PIM : Mini-projet 1

Raffinages

R0 : Jouer au jeu des 13 allumettes

R1 : Comment "Jouer au jeu des 13 allumettes" ?

nombre_allumettes <-- 13

Demander à l'humain le type de joueur qu'il veut affronter (en clavier)

mode_de_jeu: caractère out

Demander à l'humain s'il veut commencer (en clavier)

variable_commencer: caractere in out

commencer: boolean out

Affronter l'humain selon le mode choisi

nombre_allumettes: entier in

R2 : Comment " Demander à l'humain quel type de joueur il veut affronter" ?

Ecrire('Niveau de l'ordinateur (n)aïf, (d)istrait, (r)apide ou (e)xpert ?')

Lire(mode_de_jeu)

Selon mode_de_jeu Dans

'n' | 'N' => Ecrire('Mon niveau est naif.')

'd' | 'D' => Ecrire('Mon niveau est distrait.')

'r' | 'R' => Ecrire('Mon niveau est rapide.')

Autre => Ecrire('Mon niveau est expert.')

R2: Comment "Demander si l'humain veut commencer" ?

Écrire('Est-ce que vous commencez (o/n) ?')

Lire(variable_commencer)

Si variable_commencer='o' ou variable_commencer='O' **Alors**

commencer<--Vrai

FinSi

R2 : comment "Affronter l'humain selon le mode choisi"?

Tant que nombre_allumettes>0 **Faire**

Affronter l'humain selon son choix de départ

FinTQ

Annoncer le gagnant

gagne_hum: boolean in

{ nombre_allumette=0}

R3: Comment "Affronter l'humain selon son choix de départ"?

Si commence

Déterminer le nombre maximal d'allumettes qu'on peut choisir

nombre_max: entier out

nombre_allumettes: entier in

Faire le choix d'allumette pour l'humain

valide_humain: boolean in out

nombre_allumettes: entier in out

nb_choix_hum: entier in

Vérifier si l'humain a perdu

gagne_hum:boolean out

nombre_allumettes: entier in

gange_joueur: boolean in

Déterminer le nombre maximal d'allumettes qu'on peut choisir

nombre_max: entier out

nombre_allumettes: entier in

Faire le choix d'allumette pour le joueur

valide_joueur: boolean out

nombre_allumettes: entier in out

nb_allu_choix_pc: entier in

Vérifier si le joueur a perdu

nombre_allumettes: entier in

gagne_hum: boolean in

gagne_joueur: boolean out

Sinon

Déterminer le nombre maximal d'allumettes qu'on peut choisir

nombre_max: entier out

nombre_allumettes: entier in

Faire le choix d'allumette pour le joueur

valide_joueur: boolean out

nombre_allumettes: entier in out

nb_allu_choix_pc: entier in

Vérifier si le joueur a perdu

nombre_allumettes: entier in

gagne_hum: boolean in

gagne_joueur: boolean out

Déterminer le nombre maximal d'allumettes qu'on peut choisir

nombre_max: entier out

nombre_allumettes: entier in

Faire le choix d'allumette pour l'humain

valide_humain: boolean in out

nombre_allumettes: entier in out

nb_choix_hum: entier in

Vérifier si l'humain a perdu

gagne_hum:boolean out

nombre_allumettes: entier in

gange_joueur: boolean in

Fin Si

R3: Comment “Annoncer le gagnant”?

Si gagne_hum Alors

Ecricre(‘Vous avez gagner’)

Sinon

Ecrire(‘Vous avez Perdu’)

R4: Comment “Faire le choix d’allumette pour l’humain”

Tant que valide_humain **Faire**

Demander à l’humain le nombre d’allumette qu’il veut prendre

nb_allu_choix_hum: entier out

Vérifier si le nombre d’allumettes prisent par l’humain convient

valide_humain: boolean out

nb_all_choix_hum: entier in

nombre_max: entier in

FinTQ

valide_humain<-- Vrai

nombre_allumettes <-- nombre_allumettes – nb_allu_choix_hum

R4: Comment “Vérifier si l’humain a perdu”

Si nombre_allumettes=0 et non gagne_joueur **Alors**

gagne_hum <-- Vrai

Sinon

Afficher les allumettes restant

nombre_allumettes: entier in

FinSi

R4: Comment “Faire le choix d’allumette pour le joueur”

Tant que valide_joueur **Faire**

Jouer selon le mode choisi

mode_de_jeu: boolean in

nb_allu_choix_pc: entier out

nombre_max: entier in

Vérifier si le nombre d’allumettes prisent par le joueur convient

valide_joueur: boolean out

nb_all_choix_pc: entier in

nombre_max: entier in

FinTQ

valide_joueur <-- Vrai

nombre_allumettes <-- nombre allumettes – nb_allu_choix_pc

R4: Comment “Vérifier si le joueur a perdu”

Si nombre_allumettes=0 et non gagne_hum **Alors**

gagne_joueur <-- Vrai

Sinon

Afficher les allumettes restant

nombre_allumettes: entier in

FinSi

R4: Comment “Déterminer le nombre maximal d’allumettes qu’on peut choisir”?

Selon nombre_allumettes **Dans**

1..3 => nombre_max <-- nombre_allumettes

3..13 => nombre_max <-- 3

FinSelon

R5: Comment “Demander à l'humain le nombre d’allumette qu’il veut prendre”

Ecrire(‘Combien d'allumettes prenez-vous ?’)

Lire(nb_allu_choix_hum)

R5: Comment “Jouer selon le mode choisi”

Selon mode_de_jeu **Dans**

‘n’ | ‘N’ => Choisir un nombre d’allumette aléatoirement entre 1 et nombre_max

‘d’ | ‘D’ => Choisir un nombre d’allumette aléatoirement entre 1 et 3

‘r’ | ‘R’ => nb_allu_choix_pc <-- nombre_max

Autre =>Choisir Le nombre d’allumette d’une manière calculée

nb_allu_choix_pc : entier out

nombre_Allumettes : entier in

nombre _max : entier in

FinSelon

R5: Comment “Vérifier si le nombre d’allumettes prisent par l’humain convient”

Si nb_allu_choix_hum<1 **Alors**

Put("Arbitre : Il faut prendre au moins une allumette.")

SinonSi nb_allu_choix_hum>3

Ecrire("Arbitre : Il est interdit de prendre plus de 3 allumettes.")

SinonSi nb_allu_choix_hum>nombre_max and nombre_max=2

Ecrire("Arbitre : Il reste seulement 2 allumettes.")

SinonSi nb_allu_choix_hum>nombre_max and nombre_max=1

Ecrire("Arbitre : Il reste une seule allumette.")

Sinon

valide_hum <-- Faux

Fin Si

R5: Comment “Vérifier si le nombre d’allumettes prisent par le joueur convient”

Si nb_allu_choix_pc<1 **Alors**

Put("Arbitre : Il faut prendre au moins une allumette.")

SinonSi nb_allu_choix_pc>3

Ecrire("Arbitre : Il est interdit de prendre plus de 3 allumettes.")

SinonSi nb_allu_choix_pc>nombre_max and nombre_max=2

Ecrire("Arbitre : Il reste seulement 2 allumettes.");

SinonSi nb_allu_choix_pc>nombre_max and nombre_max=1

Ecrire("Arbitre : Il reste une seule allumette.")

Sinon

valide_joueur <-- Faux

Fin Si

R5: Comment “Afficher les allumettes restant”

Pour j **De** 1 **A** 3 **Faire**

Pour i **De** 1 **A** nombre_allumettes **Faire**

Ecrire(‘|’)

FinPour

FinPour

R6: Comment “Choisir Le nombre d’allumette d’une manière calculée”

Si nombre_allumettes mod 4 = 0 **Alors**

Choisir un nombre d’allumette aléatoirement entre 1 et nombre_max

Sinon

nb_allu_choix_pc <-- nombre_allumettes mod 4

FinSi

Évaluation des raffinages par l’étudiant

		Evaluation	**Justification**	**Evaluation**

		Etudiant **(I/P/A/+)**	**/ commentaire**	**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	P		
	Une seule structure de contrôle 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	P		
	Le raffinage d'une action décrit complètement cette action	P		
	Le raffinage d'une action ne décrit que cette action	A		
	Les flots de données sont cohérents	A		
	Pas de structure de contrôle déguisée			

		P		
	Qualité des actions complexes	P		

Remarques diverses.

Évaluation 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)**
	A	Le programme ne doit pas contenir d'erreurs de compilation.	
	A	Le programme doit compiler sans messages d'avertissement.	
	A	Le code doit être bien indenté.	

	A	Les règles de programmation du cours doivent être respectées : toujours un Sinon pour un Si, pas de sortie au milieu d'une répétition...	
	P	Pas de code redondant.	
	A	On doit utiliser les structures de contrôle adaptées (Si/Selon/TantQue/Répéter/Pour)	
	A	Utiliser des constantes nommées plutôt que des constantes littérales.	
	A	Les raffinages doivent être respectés dans le programme.	

	A	Les actions complexes doivent apparaître sous forme de commentaires placés AVANT les instructions correspondantes, avec la même indentation	
	A	Une ligne blanche doit séparer les principales actions complexes	
	A	Le rôle des variables doit être explicité à leur déclaration (commentaire).	