TP4 : Automates à états finis

Séances 6 et 7 de TP de C – Evaluation du binôme en séance 7

1. Objectifs

L'objectif de cette séance est d'appliquer, en langage C, les notions d'algorithmiques sur les automates à états finis vus en cours d'algorithmique.

Les notions de langages C qui vous seront utiles sont les tableaux à 2 dimensions et l'allocation dynamique en mémoire. Puis les fonctions avec passages de paramètres par adresse

Dans 1°temps, vous reprendrez les éléments concernant l'exemple sur les nombres avec partie décimale.

Dans 2°temps, vous reprendrez les éléments concernant le calcul de ce nombre s'il s'avère être un nombre décimal.

Vous devrez mettre en place des tests unitaires qui permettront de vérifier le bon comportement de vos fonctions. L'utilisation de la fonction « scanf » est encore et toujours interdite, même durant la phase de développement et de mise au point.

Vous devrez donc concevoir vos tests unitaires en même temps que vos fonctions.

2. Les fonctionnalités attendues

1° partie

Tout cela a déjà été codé en TD d'algorithmique, cela devrait aller très vite.

- Allouer dynamiquement les matrices de transitions
- Faire les fonctions de vérifications d'étiquettes
- Faire la fonction de correspondance entre l'étiquette et la colonne de transition
- Faire la fonction qui vérifie, caractère par caractère, la véracité de l'expression passée en paramètre

Pour les tests unitaires, proposez un jeu d'essais pertinents

 20 minutes pour les fonctions + 30 minutes pour les tests unitaires

2° partie

Ces fonctions reprennent des fonctions de la 1° partie en les dupliquant pour le calcul de l'expression

- Allouer dynamiquement la matrice des actions
- Ecrire la fonction qui fait note le signe et calcule les parties entières et décimales
- Ecrire à partir de la fonction qui examine la véracité de l'expression, une nouvelle fonction qui effectue en plus le calcul

Reprendre le précédent jeu d'essais et les tests unitaires

 20 minutes pour les fonctions + 30 minutes pour les tests unitaires.

3° partie

Définir un nouveau graphe de l'automate pour des expressions décimales avec un exposant de la forme :

$10.2^{E}-5$, $-3.002^{E}12$, $45^{E}-3$

Il n'y a d'espace en la partie décimale (ou entière) et l'exposant
Coder les fonctions suivantes

- Allouer une nouvelle matrice de transition pour le graphe correspondant.
- Allouer une nouvelle matrice d'actions pour le calcul de l'expression.
- Créer une nouvelle fonction pour le traitement des actions
- Ecrire une nouvelle fonction qui examine la véracité de l'expression et qui effectue le calcul
- Comment faire évoluer la dernière fonction de la 2° partie afin de prendre en compte des fonctions différentes qui traitent le calcul

Pour les tests unitaires, proposez un jeu d'essais pertinents

 45 minutes pour le graphe et la matrice de transitions, 45 minutes pour l'évolution du code et 30 minutes pour les tests unitaires.

3. Astuces

- Démarrez un nouveau projet – Ne repartez pas d'un ancien TP – Réfléchissez à l'organisation de vos fichiers sources
- Aidez-vous de vos TD d'algorithmique
- La fonction qui examine la véracité de l'expression doit être indépendante de la matrice de transition (et d'action). Ces matrices seront passées sous forme de paramètre.
- Une façon de réduire le nombre de paramètres pour une fonction (et surtout des pointeurs) est d'utiliser une structure.
- En langage C, il existe aussi des pointeurs sur fonctions (comme avec qsort au TP3)
- Réutilisez ce que vous avez fait aux TPs précédents pour les tests unitaires
- Concevez vos tests unitaires avant de coder vos fonctions correspondantes

4. Evaluations

- Maîtrise de Netbeans (Création d'un projet, ajout de fichiers sources c,h, compilation, debugger)
- Conventions de codage (commentaires, indentation, nom des fichiers, des fonctions, des variables, ...)
- Compilation séparée .h/.c, séparation du code en fonctions
- Tableaux, boucles
- Structures
- Allocation dynamique de mémoire
- Pointeurs, passage de paramètres par valeur/par adresse
- Respect des contraintes de l'énoncé
- Tests unitaires & jeux d'essais