Projet langage C
Conway's Game of Life

28 novembre 2012

Sujet : réaliser en `C`, et en mode texte, une implémentation de l'automate cellulaire imaginé par John Horton Conway.

Objectifs :
- analyser un cahier des charges ;
- traduire l'analyse avec un souci de modularité ;
- élaborer une documentation.

1 Présentation du jeu de la vie de *John Horton Conway*

1.1 Principe du jeu

Le jeu de la vie est un automate cellulaire imaginé par John Horton Conway en 1970.

Le principe est simple. Sur une grille théoriquement infinie (appelée plan de Conway), les cases, qui représentent des cellules, sont soit vivantes, soit mortes. À chaque étape, l'évolution d'une cellule est calculée en fonction de l'état de ses huit voisines de la façon suivante :
- une cellule morte avec exactement 3 voisins vivants devient vivante (naissance liée à un environnement optimal) ;
- une cellule vivante avec 2 ou 3 voisins vivants reste vivante (elle n'est ni isolée ni étouffée), sinon elle meurt (mort par désertification ou surpopulation).

C'est l'analogie entre ces règles et certains critères d'évolution de populations de bactéries qui a conduit à donner à cet automate le nom de jeu de la vie.

1.2 Explication par l'exemple

La séquence des quatre tableaux ci-dessous illustre un exemple d'évolution. Le premier tableau montre la population de départ, les cellules vivantes y sont représentées par des cases avec un `+`, et les mortes par des cases vides. Le deuxième tableau donne, pour chacune des cellules du plan précédent le dénombrement des voisins vivants. Le troisième tableau indique, pour chacune des cellules, s'il y a naissance (`n`), mort (`m`), conservation de la cellule vivante (`c`), ou conservation de la cellule morte (case vide). Le dernier tableau donne enfin le nouvel état de la population du premier tableau après une évolution.

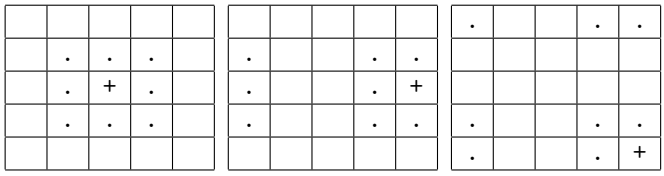
		+		
	+	+	+	
	+			

0	1	1	1	0
1	3	3	3	1
2	3	4	3	1
2	2	4	2	1
1	1	1	0	0

	n	c	n	
	c	m	c	
	c			

	+	+	+	
	+		+	
	+			

Comme nous ne pouvons avoir un plan infini, nous utiliserons un plan torique : la ligne du haut est voisine de la ligne du bas, et la colonne de gauche est voisine de la colonne de droite. Les trois tableaux ci-dessous présentent trois cellules vivantes (matérialisées par un `+`) et leur huit voisines associées (matérialisées par des `.`) sur un plan de Conway de 5 par 5.



L'élaboration d'un tel projet nécessitera l'utilisation d'une bibliothèque de fonctions d'affichage avancées en mode texte. Il s'agit de la librairie *Ncurses*. Elle permet, par exemple, de pouvoir afficher un caractère aux coordonnées x et y du terminal.

2 Cahier des charges

Lorsque l'utilisateur lance le jeu, il se voit offert, par l'intermédiare du **menu principal**, les alternatives suivantes :

1. Lancer le jeu de la vie ;
2. Générer une population aléatoire ;
3. Population par défaut ;
4. Charger une population ;
5. Sauver la population courante ;
6. Aide ;
7. Quitter.

Lancer le jeu de la vie

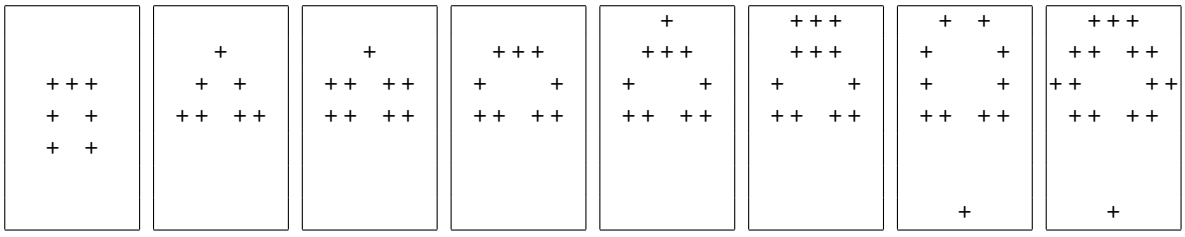
Cette action permet de lancer le jeu de la vie avec la population courante. Lorsque le jeu est lancé, il passe d'un état au suivant tout seul après un certain délai.

La population par défaut du jeu de la vie est la suivante :

```
+++
+ +
+ +
```

Cette population est centrée sur un plan de dimension 15 par 40.

Voici comment évolue une telle population sur un plan de dimension 7 par 7 :



Cet exemple illustre bien l'aspect torique de notre plan comme le montre la cellule qui naît en bas du plan.

Quand le jeu de la vie est lancé, l'application propose un autre menu (**menu secondaire**) à l'utilisateur :

1. Accélérer le déroulement du jeu
2. Ralentir le déroulement du jeu
3. Arrêter le déroulement du jeu

Pour ralentir l'évolution du jeu de la vie, il suffit d'insérer une temporisation dans la boucle principale du jeu. Ralentir ou accélérer le jeu revient donc à augmenter ou à diminuer la durée de la temporisation. En `C`, la fonction `void usleep(int micro_seconds)` permet de réaliser une temporisation en précisant en paramètre un nombre de micro-seconde. Pour utiliser cette fonction, il suffit d'ajouter la directive de compilation suivante : `#include <unistd.h>`.

L'action « Arrêter le déroulement du jeu » permet de figer le jeu dans l'état dans lequel il se trouve et de retourner au menu principal.

Générer une population aléatoire

Permet de générer une population aléatoire sur le plan de Conway. Chaque case se voit affecter un état soit vivant, soit mort avec une probabilité équivalente pour les deux états.

Population par défaut

Permet de restaurer la population par défaut.

Charger une population

Permet de charger une population depuis un fichier texte.

Sauver la population courante

Permet de sauver une population dans un fichier texte.

Aide

Permet d'afficher une aide qui explique, entre autres, le fonctionnement du jeu de la vie.

Quitter

Permet de quitter l'application.

3 Librairie Ncurses

3.1 Présentation générale

Ncurses est une bibliothèque qui offre des fonctionnalités de manipulation de l'écran. Le principe de fonctionnement de cette librairie est d'assimiler le terminal à un tableau de caractères stocké dans une structure nommée `WINDOW`. Afin d'optimiser au maximum les programmes, tous les affichages sont bufferisés (*i.e.* mémorisés avant d'être effectifs). Un changement n'est actif que lorsque la commande `refresh()` est appelée. En attendant que cette commande soit appelée, les changements sont opérés dans l'image mémoire de la fenêtre (`WINDOW`).

Avant de pouvoir utiliser les fonctions de la bibliothèque Ncurses, il faut l'initialiser, c'est-à-dire la renseigner sur le type de terminal et les caractéristiques qui lui sont associés et allouer la mémoire nécessaire. Tout cela se fait avec la fonction `initscr()`.

Son pendant, la fonction `endwin()` permet de libérer la mémoire occupée par Ncurses et de restaurer le terminal dans son état initial. Cette instruction met fin à toutes les modifications opérées par Ncurses lors de son fonctionnement.

Ces deux instructions doivent donc toujours être appelées, respectivement au début et à la fin, dans un programme qui utilise Ncurses !

Lorsque l'on utilise la librairie Ncurses, tout ce qui est affiché à l'écran doit l'être par l'intermédiaire de cette librairie. Il n'est pas possible d'avoir simultanément dans un terminal des caractères affichés de manières standard et des caractères affichés en utilisant le mode Ncurses. En d'autres termes, il n'est pas possible de commencer à afficher un menu dans le terminal en utilisant des `printf` puis de décider d'afficher quelque chose au milieu de l'écran en passant en mode Ncurses et en utilisant un `printw`. En effet, le passage au mode Ncurses initialise le terminal, ce qui efface le menu précédemment affiché.

3.2 Programme minimal : Hello world !

Voici un programme minimal qui affiche « Hello world ! » au centre du terminal et attend que l'utilisateur appuie sur la touche q :

```
#include <ncurses.h>
int main(void) {
  int x,y;
  char message[]="Hello world! (q pour quitter)";
  initscr();       // Initialise le terminal en mode Ncurses
  y=LINES/2;       // Pour afficher le message sur la ligne du milieu du terminal
  x=(COLS-strlen(message))/2; // Pour centrer le message sur la ligne
  move(y,x);       // Positionnement du curseur
  printw(message); // Ecriture du message dans le terminal virtuel
  refresh();       // Affichage du terminal virtuel dans le terminal réel
  nodelay(stdscr,TRUE); // Pour un getch() non bloquant
  noecho();        // Les caractères tapés ne sont pas affichés
  while(getch()!='q'); // Attente de la pression de la touche 'q'
  endwin();        // Met fin au mode Ncurses
  return 0;
}
```

Pour utiliser la librairie Ncurses, il faut l'inclure : `#include <ncurses.h>`. Lors de l'édition des liens, il faut ajouter l'option `-lncurses`. Si le programme ci-dessus est sauvé dans le fichier `hello.c`, la génération de son exécutable s'effectue donc de la façon suivante :

```
gcc -c hello.c -o hello.o
gcc -o hello -lncurses hello.o
```

Les constantes `LINES` et `COLS` contiennent le nombre de lignes et de colonnes du terminal. Les différentes fonctions utilisées sont explicitées dans la section qui suit.

3.3 Quelques fonctions d'affichage de Ncurses

Pensez à utiliser les pages du manuel (`man`) pour avoir une information détaillée sur une fonction de la librairie.

`WINDOW *initscr(void)`

Cette fonction initialise le terminal en mode Ncurses.

`int endwin(void)`

Cette fonction permet de mettre fin au mode Ncurses. Le terminal risque d'avoir un comportement étrange si cette fonction n'est pas appelée avant que le programme ne prenne fin.

`int refresh(void)`

Cette fonction permet aux modifications, généralement effectuées sur une fenêtre virtuelle en mémoire, d'être effectives sur l'écran.

`int clear(void)`

Cette fonction efface l'écran.

`int addstr(const char *str)`

Cette fonction affiche la chaîne de caractères `str` à la position courante du curseur. Il est important de faire attention à ce que la chaîne de dépasse pas la fin de la ligne.

`int printw(const char *fmt, ...)`

Cette fonction est l'équivalent dans Ncurses de la fonction `printf` pour l'affichage formaté des chaînes de caractères.

`int move(int y, int x)`

Cette fonction permet de positionner le curseur aux coordonnées (x, y) du terminal. Attention, l'origine de la fenêtre, $(0, 0)$, est le coin supérieur gauche.

`int getch(void)`

Cette fonction permet de lire un caractère dans le tampon. Dans le mode *nodelay*, si aucun caractère n'est disponible, la valeur `ERR` est retournée. Dans le mode *delay*, le programme attend que le systeme rende un caractère disponible dans le tampon. Dans le mode *cbreak*, un caractère est disponible dès qu'un caractère est saisi au clavier. Dans le mode *nocbreak*, un ou plusieurs caractères sont disponibles quand l'utilisateur saisit un retour chariot.

`int nodelay(WINDOW *win, bool bf)`

L'option *nodelay* activée (*i.e.* `bf` est `TRUE`) fait de `getch` un appel non bloquant. Si aucune entrée n'est prête, `getch` retourne `ERR`. Si *nodelay* est désactivée (`bf` est `FALSE`), `getch` attend qu'un caractère soit disponible dans le tampon. La fenêtre standard est `stdscr`. L'instruction `nodelay(stdscr,TRUE)` permet donc de passer en mode *nodelay*.

`int cbreak(void)` **et** `int nocbreak(void)`

La fonction `cbreak` rend les caractères saisis par l'utilisateur immédiatement disponibles au programme. La fonction `nocbreak` remet le terminal en mode normal, c'est-à-dire que le pilote *tty* met en tampon les caractères saisis jusqu'à ce qu'un retour chariot soit saisi.

`int echo(void)` **et** `int noecho(void)`

Les fonctions `echo` et `noecho` contrôlent si les caractères saisis par l'utilisateur sont affichés par `getch` quand ils sont saisis. L'affichage par le pilote *tty* est toujours désactivé, mais initialement `getch` est en mode *echo*, alors les caractères saisis sont affichés.

4 Déroulement du projet

4.1 Que rendre ?

- Un guide d'utilisation du logiciel.
- Un descriptif technique : organisation générale, choix des structures de données (les variables et leur type, les structures, etc.), découpage en fonctions, etc.
- Une analyse succincte expliquant ces choix techniques.
- Un listing du programme (commenté et indenté avec une entête pour le programme principal et une entête pour chaque fonction).
- Un exécutable sous Linux
- Le projet doit être rendu par binôme

4.2 Comment s'y prendre ?

Le projet est à faire en binôme.

Dans un premier temps, élaborez les structures de données et testez un programme (sous forme d'un projet C) implémentant l'automate du jeu de la vie sur un plan de dimension 15 par 40 (cf. section 2).

Vous devrez ensuite étendre le programme précédent en y incorporant la gestion des menus, puis, une à une, toutes ses fonctionnalités.

5 Pour ceux qui veulent aller plus loin (travail non noté)

5.1 Plan de Conway dynamique

Le tableau permettant de mémoriser le plan de Conway peut être dynamique (pointeur sur pointeurs) de manière à pouvoir représenter un plan de taille quelconque. Une telle représentation conduit également à des fonctions plus générales (*i.e.* non dédiées à un tableau de taille prédéfinie).

5.2 La librairie Ncurses

La librairie Ncurses permet de faire de nombreuses choses comme :
- gérer des fenêtres ;
- utiliser des couleurs ;
- mettre en place des menus déroulants ;
- etc.

Vous trouverez plus d'informations sur la librairie Ncurses sur Internet. Voici quelques sites intéressants :
- `http://www.tldp.org/HOWTO/NCURSES-Programming-HOWTO/`
- `http://www.apmaths.uwo.ca/~xli/ncurses.html`
- `http://perso.club-internet.fr/ariffart/ncurses/ncurses01.html`

Attention, le dernier site mentionné, bien que très pédagogique, comporte des coquilles. Par exemple, pour ajouter une chaîne, il faut utiliser la fonction `addstr` et non pas `addchstr`.

5.3 Le jeu de la vie

Malgré sa simplicité, ce jeu est une machine de Turing universelle : il est possible de calculer tout algorithme pourvu que la grille soit suffisamment grande et les conditions initiales correctes.

Le principal intérêt de ce jeu est qu'il permet, à partir de règles simples, de faire émerger des phénomènes ou des concepts complexes comme :
- les configurations dites de Jardin d'Eden, qui ne peuvent pas avoir de prédécesseur ;
- les glisseurs (une figure mobile en diagonale qui apparaît assez spontanément dans les configurations du jeu) ;
- les canons à glisseurs ;
- la remontée du temps (quitte à imaginer plusieurs passés possibles).

Liens intéressants :
- `http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScientificAmerican.htm`
- `http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life`