Rappels des notions de programmation

Notion de test de condition

# Introduction

On souhaite réaliser un jeu qui permet à une personne de rechercher un nombre entier aléatoire qu’a choisi l’ordinateur.

On peut se baser sur le même principe que le jeu du pendu qui consiste à partir avec un nombre de points donnés et de soustraire des points à chaque mauvaise réponse.

# Test de condition (algorithme)

Pour tester une condition dans un algorithme :

SI condition VRAIE ALORS

Liste de commandes à exécuter si test vrai

SINON (condition fausse)

Liste de commandes à exécuter si test faux

FIN SI

Ton algorithme est alors le suivant :

DEBUT

Déclarer la variable nombre\_a\_trouver

Déclarer la variable nombre\_joueur

Nombre\_a\_trouver <- nombre aléatoire (compris entre 1 et 50)

REPETER TOUJOURS

Demander nombre à l’utilisateur

Nombre\_joueur <- reponse tapée

SI nombre\_joueur = ……… ALORS

Afficher (dire) « nombre trouvé »

SINON

Afficher (dire) « ………………… »

FIN SI

FIN REPETER

FIN

# Conversion de l’algorithme en programme (codes)

Créer les objets nécessaires

Rechercher les événements déclenchants

Pour chaque événement, rechercher l’algorithme. Ici un seul événement et donc un seul algorithme.

Convertir son algorithme en programme

  

Effectuer des tests à partir d’un jeu de tests aux bornes du domaine de fonctionnement du programme.

Valider sa solution.

# Ajout d’aide

Modifie ton algorithme pour donner de l’aide en informant ton joueur si le nombre du joueur est plus grand ou plus petit que le nombre choisi par l’ordinateur.

Rédige to algorithme et modifie ton programme.

Recherche une stratégie pour trouver le nombre de l’ordinateur en un minimum de coups.

# Ajout de points

Ajoute une gestion de points pour démarrer d’un certain nombre de points au départ. A chaque erreur, le nombre de points diminue.

On peut aussi ajouter un changement d’image à chaque mauvaise réponse.