entier pour Get_Random_Number. Cela va aussi modifier différents intervalles dans la condition SI pour la conclusion sur le nombre d'erreurs de l'utilisateur, mais aussi le calcul du temps moyen pour répondre à un calcul.

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

Réponse : Pour cela, il faut se rendre dans le Case lors de la conclusion sur le nombre d'erreurs et rajouter exactement :

when $2 \mid 3 \Rightarrow Put$ ("Bien.");

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 : Pour cela, il faut se rendre dans la structure de contrôle Si de la vérification du calcul posé, remplacer < if Reponse /= Table*Nombre_Aleatoire then > par < elsif Reponse /= Table*Nombre Aleatoire then >, et rajouter juste au-dessus :

if Reponse = -1 then exit;

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

Réponse : Dans la structure de contrôle While pour la vérification de la valeur de la table choisie, il faut remplacer < if (0<=Table) and (Table<=10) then > par :

if (0<=Table) and (Table<=20) then

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 : Il serait intéressant d'ajouter un tableau qu'on définirait avant Begin, de taille égale à l'intervalle de table de multiplication possible. On associerait une case du tableau à une table, par exemple : la case 1 correspond au nombre d'erreurs commises pour la table de 0. Ainsi, dans chaque case, on compte les erreurs émises pour chaque table et à la fin, on regarde le tableau pour laquelle l'utilisateur a commis le plus d'erreurs en parcourant le tableau et on lui conseille de réviser cette table.

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 : Dans la boucle For proposant les dix calculs pour une table donnée et aprés la structure Si incrémentant la valeur de Erreurs, il faut ajouter :

exit when Erreurs = 4;

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

Réponse : On définit la variable doublon initialisé à -1 (pour être sûr que Nombre_aleatoire est différent de Doublon au premier passage dans la boucle For pour les calculs). Au début de la boucle For (pour les 10 calculs) il suffit de créer une boucle While qui tant que Doublon est égale à Nombre_aleatoire alors, on re-pioche un nombre aléatoire avec Get_Rnadom_Number. Une fois la boucle passée, on donne à Doublon la valeur de Nombre_aleatoire pour la vérification lors du prochain calcul.

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

Réponse : Il a fallu dans un premier temps ajouter les extensions au programme avec la commande "with" puis créer les variables nécessaires aux différents calculs qu'on souhaite faire tels que : Debut, Fin, Temps_max, Temps_moy, Table_conseillee. Avant de poser le calcul dans la structure de contrôle For, on mesure le temps avec Debut et une fois que l'utilisateur a répondu, on mesure le temps avec Fin. Avec un Si on vérifie si Fin-Debut est supérieur à Temps_max et si c'est le cas, on lui attribue la valeur et on retient la valeur de la table en question dans Table_conseillee. Pour calculer Temps_moy on somme cette variable avec Fin-Debut pour chaque calcul et une fois toutes les questions passées, on la divise par le nombre de questions. Ensuite, on regarde s'il y a au moins une seconde de différence entre Temps_max et Temps_moy, si c'est le cas, on avertit l'utilisateur comme présenté dans l'énoncé.

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)

Le principal problème que j'ai rencontré était la correction de mes erreurs lors de la compilation de mon programme. J'ai fait des erreurs sur des inégalités interdites dû au type de mes variables. Il a fallu comprendre pourquoi il s'agissait d'erreurs et trouver la syntaxe adaptée pour modifier mes variables.

Un autre problème étant la disposition des phrases dans la console, il faut comprendre où placer New_Line et utiliser Put_Line() si je le pouvais.

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 !