RAPPORT

Automate Cellulaire

Langage : C

Par :

Mahafaly Randriamiarisoa

Et

Robert Hansana

Structures mis en place :

struct cellule
int state // l'état de la cellule

struct Rule
char* rule // règle représenté par une chaine de caractère, l'expression régulière qui correspond et la suivante : *[0-3]{10}|[01]{8}*
cell (*getNextCell)(cell c,cell* voisinage, char* rule) //cette fonction permettra d'appliquer la regle dans un contexte.

struct automate
int MAX_VALUE // cette valeur représente la valeur maximale d'un état de cellule
int FIRST_VALUE // valeur de la toute première valeur
int WIDTH // nombre de colonne qu'occupera l'automate cellulaire
int HEIGHT // nombre de ligne qu'occupera l'automate cellulaire
int POSITION_FIRST_VALUE // position de la première valeur
_Rule _rule // il s'agit d'un attribut de type struct Rule*, il contiendra la fonction permettant aux cellules de l'automate d'évoluer.

Indications :

Afin de rendre l'affichage console plus agréable j'ai fait appel à une Library nommée Termcap, or lors de l'initialisation de ses fonctionnalités il s'avère qu'elle alloue de la mémoire qui n'est pas atteignable pour la libération, je me suis renseigné sur Internet et il n'y a pas de moyen connu à ce jour pour libérer les 15,300 bytes.

```
==8670== HEAP SUMMARY:
==8670==     in use at exit: 15,300 bytes in 16 blocks
==8670==   total heap usage: 1,528 allocs, 1,512 frees, 17,108,485 bytes allocated
==8670==
==8670== LEAK SUMMARY:
==8670==    definitely lost: 0 bytes in 0 blocks
==8670==    indirectly lost: 0 bytes in 0 blocks
==8670==      possibly lost: 0 bytes in 0 blocks
==8670==    still reachable: 15,300 bytes in 16 blocks
==8670==         suppressed: 0 bytes in 0 blocks
==8670== Reachable blocks (those to which a pointer was found) are not shown.
==8670== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==8670==
==8670== For counts of detected and suppressed errors, rerun with: -v
==8670== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
# valgrind --leak-check=full --show-leak-kinds=all ./out/exe
# valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./out/exe
```

Pour la première règle :

1	1	1	1	1	0	1	0	1	1	0	0	0	1	1	0	1	0	0	0	1	0	0	0

on convertira les états des voisinages de cellule en binaire (méthode indiquée dans l'énoncé) vers un entier de base 10

7	6	5	4	3	2	1	0

on fera une correspondance avec la règle fournit qu'on inversera:

0	0	0	1	1	1	1	0

0	1	1	1	1	0	0	0

Pour la deuxième règle :

on récupère le voisinage de la cellule et nous effectuons la somme des états(valeur_état_MAX = 3),

on fait ensuite correspondre cette somme à un tableau de 10 indices, contenant à chaque case une valeur comprise entre 0 et 3 qui sera la valeur de la cellule suivante.

0	1	2	3	4	5	6	7	8	9
0	0	0	2	3	0	2	1	0	0

pour rendre cette règle plus modulable il est possible d'attribuer n'importe quel état à chaque cellule du moment que la taille de la règle est égale à (*nombre_de_voisin * valeur_état_MAX*),

mais je n'ai pas eu le temps d'implémenter cette fonctionnalité.

La syntaxe du fichier de configuration, "conf" avec les correspondances des expressions régulières :

*** INIT_FILE ***

RULE: `[0-3]{10}|[01]{8}`

MAX_VALUE: `\d+`

FIRST_VALUE: `\d+`

WIDTH: `\d+`

HEIGHT: `\d+`

POSITION_FIRST_VALUE: `(LEFT|CENTER|RIGHT|\d+)`

Le programme indiquera une erreur si une des valeurs est négative.

Makefile :

Il est conseillé de procéder dans cet ordre: *make clean; make;*

À partir d'ici s'offre plusieurs options,

pour un affichage standard : *make exe*

pour un affichage "agrèable" : *make fexe*

pour créer l'image pgm : *make pgm*

pour du bonus : *make bonus (c-c pour arrêter)*

pour effacer les images pgm : *make cleanPgm*