Compiler Design Lab - CS-307

MATHLIB
LANGUAGE FOR MATHEMATICS

Neeraj Rajpurohit - 201751027

Akshita Agarwalla - 201751066

Divy Pandya - 201751013

Introduction

MathLib programming language that performs complex mathematical operations by using basic mathematical symbols. The target is to make it more readable for mathematicians and help them code in the same manner as they solve questions on paper. There are two types of variables in the grammar, named Vector and Polynomials. Declaration of polynomials begin with '@'.

Syntax

- Identifier initialization: <var_name> = <value>;
    - Example: a = 2;
    - Identifiers can be of one english letter. (A-Z, a-z)
- Operations:
    - Addition: <exp> + <exp>
    - Subtraction: <exp> - <exp>
    - Multiplication: <exp> * <exp>
    - Division: <exp> / <exp>
    - Exponent: <exp> ^ <exp>
    - Factorial: <exp> !
    - Power of e: exponent (<exp>)
    - Logarithm: logarithm (<exp>)

- Array:
    - Declaration: <array_name> = { <array elements separated by ‘,’ } ;
    - Accessing an element: <array_name>< <index_value> > ;
    - Appending Values: <array_name> . append(<new_values separated by ‘,’>);
    - Removing last element: <array_name> . pop ;
    - Example:
        - a = { 1, 2, 3, 4 };
        - a<0>; // gives the first element of array i.e. 1
        - a.append(5, 6); // changes a to {1,2,3,4,5,6}
        - a.pop; // changes a to {1,2,3,4,5};
- Polynomial Functions:
    - Declaration: <func_name> = @ (<func coefficients separated by ‘,’) ;
    - Evaluating f(x): evalFun(<func_name> <val_x>);
    - Differentiating a function: d = diff(f);
    - Evaluating differentiation of f(x) given x: evalDiff(<func_name> <val_x>);
    - Finding root: root <func_name>;
    - Example:
        - f = @ (2, 3, 4); // Gives a polynomial of the form $4x^2 + 3x + 2$
        - evalFun(f 5); // returns 117
        - evalDiff(f 3); //returns 27
- Other commands:
    - show<identifier>; // prints the value of the identifier
    - typeof<identifier>; // tells whether the identifier is a vector or a polynomial
    - over; // to exit from idle
    - Example:
        - a = @(2,3,4);
          typeof a;
          => Polynomial
        - b = {4,2,3};
          typeof b;
          => Vector

How to Run

In console type the following commands:

1. yacc -d mathLib.y
2. lex mathLib.l
3. gcc lex.yy.c y.tab.c -o mathLib
4. ./mathLib < input.txt // this to run the input file
5. ./mathLib // this to access the idle

Grammar

statement -> assignment ;

| over ;

| evalExp expression ;

| variable . operation ;

| show variable ;

| typeof variable ;

| statement assignment ;

| statement evalExp expression ;

| statement variable . operation ;

| statement show variable ;

| statement typeof variable ;

| statement over ;

assignment -> variable = DS

| variable = polynomial

| variable = differentiation

polynomial -> @ (arrayelements)

differentiation -> differentiate (variable)

DS -> { arrayelements }
| expression

operation -> append (arrayelements)
| pop

arrayelements -> arrayelements , expression
| expression

expression -> E

E -> E + E
| E - E
| E / E
| E * E
| E ^ E
| (E)
| E !
| exponent (E)
| logarithm (E)
| - E
| element

| evaluateFunction (variable E)

| root variable

| evaluateDifferentiation (variable E)

element -> digits

| variable

| variable < digits >