Projet de transition

Langage C – Techniques de programmation en C

Il s'agit de produire une application (un exécutable testé et opérationnel) en situation de projet au sein d'une équipe. Le sujet est présenté dans la suite du document. Le code à produire doit être du C ANSI. La présentation de l'application se fait sous environnement Linux.

Dans ce projet vous devrez respecter strictement certaines contraintes. En revanche vous aurez à choisir et spécifier vous-même le thème.

Objectifs du projet dans le cursus ITII

Approche « Programmation »

- Il s'agit de valider les compétences de la première partie du cours de programmation par un programme de 500 à 1 000 lignes de code en langage C, faisant un tout cohérent.
- Le projet vous permet de faire du développement autrement qu'en temps limité et sur des petits exercices de validation.
- La notion d'application vous demande un travail depuis les spécifications jusqu'aux tests et à la notion d'exécutable opérationnel,
- Le but principal est donc de rendre un ensemble contenant l'exécutable testé et opérationnel en environnement Linux, un document de conception, une documentation d'installation et d'utilisation.
- L'attention doit être mise sur le code : présentation (indentation, nommage des variables, commentaires), l'architecture : les fonctions (dédiées à un usage, d'environ 20 lignes maximum avec variables locales et passage de paramètres).

Approche « Ingénieur »

- Techniquement vous allez aborder les aspects spécification et architecture, vous préoccuper du cycle de développement complet.
- Enfin il s'agit de « livrer » un produit fini, testé et opérationnel et d'intégrer complètement cette démarche.

Exigences Organisationnelles

Les étudiants travaillent en binôme, à titre exceptionnel (dérogation) seuls.

Rôle au sein de l'équipe

Rôle technique

1. Rôle concepteur (expression des besoins, définition des fonctionnalités)
2. Rôle architecte technique (architecture du programme, spécification des fonctions, recette du code C, assemblage des parties, tests d'intégration)
3. Rôle développeur (production du code en C, développement suivant spécifications, tests unitaires)

Rôles gestion de projet

1. Rôle communication (présentation : préparation, animation)
2. Rôle qualité (plan de tests, définition des objectifs, validation des objectifs, mise au point des jeux de tests et validation des tests)
3. Rôle suivi du projet (gestion de l'avancement du projet)

Dans tous les cas, ceux qui découvrent la programmation doivent impérativement coder (rôle développeur).

Délivrables et évaluation

Evaluation du projet
Chacun rend : une note sur son rôle dans l'équipe, son travail technique personnel, son travail en gestion de projet. Chaque étudiant peut être interrogé individuellement sur des aspects techniques (conception et développement).

La présentation du projet se déroule devant les autres élèves. LE temps imparti à la présentation vous sera précisé par la suite. Il s'agit de 10 à 15 minutes. Au cours de la présentation tous les élèves notent tous les projets (sauf le leur).

La note attribuée est fonction de la qualité du code produit, de la démarche adoptée, des outils utilisés, mais aussi de la qualité des documents écrits (orthographe et lisibilité). La note tient aussi compte de la présentation du projet et des évaluations de vos collègues.

Le temps de travail est environ évalué à 20 heures dont 10 à 15 heures sur la conception et le développement en C.

Critères d'évaluation
Un projet dont l'exécutable ne fonctionne pas correctement n'a pas la moyenne. Il est donc impératif de viser des objectifs atteignables et de le tester de manière approfondie. La note du projet vient en appréciation de la note du test de mise à niveau.
La note prend en compte les aspects suivants :
- Qualité technique de la réalisation,
- Stabilité et robustesse de l'application (erreurs,…)
- Qualité de la présentation et originalité de la mise en œuvre
- Entretien individuel si nécessaire

Ce qu'il faut rendre (délivrable)
- un document écrit décrivant le thème adressé, les spécifications et la manière (outils, démarche, organisation de l'équipe dont vous avez abordé le projet)
- la note personnelle concernant son travail et son rôle dans l'équipe
- le code
- l'exécutable testé et opérationnel avec sa documentation d'installation

A rendre sur clé USB (à envoyer sur la messagerie) ou tout autre support magnétique le jour de la soutenance.

Calendrier et notation
- Les documents devront être envoyés par messagerie (date précisée ultérieurement).
- La soutenance est prévue JJ MM 2016 sur la base des documents envoyés. La présentation finale est de 15 minutes.
- Suivant les projets, un examen individuel sur les aspects techniques peut se dérouler à la séance suivante.

Présentation du sujet

Il s'agit de mettre en œuvre un automate cellulaire sur une problématique donnée.

Principe : les automates cellulaires sont des programmes faisant évoluer des cellules sur une zone géographique (1D, 2D ou 3D). On utilise communément une grille pour les représenter. Les cellules sont susceptibles de changer d'états et la transition d'un état à un autre dépend de règles de voisinage plus ou moins simples. La simplicité apparente des automates cellulaires permet cependant de modéliser des phénomènes complexes tels que : la propagation de feux de forêts, la propagation de phénomènes biologiques (épidémies), les embouteillages sur les réseaux routiers, les phénomènes de panique de foule.

Les exigences techniques et organisationnelles sont à respecter strictement.

Le thème de l'automate cellulaire est libre.

Présentation du projet

Il s'agit de mettre en oeuvre un automate cellulaire sur une problématique donnée. La problématique est à votre appréciation.

Note et références concernant les automates cellulaires :

Principe : les automates cellulaires sont des programmes faisant évoluer des cellules sur une zone géographique (1D, 2D ou 3D). On utilise communément une grille pour les représenter. Les cellules sont susceptibles de changer d'états et la transition d'un état à un autre dépend de règles de voisinage plus ou moins simples. La simplicité apparente des automates cellulaires permet cependant de modéliser des phénomènes complexes tels que : la propagation de feux de forêts, la propagation de phénomènes biologiques (épidémies), les embouteillages sur les réseaux routiers, les phénomènes de panique de foule.

Exemple d'automate cellulaire avec le jeu de la vie

Les automates de Conway sont un bon exemple :
Les cellules sont les cases d'un tableau. Elles peuvent avoir 2 états : morte, vivante.
Une cellule a 8 voisines, au temps t :

(1) la cellule reste dans l'état vivant si $2 \leq$ nombre_voisines_vivantes ≤ 3
(2) la cellule passe de vivant à mort si nombre_voisines_vivantes < 2 ou nombre_voisines_vivantes > 3
(3) la cellule passe de l'état mort à vivant si nombre_voisines_vivantes $= 3$
(4) la cellule reste dans l'état mort si nombre_voisines_vivantes $\neq 3$

A titre informatif, ci-dessous les liens pour découvrir les automates cellulaires :

- http://fr.wikipedia.org/wiki/Automate_cellulaire
- http://fr.wikipedia.org/wiki/Jeu_de_la_vie

En fonction de votre niveau en programmation vous choisirez de mettre en oeuvre des automates cellulaires dans des situations plus ou moins complexes qui exigeront de vous des compétences plus ou moins élevées.

- Le niveau de base est l'automate dont les transitions des cellules ne dépendent que de l'état de leurs voisins (phénomènes de panique de foule, formation de motifs géométriques,...).

- Suivant vos compétences : automates dont les transitions des cellules dépendent de l'état de leurs voisins et d'une couche cartographique sous-jacente (feux de forêts, tsunamis,...), choix de la modélisation, modélisation 3D, travail sur les optimisations des algorithmes de calcul,...

En fonction du choix de votre thème, vous penserez à citer vos sources : articles ou pages web qui vous ont permis de programmer votre automate.

Exigences techniques

Exigences concernant la manipulation des données

Les cellules de l'automate doivent être gérées via des structures stockées dans un tableau à une dimension (données) et à travers un tableau de caractères à deux dimensions (affichage). Typiquement une cellule de l'automate aura les champs suivants :

CELLULE

```
typedef struct cel {
char type[CMAX]; // type de la cellule
char etat[CMAX]; // état de la cellule par rapport au type
int tps;         // temps courant de la simulation
int posH;        // position horizontale
int posV:        // position verticale
...              // champs spécifiques au thème
} T_CEL;
```

A chaque t donné on mettra à jour le tableau puis l'affichage. Mode opératoire pour une transition de t à t+1 :

1) recopie du tableau de initial de cellules, ainsi pour la transition de t à t+1 il y a deux tableaux de cellules TABCELL(t) et TABCELL(t+1).
2) pour chaque cellule de TABCELL(t+1) calculer la transition à partir de l'état des cellules à TABCELL(t).
3) recopie de TABCELL(t+1) dans TABCELL(t).
4) affichage (si demandé) de tout ou partie de la carte.

Nota : l'utilisation d'un tableau à une seule dimension permet de ne traiter que les cellules vivantes. Il est plus efficace lorsque le rapport entre les cellules vivantes et l'ensemble de la grille est petit. Toutefois il faut prendre en compte les opérations de tassement du tableau: suppression des cellules devenues vides, la création des cellules,...

Exigences concernant l'affichage

L'affichage se fait en mode console (caractère) de gauche à droite et de haut en bas. La taille de la grille calculée est H x V : H largeur horizontale, V longueur verticale. Typiquement un tableau d'affichage sera déclaré sous la forme :

AFFICHAGE

```
char MAP[H][V];   //H largeur réelle, V hauteur réelle
```

Fonctionnalité recadrage : taille de la grille affichage en mode recadrage
La taille de la grille de l'automate peut-être plus grande que la taille d'affichage de la console. On doit pouvoir ne faire afficher qu'une partie de la grille.
On peut demander de n'afficher que h x v cases de la grille ($0< h < H$, $0 < v < V$) à partir de x0 et y0. Contraintes : $x1 = x0 + h \leq H$, $y1 = y0 + v \leq V$.
Nota : pour l'affichage vous pouvez utiliser une bibliothèque graphique telle qu'OpenGL, à la condition expresse que vous ayez respecté les spécifications impératives du projet

Exigences concernant le calcul des états
Fonctionnalité période de la simulation : on doit pouvoir demander d'afficher la grille entre t1 et t2 dans les conditions suivantes : $0 \leq t1 < t2$.

Exigences concernant le chargement et la sauvegarde d'un fichier
Pour cette partie, vous avez le choix d'utiliser les fichiers binaires ou les fichiers textes.

1. Le programme doit permettre de charger une configuration particulière de l'automate cellulaire dans le tableau des structures.
2. Le programme doit permettre de sauvegarder la configuration courante de l'automate cellulaire.

Remarques :
On ne demande pas de sauvegarder le tableau d'affichage qui peut être reconstruit à tout moment à l'aide du seul tableau de structures.
Les fichiers binaires sont plus simples à mettre en oeuvre que les fichiers textes.
Si vous faites une sauvegarde au format texte (resp. binaire), vous devez avoir un chargement au format texte (resp. binaire).

La fonction main() doit contenir un tableau de structures T_CELL et un tableau de caractères MAP pour l'affichage.

Exigences fonctionnelles
Le menu du programme doit permettre *a minima* les options suivantes :

(1) Quitter le programme
(2) Sauvegarder l'automate dans un fichier
(3) Charger l'automate depuis un fichier
(4) Créer et/ ou modifier les cellules de l'automate
(5) Paramétrer le recadrage de l'affichage (x0,h, y0, v)
(6) Calculer l'automate jusqu'à un temps (t) donné sans affichage
(7) Afficher l'automate au temps (t) courant
(8) Calculer et afficher entre t0 et t1 ($0 < t0 < t1$)

Suivant le thème choisi et les fonctionnalités spécifiques développées vous rajouterez des options de menu.

Déclaration des variables
Il ne doit y avoir aucune variable globale.

Mise en œuvre sur les tableaux de structures
On utilisera de préférence des tableaux à une dimension pour la gestion des structures.

Tous les tableaux à une dimension doivent être passés par paramètre de la manière suivante :
On suppose T déclaré de la manière suivante : *T_CELL T[TMAX];*
void FonctionUtilisantTableau (T_CELL T[ ], int Dim) lorsque l'on ne modifie pas le nombre d'éléments du tableau *T* qui est de *TMAX* et qui contient *Dim* cases utilisées.

*void FonctionModifiantTableau (T_CELL T[ ], int *Dim, int TMAX)* lorsqu'on modifie le nombre d'éléments du tableau. *TMAX* est la taille du tableau, *Dim* est le nombre d'éléments en cours avant l'appel de la fonction et qui peut être modifié à l'exécution de la fonction.

Exemple d'automate cellulaire

Automate de Conway
Propagation de feux de forêts
Propagation de virus dans une population

Bon travail !!