Projets possibles pour Lex et Yacc

Date ultime de remise du projet: vendredi 8 juin 2008.

Envoyez moi une archive tar de votre projet à dmichel@u-bourgogne.fr. Mettez votre nom en commentaire dans vos sources. Joignez le makefile, quelques fichiers d'exemples, et les quelques explications nécessaires dans un fichier LISEZ-MOI (n'abusez pas des erreurs d'orthographe: elles m'exaspèrent).

Une grosse journée de travail suffit, mais n'attendez pas le dernier moment pour commencer. N'hésitez pas à me poser des questions lorsque je vous verrai en tp.

Vous pouvez travailler en binôme, ou seul. A chaque projet, ajoutez une petite touche personnelle.

N'utilisez pas de lettres accentuées dans vos sources, vos fichiers de données, votre langage!

Choisissez un des projets suivants. Eventuellement, proposez moi votre propre sujet, mais demandez mon accord.

1. Modifiez le "grapheur"[1], pour dessiner des régions définies par des inéquations, ainsi que des combinaisons booléennes d'inéquations.

2. Modifiez le "grapheur" pour dessiner des régions définies par des inéquations. Chaque région a une couleur associée, et une profondeur associée (un entier). Vous afficherez les régions par profondeur décroissante.

3. Modifiez le "grapheur" pour prendre en compte les fonctions cos et sin. Pour calculer le cosinus ou le sinus d'un intervalle, il vous faudra décomposer l'intervalle en parties monotones. Penser à afficher les courbes $y - \cos x = 0$, et $y - \sin x = 0$ pour contrôler. Enfin, ajoutez des variables pour définir des constantes comme dans "c=2.5 ;", et pour factoriser comme dans: "soit t=(x+y)(x-y) dans (t*x-1)*(t*y+1)". Choisissez librement votre syntaxe; elle doit cependant être cohérente, donc n'hésitez pas à modifier la syntaxe du grapheur.

3. Modifiez le "grapheur", en ajoutant une variable z. Vous créerez des animations en faisant varier la valeur de z, et en visualisant les points x, y, z tels que $f(x, y, z) = 0$. Prévoir des boucles pour faire varier z : par exemple avec la syntaxe "for z = 0 to 10 step 0.01 do". Ce qui est visualisé est donc une coupe d'une surface par un plan.

4. Modifiez la calculatrice, pour écrire un petit interprète. La syntaxe de votre langage est libre. Seul le type entier est demandé. Votre langage doit permettre de définir la fonction factorielle et la fonction fibonacci avec

[1] http://ufrsciencestech.u-bourgogne.fr/master1/mi1-tc5/SOURCES/GRAPHER_INTERVAL_3/

la récursion naïve. Pour gérer l'évaluation, vous pouvez utiliser une pile de couples : (nom de variable, valeur entière). La valeur d'une variable est la valeur entière stockée la plus haut dans la pile. Pour évaluer un appel de fonction, il faut évaluer (par un appel récursif à votre fonction d'évaluation) les arguments de la fonction, puis empiler les couples pour les paramètres de la fonction et leurs valeurs. Variante : vous pouvez gérer une pile de valeurs par nom de variable.

5. Décrivez des scènes 2D ou 3D, dans un format libre, et visualisez les en openGl. Le langage de description n'a pas à fournir les variables, les boucles, ni les fonctions. Vous pouvez récupérer et modifier le fichier joint fig.cpp.

7. Ré-écrivez le mini interprète forth sur ce site (miniforth.ml). Il est actuellement écrit en ocaml (moins de 100 lignes). Le programme ocaml n'effectue pas d'analyse syntaxique, ni lexicale. Vous devez l'effectuer. Une syntaxe possible est: "fib: dup 2 le if dup -1 add fib swap -1 add fib add else fi ." Ceci décrit la fonction fibonacci. Les entiers négatifs seront reconnus par lex (ou flex). Lisez un tutoriel sur Forth ou sur Joy pour plus de détail. Seul le type entier est demandé, et la possibilité de définir des fonctions récursivement, comme factorielle et Fibonacci. Ce language est sans variable.