Le jeu des multiples

Description du jeu

A partir d'un nombre de départ égal à 2, ce petit jeu consiste à saisir une liste de multiples respectant la contrainte suivante : chaque multiple saisi doit être un multiple supérieur à la valeur précédente (ou au nombre de départ qui est égal à 2 pour la première saisie).

Au début de la partie le joueur définit un niveau de difficulté en choisissant le nombre de multiples à fournir pour gagner la partie.

Spécification du besoin

Le jeu s'excécute dans un terminal. Le déroulement nominal est le suivant :

- Au début du jeu, le programme demande au joueur de saisir un degré de difficulté en choisissant le nombre de multiples à fournir pour gagner la partie.
- Le programme vérifie que le nombre saisi par l'utilisateur est supérieur à 1 afin que la partie soit au moins composée d'une saisie de la part du joueur. Si ce n'est pas le cas, le programme affiche un message d'erreur et invite le joueur à saisir une nouvelle valeur jusqu'à ce que le nombre de bonnes réponses pour gagner soit supérieur ou égal à 1. **Remarque** : on ne traitera pas d'autres cas d'erreur que celui-ci.

Ensuite la partie commence…

- Le programme demande au joueur de donner un multiple de 2 supérieur à 2.
- Le joueur propose un mutliple au choix (6 par exemple).
- Pour chaque proposition faite par le joueur, le programme affiche un retour du type *"Bonne réponse"* ou *"Mauvaise réponse"*
- Si la réponse du joueur est correcte (il s'agit d'un multiple supérieur à la valeur précédente), le programme félicite le joueur et lui indique le nombre de bonnes réponses restantes pour atteindre la victoire.
- Si la réponse du joueur est incorrecte, le jeu s'arrête. Le programme indique au joueur qu'il a perdu ainsi que le nombre de bonnes réponses qu'il a pu fournir.

Scénario nominal : Le joueur gagne la partie

Ce scénario décrit le déroulement du jeu lorsque le joueur gagne la partie. Dans cet exemple, le joueur décide de gagner s'il fournit 3 multiples corrects.

- il commence par fournir la valeur *4* qui est bien une valeur supérieure *et* multiple de 2;
- il fournit ensuite la valeur *16* qui est bien une valeur supérieure *et* multiple de 4;
- il fournit enfin la valeur *32* qui est également une valeur supérieure *et* multiple de 16.

Le joueur gagne donc la partie car il a été capable de donner 3 fois de suite un multiple supérieur à la valeur précédente.

Scénario alternatif : Le joueur perd car sa valeur n'est pas un multiple de la valeur précédente

Ce scénario décrit le déroulement du jeu lorsque le joueur perd la partie car la valeur qu'il a donné n'est pas multiple de la précédente.

Dans cet exemple, le joueur décide de gagner s'il fournit 4 multiples corrects.

- il commence par fournir la valeur *6* qui est bien une valeur supérieure *et* multiple de 2;
- il fournit ensuite la valeur *12* qui est bien une valeur supérieure *et* multiple de 6;
- mais il fournit ensuite la valeur *15* qui est bien une valeur supérieure à 12 mais qui n'est pas un multiple de 12.

Il perd dont la partie avec un score égal à 2 car le joueur n'a fourni que 2 multiples corrects.

Scénario alternatif : Le joueur perd car sa valeur n'est pas supérieure à la valeur précédente

Ce scénario décrit le déroulement du jeu lorsque le joueur perd la partie car la valeur qu'il a donné n'est pas supérieure à la précédente.

Dans cet exemple, le joueur décide de gagner s'il fournit 3 multiples corrects.

- il commence par fournir la valeur *4* qui est bien une valeur supérieure *et* multiple de 2;
- il fournit ensuite à nouveau la valeur *4* qui est bien un multiple de 4 mais qui n'est pas supérieur à 4.

Le joueur perd dont la partie avec un score égal à 1 car il n'a fourni qu'un seul multiple correct.

Scénario alternatif : Le joueur perd car sa valeur n'est ni multiple ni supérieure à la valeur précédente

Ce scénario décrit le déroulement du jeu lorsque le joueur perd la partie car la valeur qu'il a donné n'est pas supérieure à la précédente et n'est pas non plus un multiple de la valeur précédente.

Dans cet exemple, le joueur décide de gagner s'il fournit 3 multiples corrects.

- il commence par fournir la valeur *4* qui est bien une valeur supérieure *et* multiple de 2;
- il fournit ensuite la valeur *3* qui n'est ni multiple de 4 ni supérieure à 4.

Le joueur perd dont la partie avec un score égal à 1 car il n'a fourni qu'un seul multiple correct.

Scénario d'exception : Le joueur saisit une difficulté de jeu incorrecte

Ce scénario décrit le comportement du jeu lorsque le joueur choisit un niveau de difficulté incorrect. On juge qu'un niveau de difficulté doit au moins être égal à 1 pour lancer une partie.

Dans cet exemple, le joueur décide de gagner s'il fournit 3 multiples corrects.

- le joueur tente d'abord de saisir un niveau de difficulté négatif pour tester le programme : Le programme affiche un message d'erreur indiquant que le niveau de difficulté est incorrect ;
- le joueur tente ensuite de fournir un niveau de difficulté pour voir s'il peut gagner la partie sans saisir aucun multiple : Le programme affiche à nouveau un message d'erreur indiquant que le niveau de difficulté est incorrect.

Important : le programme ne gèrera pas d'autres cas d'erreur : Le cas où l'utilisateur saisit une chaîne de caractères par exemple ne sera pas pris en compte…

Ressources à disposition

Schéma de solution pour déterminer si un nombre A est multiple d'un nombre B

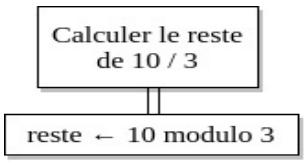
Pour déterminer si un nombre A est multiple d'un nombre B, il faut vérifier que B divise A. B divise A si le reste de la division euclidienne B/A égale 0.

Le reste d'une division euclidienne se nomme le modulo. La plupart des langages de programmation propose un opérateur modulo pour calculer le reste de la division euclidienne de deux nombres entiers.

En C++, l'opérateur % permet de calculer le modulo de deux nombres entiers. Ainsi :

- l'instruction `cout << 10 % 2;` affichera 0 (car 10 / 2 = 5 avec un reste égal à 0) ;
- l'instruction `cout << 10 % 3;` affichera 1 (car 10 / 3 = 3 avec un reste égal à 1) ;
- l'instruction `cout << 10 % 4;` affichera 2 (car 10 / 4 = 2 avec un reste égal à 2).

D'un point de vue algorithmique, l'opérateur sera simplement noté "modulo". A titre d'exemple, le calcul du modulo pour la division entière 10/3 sera formalisé de la façon suivante :



Utilisation de l'opérateur modulo dans un algorithme

Des fonctionnalités pour agrémenter votre jeu

Bien que le jeu des multiples n'en ait pas besoin vous pouvez utiliser (si vous le souhaitez) les fonctionnalités proposées dans le module `game-tools` :

https://github.com/patrick-etcheverry/game-tools.

Ce module met à disposition des fonctionnalités simples permettant d'effacer le terminal, de mettre le programme en pause, d'afficher des éléments en couleur ou encore de générer un nombre entier aléatoire.

Attention : Rappelez-vous que votre production doit respecter les spécifications fournies dans la section *Spécification du besoin*. Les fonctionnalités de la librairie `game-tools` doivent être utilisées avec **parcimonie** :

- soit pour répondre à un besoin clairement spécifié dans la section *Spécification du besoin* et les scénarios décrits;
- soit pour agrémenter l'interface du jeu, sans toutefois dénaturer à l'excès le comportement attendu et décrit dans les différents scénarios.

En cas de doute, n'hésitez pas à en discuter avec votre enseignant.

Extensions possibles

Ces extensions, si elles sont modélisées (algorithme) et programmées correctement, peuvent rapporter des points bonus. Vous n'êtes pas obligé de les traiter et si vous décidez de les traiter, vous pouvez choisir lesquelles vous souhaitez traiter.

Attention : Ces extensions sont **secondaires** et ne doivent pas prendre le pas sur les spécifications fournies. L'élaboration d'une extension ne pourra en aucun cas compenser l'absence d'une fonctionnalité demandée dans la *Spécification du besoin* ou les différents scénarios décrits.

- **Extension 1 :** Dans la version initiale du jeu, le programme commence toujours par demander un multiple de 2. Pour corser un peu le jeu, la proposition est de faire en sorte que cette valeur de départ (2) soit remplacée par une valeur aléatoire tirée dans l'intervalle [6..29].
- **Extension 2 :** Dans la version initiale du jeu, lorsque le joueur perd, le programme reste flou sur les raisons de cet échec. Cette extension propose de fournir un message plus précis au joueur. Ainsi, lorsqu'un joueur perd il s'agira, à la fin du jeu, de lui indiquer s'il a perdu parce que :
    - sa réponse est supérieure à la valeur précédente mais n'est pas un multiple ;
    - sa réponse est un multiple de la valeur précédente mais n'est pas supérieure ;
    - sa réponse n'est ni supérieure ni multiple de la valeur précédente.