Rapport de projet

Génération de Labyrinthe en MIPS

Léo PIVETEAU–WERNERT et Jaenai RUGENGANDE IHIMBAZWE

UFR MATH-INFO
UNIVERSITE DE STRASBOURG

Etat d'avancement

Les fonctions ont été écrites dans l'ordre des sections données. Nous écrivions d'abord les fonctions d'une section avant de passer à la section suivante puis chaque fonction était testée avant de la commit. Nous avons d'abord écrit les fonctions proposées. Ensuite nous avons rajouter des fonctions lors de l'implementation de l'algorithme. Enfin nous avons écrit des fonctions pour pouvoir gérer les arguments données lors de la lecture des arguments dans le terminal.

Lors de l'implementation de l'algorithme, nous avons rajouté les fonctions marquer_cell_visite, marquer_cell_debut, marquer_cell_fin et casser_mur_entre_cell. Ces derniers representent un certain nombre des actions à effectuer dans l'algorithme. Ils nous permettaient d'implementer l'algorithme plus facilement. Sans ces fonctions, on aurait moins de lisibilité pour la génération du labyrinthe. Parce que l'on doit mettre le corps de code de chacune de ces fonctions dans la même fonction, ce qui donne un code trop long et dans lequel on va se perdre lors de la correction des erreurs. La division en différentes fonctions suit également un principe utilisé souvent en informatique "divide and conquer". On divise le problème en plusieurs problèmes plus petits que nous pouvons résoudre.

Nous avions également besoin des fonctions longueur_str et str_int. Lorsque nous prenons un argument lors de l'execution du code, c'est une chaine de caractères. Or nous voulons un entier pour pouvoir generer le labyrinthe. Ainsi il nous faut transformer cet argument en entier. Avec les fonctions ajoutées, nous pouvons faire cela.

Difficultés rencontrées

Nous avons rencontré une difficulté majeur lors de l'implementation de l'algorithme. Les fonctions provenant des sections données que nous avions écrit ne marchaient pas correctement. Cependant lors des tests individuelles effectués, tout marchait bien.

Nous n'avions pas testés plusieurs fonctions à la fois avant et nous avions fait l'erreur d'utiliser trop de variables globales. Cela était un problème lors de l'appel de plusieurs fonctions à la fois surtout si ils utilisaient les mêmes variables globales. Les variables pouvaient etre perdus ou changés sans le vouloir. Nous avons du modifier cela avant d'implementer l'algorithme.

Nous avons eu également un problème mineur d'incrémentation incorrecte de la pile mais cela était assez vite résolu.

Choix d'implementations et des structures de données

Il nous a été imposé d'utiliser une pile comme structure de données pour sauvegarder le labyrinthe. Nous supposons que ce choix était fait pour nous permettre de sauvegarder le labyrinthe de la fin au début mais de pouvoir l'imprmer dans le bon ordre.

La pile devait être alloué sur le tas comme un tableau dynamique. Ce choix était imposé mais la raison pour cela est compréhensible. La pile sp aurait pu être également utilisé mais il y aurait eu une difficulté à gérer les arguments et variables locales en même temps que le labyrinthe. On aurait pas pu utiliser un tableau statique car on doit pouvoir générer un labyrinthe de n'importe quelle taille donnée.