Cindy Marie Méndez Avilés

COMP4036-070: Programming Languages
Prof. Alcibiades Bustillo Zárate
April 26, 2021

Proposal: Nero Programming Language

Introduction

The Nero progrogramming language will be a simple, yet typed programming language aimed mostly for young users to expose them to what programming is. A lexer and a parser will be created using Flex and Yacc in C, in order to develop the wanted programming language. The language will be simple, fun, and useful for many science classes that every kid has to go through. The program will have a gamma of functions that will help with the day to day school activities of today's future and they will be able to do all kinds of mathematical equations inside of it.

Motivations & Reasons

There are many young students from seventh through twelfth grade that love using computers, and are interested in learning more about them.The constant use of technology is something that is growing every day more and more. For this reason, the main goal of the Nero programming language is on those students who like the idea of programming and would like to learn the basics of programming. In many schools, there is no presence at all of any programming. With Nero, the plans are to design a simple, useful programming language that will help kids through their science assignments and at the same time expose them to the new world of programming languages. The more programmers we have, the better for our future.

Features

- One of the most important features of our programming language is that users will have the ability to enter the help command. Once given, the program will list the list of features available in our programming language, including all the functions that the users can use.

- The programming language will allow users to use either int or float variables. However, for the sake of simplicity and better understanding, the int variables will be converted to float variables.

- Users will have the ability to print a string, term or an expression with the print command.

- The programming language will store users defined variables, their type and their value. Users will be able to assign and re-assign values and use those variables in any kind of mathematical expressions. For example:

```
int x = 6;
print x;
// This would print 6
x = x + 4;
print x;
// This would print 10;
```

- We will allow the following mathematical expressions:
  - Addition
    - int x = 3 + 6 + 5;
  - Subtraction
    - float y = 4 - 5 - 3;
  - Multiplication
    - float z = 4 * 3 * 3;
  - Division
    - float a = 4.2 / 3.5;
  - Power
    - float b = 4^3;
  - Square Roots
    - float c = square_root(4);
  - Sine
    - float d = sin(90);
  - Cosine
    - float e = cos(0);
  - Tangent
    - float f = tan(1);
  - Mathematical expressions in parenthesis
    - float g = (3 + 5) * (4 + 5.2);
  - Elaborate mathematical expressions
    - float h = 3^2 + 5 + (3/2.5) - 4;

- Now for the main feature of our programming language, will have several functions that the users will be able to use for their daily science activities. The functions are the following:
  - Calculate the area of a triangle. Given a base and a height, the function will return the area of the wanted triangle.
    - float **calc_tri_area**(float base, float height);
  - Calculate the area of a rectangle. Given the length and width of a rectangle, the program will calculate the area.
    - float **calc_rect_area**(float length, float width);

    - Calculate the area of a circle. Given the radius of the circle, the program will calculate the area of a circle.
        - float **calc_circ_area**(float radius);
    - Calculate the volume of a rectangle. Given the length, width and height of the rectangle, the program will calculate the volume.
        - float **calc_vol_rect**(float length, float width, float height);
    - Calculate the volume of a sphere. Given the radius of a sphere, the program will calculate the volume.
        - float **calc_vol_sphere**(float radius);
    - Calculate the volume of a cylinder. Given the radius and height of a cylinder, the program will calculate the volume.
        - float **calc_vol_cyl**(float radius, float height);
    - Calculate the roots of a quadratic equation. Given the equation: ax^2 + bx + c = 0, and using those constants, a, b and c, the program will print the real roots, if any and it will return the amount of roots the quadratic equation has.
        - float **find_roots**(float a, float b, float c);
    - Calculate the circumference of a circle. Given the radius, the PL will calculate the circumference of a circle.
        - float **calc_circ_circum**(float radius);
    - Calculate the force given mass and acceleration.
        - float **calc_force**(float mass, float acceleration);
    - Calculate the voltage given a current and a resistance.
        - float **calc_voltage**(float current, float resistance);

- There will be a command to exit the program called exit.

Programming Language Architecture

Nero's Lexer:

```
%{
#include "y.tab.h"
#include <string.h>
#include <stdlib.h>
void yyerror (char *s);
int yylex();
%}
%%
"print"                                 {return PRINT_TOKEN;}
"exit"                                  {return EXIT_TOKEN;}
"calc_tri_area"                         {return T_AREA_TOKEN;}
"calc_rect_area"                        {return R_AREA_TOKEN;}
"calc_circ_area"                        {return C_AREA_TOKEN;}
"calc_circ_circum"                      {return C_CIRC_TOKEN;}
"calc_vol_rect"                         {return R_VOL_TOKEN;}
"calc_vol_sphre"                        {return S_VOL_TOKEN;}
"calc_vol_cyl"                          {return C_VOL_TOKEN;}
"square_root"                           {return S_ROOT_TOKEN;}
"sin"                                   {return SIN_TOKEN;}
"cos"                                   {return COS_TOKEN;}
"tan"                                   {return TAN_TOKEN;}
"find_roots"                            {return ROOTS_TOKEN;}
"calc_force"                            {return FORCE_TOKEN;}
"calc_voltage"                          {return VOLTAGE_TOKEN;}
int|float                               {yylval.myType = strdup(yytext); return DATA_TYPE_TOKEN;}
[a-zA-Z]*                               {yylval.myString = strdup(yytext); return VAR_TOKEN;}
[0-9]+                                  {yylval.num = atoi(strdup(yytext)); return INT_TOKEN;}
[0-9]+[.][0-9]+                         {yylval.fl = atof(strdup(yytext)); return FLOAT_TOKEN;}
\"([^\\\"]|\\.)*\"                      {yylval.stringText = strdup(yytext); return STRING_TOKEN;}
[ \t\n]                                 ;
[-+*/=;(),^]                            {return yytext[0];}
.                                       {yyerror ("unexpected character");}

%%
int yywrap (void)
{
    return 1;
}
```

Nero's Parser:

```
//GRAMMAR//
/* descriptions of expected inputs                          corresponding actions (in C) */

STATEMENT       : ASSIGNMENT ';'                            {;}
                | EXIT_TOKEN ';'                            {exit(0);}
                | PRINT_TOKEN EXPR ';'                      {printf("%.2f\n", $2);}
                | PRINT_TOKEN STRING_TOKEN ';'              {printf("%s\n", $2);}
                | EXPR ';'                                  {printf("%.2f\n", $1);}
                                                            //Con esto el programa es mas simple y puedo escribir sin(5)
                                                            //en el cmd y obtener la respuesta sin tener que escribir print
                | STATEMENT ASSIGNMENT ';'                  {;}
                | STATEMENT EXIT_TOKEN ';'                  {exit(0);}
                | STATEMENT PRINT_TOKEN EXPR ';'            {printf("%.2f\n", $3);}
                | STATEMENT PRINT_TOKEN STRING_TOKEN ';'    {printf("%s\n", $3);}
                | STATEMENT EXPR ';'                        {printf("%.2f\n", $2);}
                ;

ASSIGNMENT : DATA_TYPE_TOKEN VAR_TOKEN '=' EXPR         {
                                                            int varIndex = getVarIndex($2);
                                                            if(varIndex == -1)
                                                            {
                                                                //Uso la funcion strcpy para copiar el
                                                                //valor de VAR_TOKEN en el arreglo varNames.
                                                                strcpy(varNames[currentVarCounter], $2);
                                                                //Copio el valor de la expresion en el arreglo varValues
                                                                varValues[currentVarCounter] = $4;
                                                                //Uso la funcion strcpy para copiar el
                                                                //valor de DATA_TYPE en el arreglo varTypes
                                                                strcpy(varTypes[currentVarCounter], $1);
                                                                //Le anado 1 al var counter
                                                                currentVarCounter++;
                                                                //Ex. DATA_TYPE_TOKEN = int, VAR_TOKEN = x, EXPR = 5
                                                            }
                                                            else
                                                            {
                                                                //Si la variable que el usuario entro ya estaba
                                                                //definida me da error
                                                                yyerror("Var already exists!");
                                                                exit(0);
                                                            }
                                                        }
```

```
115             | VAR_TOKEN '=' EXPR                    {
116                                                       //Busca el indice en donde la variable esta almacenada
117                                                       int varIndex = getVarIndex($1);
118                                                       //Si no lo encuentra me da un error
119                                                       if(varIndex == -1)
120                                                       {
121                                                           yyerror("Var hasn't been initialized!");
122                                                           exit(0);
123                                                       }
124                                                       //Si lo encuentra entonces el valor de la variable se reasigna
125                                                       else
126                                                       {
127                                                           varValues[varIndex] = $3;
128                                                       }
129                                                   }
130             ;
131   EXPR      : TERM                                  {$$ = $1;}
132             | EXPR '+' EXPR                         {$$ = $1 + $3;}
133             | EXPR '-' EXPR                         {$$ = $1 - $3;}
134             | EXPR '*' EXPR                         {$$ = $1 * $3;}
135             | EXPR '/' EXPR                         {$$ = $1 / $3;}
136             | EXPR '^' EXPR                         {$$ = pow($1, $3);}
137             | '(' EXPR ')'                          {$$ = $2;}
138             | T_AREA_TOKEN '(' EXPR ',' EXPR ')'    {$$ = ($3 * $5) / 2.0;}
139             | R_AREA_TOKEN '(' EXPR ',' EXPR ')'    {$$ = $3 * $5;}
140             | C_AREA_TOKEN '(' EXPR ')'             {$$ = pi * pow($3, 2.0);}
141             | C_CIRC_TOKEN '(' EXPR ')'             {$$ = 2.0 * pi * $3;}
142             | S_ROOT_TOKEN '(' EXPR ')'             {$$ = sqrt($3);}
143             | R_VOL_TOKEN '(' EXPR ',' EXPR ',' EXPR ')'  {$$ = $3 * $5 * $7;}
144             | S_VOL_TOKEN '(' EXPR ')'              {$$ = (4.0/3.0) * pi * pow($3, 3.0);}
145             | C_VOL_TOKEN '(' EXPR ',' EXPR ')'     {$$ = pi * pow($3, 2.0) * $5;}
146             | SIN_TOKEN '(' EXPR ')'                {$$ = sin($3);}
147             | COS_TOKEN '(' EXPR ')'                {$$ = cos($3);}
148             | TAN_TOKEN '(' EXPR ')'                {$$ = tan($3);}
149             | ROOTS_TOKEN '(' EXPR ',' EXPR ',' EXPR ')'  {$$ = findRoots($3, $5, $7);}
150             | FORCE_TOKEN '(' EXPR ',' EXPR ')'     {$$ = $3 * $5;}
151             | VOLTAGE_TOKEN '(' EXPR ',' EXPR ')'   {$$ = $3 * $5;}
```

```
155   TERM      : INT_TOKEN                 {$$ = (float) $1;}
156             | VAR_TOKEN                 {
157                                             //Busca el indice de la variable y si existe me da
158                                             //el valor de la variable en la expresion
159                                             //Si la variable no existe me da error
160                                             int varIndex = getVarIndex($1);
161                                             if(varIndex == -1)
162                                             {
163                                                 yyerror("Var doesn't exist!");
164                                                 exit(0);
165                                             }
166                                             else
167                                                 $$ = varValues[varIndex];
168                                         }
169             | FLOAT_TOKEN               {$$ = $1;}
170             ;
```

Implementation Requirements and Tools

- Lex/Flex - as a scanner generator in C
- YACC - as a parser generator in C

Example of a Nero Program

```
print "Program started";
print 5;
float x = 6;
print x;
print "Value of sin(1):";
sin(1);
print "Value of cos(2):";
cos(2);
print "Finding roots of x^2 + 9x + 20";
float a = 1;
float b = 9;
float c = 20;
find_roots(a, b, c);
print "Complex Math Equation:";
square_root((3*5 + 10/2)/4);
//2.24
print "Program completed";
```

Github Link

https://github.com/Cindy-Mendez/Nero

End-Goal

The main goal of our programming language is to ensure that kids around the world get introduced to programming and see how powerful it can be. Making the programming language simple enough that they feel encouraged to try it out, and do wonderful things with Nero and eventually other more powerful programming languages. We want this world to have more computer scientists/engineers.