PIM : Mini-projet 1

Emmanuel Dubois : Exercice 1 & 3
Vianey Hervy : Exercice 2

TODO : Nommer votre document PIM-MP1-Equipe-XN où XN correspond au numéro d'équipe (voir "choisir mon équipe" sur Moodle).

Raffinages exercice 1

Les raffinages

R0 : Faire jouer l'ordinateur au jeu du devin.

```
Exemple de déroulement d'une partie :
Nombre choisi par l'ordinateur : 967

Entrées de l'utilisateur       |           Sorties de l'ordinateur
- - - - - - - - - - - - - - -  | - - - - - - - - - - - - - - - - - -
```

```
900                          |      'Trop petit'
990                          |      'Trop grand'
960                          |      'Trop petit'
967                          |      'Trouvé !'
```

R1 : Comment "faire jouer l'ordinateur au jeu du devin" ?

Initialiser une intervalle dans laquelle l'ordinateur choisira son nombre — min, max : **out Entier**

Faire choisir un nombre à l'ordinateur — nombre : **out Entier**; min,max : **in Entier**

Incrémenter un compteur — c : **out Entier**

Faire deviner le nombre à l'utilisateur

Afficher le nombre d'essais lorsque l'utilisateur réussit — dev, nombre,c : **in Entier**

R2 : Comment "initialiser un intervalle dans lequel l'ordinateur choisira son nombre" ?

```
min <- 1
max <- 999
```

R2 : Comment "faire choisir un nombre à l'ordinateur" ?

```
nombre <- entier aléatoire compris dans les bornes
{ (nombre >= min) ET (nombre <= max)  - - le nombre est
compris entre 1 et 999}
```

R2 : Comment "faire deviner le nombre à l'utilisateur" ?

Répéter

Demander un nombre à l'utilisateur — dev : **out Entier**; c : **in Entier**

```
- - comme on considère que l'utilisateur ne rentrera que
des entiers et on n'a donc pas besoin de réaliser de
robustesse sur ce point
```

Vérifier si l'utilisateur a trouvé et donner un indice à l'utilisateur si nécessaire — dev, nombre : **in Entier**

JusquÀ dev = nombre

R3 : Comment "demander un nombre à l'utilisateur" ?

```
Ecrire("Proposition {c}:")
Lire(dev)
```

R3 : Comment “vérifier si l’utilisateur a trouvé et donner un indice à l’utilisateur si nécessaire” ?

```
Si dev = nombre Alors
      Ecrire(“Trouvé !”)
SinonSi dev > nombre Alors
      Ecrire(“Trop grand”)
Sinon
      Ecrire(“Trop petit”)
FinSi
```

R2 : Comment “incrémenter un compteur” ?

```
c <- c + 1
```

R2 : Comment “afficher le nombre d’essais si l’utilisateur réussit” ?

```
Si dev = nombre Alors
      Ecrire(“Bravo. Vous avez trouvé {nombre} en {c} essais.”)
```

Evaluation par l’autre étudiant

		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		
	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		
	Une seule décision ou répétition par raffinage	A		
	Pas trop d'actions dans un raffinage (moins de 6)	+		
	Bonne présentation des structures de contrôle	A		
Fond (D21-D22)	Le vocabulaire est précis	A		
	Le raffinage d'une action décrit complètement cette action	A		

	Le raffinage d'une action ne décrit que cette action	A		
	Les flots de données sont cohérents	A		
	Pas de structure de contrôle déguisée	A		
	Qualité des actions complexes	A		

Remarques diverses

Raffinages exercices 2

Les raffinages

R0 : Faire jouer le jeu du devin à l'ordinateur

Exemple :

L'utilisateur choisit 187

```
     Sorties de l'ordinateur      |  Entrées de l'utilisateur
- - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - -
Je pense que votre nombre est 500 |          'g'
Je pense que votre nombre est 250 |          'g'
Je pense que votre nombre est 125 |          'p'
Je pense que votre nombre est 187 |          't'
```

R1 : Comment faire jouer le jeu du devin à l'ordinateur ?

```
     Initialiser les bornes et l'indice des propositions
                                                  min, max, i: out
     Répéter
          Choisir intelligemment un nombre          min, max: in;
                                                          n: out
          Proposer ce nombre à l'utilisateur             n, i: in
          Demander sa réponse à l'utilisateur         reponse: out
          Traiter la réponse de l'utilisateur      n, reponse: in;
                              min, max, nombre_trouve, triche: out
          Vérifier si l'utilisateur triche          min, max: in;
```

```
                                                        triche: out
    JusquÀ nombre_trouve ou triche
    Afficher (ou non) la triche                          triche: in
    Afficher (ou non) la victoire                  n, reponse, i: in
```

R2 : Comment initialiser les bornes et l'indice des propositions ?

```
    min <- 1
    max <- 999
    i <- 0
```

R2 : Comment choisir intelligemment un nombre ?

```
    n <- (min + max)/2 -- la recherche par dichotomie impose le choix du nombre au "milieu" de l'intervalle.
```

R2 : Comment proposer ce nombre à l'utilisateur ?

```
    Écrire ("Proposition ")
    Écrire (i)
    Écrire (" :")
    Écrire (n)
```

R2: Comment demander sa réponse à l'utilisateur ?

```
    Écrire ("Trop (g)rand, trop (p)etit ou (t)rouvé ?")
    Lire (reponse)
```

R2 : Comment traiter la réponse de l'utilisateur ?

```
    Selon reponse Dans
        'p'|'P' => min <- n
                   triche <- max < min
                   i <- i + 1
        'g'|'G' => max <- n
                   triche <- max < min
                   i <- i + 1
        't'|'T' => nombre_trouve <- vrai
    FinSelon
```

R2 : Comment afficher (ou non) la triche ?

```
    Si triche  Alors
        Écrire ("Vous trichez. J'arrête cette partie.")
    FinSi
```

R2 : Comment afficher (ou non) la victoire ?

```
    Si nombre_trouve Alors
        Écrire ("J'ai trouvé ")
        Écrire (n)
```

```
        Écrire (“ en “)
        Écrire (i)
        Écrire (“ essai(s).”)
    FinSi
```

Evaluation par l’autre étudiant

		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		
	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		
	Une seule décision ou répétition par raffinage	A		
	Pas trop d'actions dans un raffinage (moins de 6)	A		
	Bonne présentation des structures de contrôle	A		
Fond (D21-D22)	Le vocabulaire est précis	A		
	Le raffinage d'une action décrit complètement cette action	+		
	Le raffinage d'une action ne décrit que cette action	A		
	Les flots de données sont cohérents	+		
	Pas de structure de contrôle déguisée	A		
	Qualité des actions complexes	A		

Remarques diverses

Raffinages exercices 3

Les raffinages

R0 : Jouer avec l'ordinateur au jeu du devin.

Exemple d'exécution :

```
     Sorties de l'ordinateur      |  Entrées de l'utilisateur
- - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - -
Voulez-vous choisir ou deviner un |
-nombre [c/d] ?                   |          'c'
Je pense que votre nombre est 500 |          'g'
Je pense que votre nombre est 250 |          'g'
Je pense que votre nombre est 125 |          'p'
Je pense que votre nombre est 187 |          't'
Voulez-vous choisir ou deviner un |
-nombre [c/d] ?                   |          'd'
                                  |          900
          'Trop petit'            |          990
          'Trop grand'            |          932
          'Trop petit'            |          940
           'Trouvé !'             |
```

R1 : Comment "jouer avec l'ordinateur au jeu du devin" ?

Initialiser un intervalle dans lequel l'ordinateur choisira son nombre — min, max : **out Entier**

Demander à l'utilisateur si il veut deviner, s'il veut que l'ordinateur devine ou s'il veut quitter — entree : **in Entier**

Si entree = 2 **Alors**
Lancer une partie où l'ordinateur devine le nombre
SinonSi entree = 1 **Alors**
Lancer une partie où l'ordinateur choisit un nombre
Sinon
Quitter
FinSi

R2 : Comment "initialiser un intervalle dans lequel l'ordinateur choisira son nombre" ?
min <- 1
max <- 999

R2 : Comment "demander à l'utilisateur s'il veut deviner, s'il veut que l'ordinateur devine ou s' il veut quitter" ?
Afficher le menu correspondant
Répéter
Lire(entree) entree : **out Entier**
Vérifier que l'entrée soit conforme
JusquÀ entree = 1 **ou** entree = 2 **ou** entree = 0

R3 : Comment "Afficher le menu correspondant" ?
Ecrire("1- L'ordinateur choisit un nombre et vous le devinez")
Ecrire("2- Vous choisissez un nombre et l'ordinateur le devine")
Ecrire ("0- Quitter le programme")

R3 : Comment "Vérifier que l'entrée soit conforme" ?
Si entree < 0 **ou** entree > 2 **Alors**
Ecrire("Choix incorrect.")

R2 : Comment "lancer une partie où l'ordinateur devine le nombre" ?
Exécuter le sous-programme de l'exercice 2

R2 : Comment "lancer une partie où l'ordinateur choisit un nombre" ?
Exécuter le sous-programme de l'exercice 1

R2 : Comment "Quitter" ?
Ecrire("Au revoir...")
Fin Jeu

Evaluation par l’autre étudiant

		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		
	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		
	Une seule décision ou répétition par raffinage	A		
	Pas trop d'actions dans un raffinage (moins de 6)	A		
	Bonne présentation des structures de contrôle	A		
Fond (D21-D22)	Le vocabulaire est précis	A		
	Le raffinage d'une action décrit complètement cette action	A		
	Le raffinage d'une action ne décrit que cette action	A		
	Les flots de données sont cohérents	A		
	Pas de structure de contrôle déguisée	A		
	Qualité des actions complexes	A		

Remarques diverses

Bilan

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'erreurs 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, avec la même indentation	
	O	Une ligne blanche doit séparer les principales actions complexes	
	O	Le rôle des variables doit être explicité à leur déclaration (commentaire).	