

Cindy Marie Méndez Avilés

COMP4036-070: Programming Languages

Prof. Alcibiades Bustillo Zárate

May 14, 2021

Final Report: Nero Programming Language

Introduction

The Nero programming language will be a simple, yet powerful 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 for 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.

Language Tutorial

<https://drive.google.com/file/d/1Ex21cNjm64bEerPzCdzMAb3JrR7IjLrr/view?usp=sharing>

Language Reference Manual

- 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. This works as a reference manual as well. The info displayed will be similar to what we depict in this report.

- 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. For example:

```
print "hello world";  
//This would print: hello world  
print 3 + 5;  
//This would print 8  
print 2;  
//This would print 2
```

- The programming language will store user 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  
float y = x + 5.5;  
//This would print 15.5
```

- We will allow the following mathematical expressions:
 - Addition
 - `int x = 3 + 6 + 5`
x would be stored in memory with a value of 14
 - Subtraction
 - `float y = 4 - 5 - 3;`
y would be stored in memory with a value of -4
 - Multiplication
 - `float z = 4 * 3 * 3;`
z would be stored in memory with a value of 36
 - Division
 - `float a = 4.2 / 3.5;`
a would be stored in memory with a value of 1.2
 - Power
 - `float b = 4^3;`
b would be stored in memory with a value of 64

- Square Roots
 - `float c = square_root(4);`
c would be stored in memory with a value of 2
 - Sine
 - `float d = sin(90);`
d would be stored in memory with a value of 0.89
 - Cosine
 - `float e = cos(0);`
e would be stored in memory with a value of 1
 - Tangent
 - `float f = tan(1);`
f would be stored in memory with a value of 1.56
 - Mathematical expressions in parenthesis
 - `float g = (3 + 5) * (4 + 5.2);`
g would be stored in memory with a value of 73.60
 - Elaborate mathematical expressions
 - `float h = 3^2 + 5 + (3/2.5) - 4;`
h would be stored in memory with a value of 11.2
- 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.

Language Development:

Programming Language Architecture

Nero's Lexer:

```
1  %{
2  #include "y.tab.h"
3  #include <string.h>
4  #include <stdlib.h>
5  void yyerror (char *s);
6  int yylex();
7  %}
8  %%
9  "print"                {return PRINT_TOKEN;}
10 "exit"                 {return EXIT_TOKEN;}
11 "calc_tri_area"        {return T_AREA_TOKEN;}
12 "calc_rect_area"       {return R_AREA_TOKEN;}
13 "calc_circ_area"       {return C_AREA_TOKEN;}
14 "calc_circ_circum"     {return C_CIRC_TOKEN;}
15 "calc_vol_rect"        {return R_VOL_TOKEN;}
16 "calc_vol_sphre"       {return S_VOL_TOKEN;}
17 "calc_vol_cyl"         {return C_VOL_TOKEN;}
18 "square_root"          {return S_ROOT_TOKEN;}
19 "sin"                  {return SIN_TOKEN;}
20 "cos"                  {return COS_TOKEN;}
21 "tan"                  {return TAN_TOKEN;}
22 "find_roots"           {return ROOTS_TOKEN;}
23 "calc_force"           {return FORCE_TOKEN;}
24 "calc_voltage"         {return VOLTAGE_TOKEN;}
25 int|float              {yylval.myType = strdup(yytext); return DATA_TYPE_TOKEN;}
26 [a-zA-Z]*             {yylval.myString = strdup(yytext); return VAR_TOKEN;}
27 [0-9]+                 {yylval.num = atoi(strdup(yytext)); return INT_TOKEN;}
28 [0-9]+[.][0-9]+       {yylval.fl = atof(strdup(yytext)); return FLOAT_TOKEN;}
29 \"([^\"]|\\\"|\\\\\\\\)*\" {yylval.stringText = strdup(yytext); return STRING_TOKEN;}
30 [ \\t\\n]              ;
31 [-+*/=;(),^]          {return yytext[0];}
32 .                     {yyerror ("unexpected character");}
33
34 %%
35 int yywrap (void)
36 {
37     return 1;
38 }
```

Nero's Parser:

```

73 //GRAMMAR//
74 /* descriptions of expected inputs                                corresponding actions (in C) */
75
76 STATEMENT : ASSIGNMENT ';' ; {}
77           | EXIT_TOKEN ';' ; {exit(0);}
78           | PRINT_TOKEN EXPR ';' ; {printf("%.2f\n", $2);}
79           | PRINT_TOKEN STRING_TOKEN ';' ; {printf("%s\n", $2);}
80           | EXPR ';' ; {printf("%.2f\n", $1);}
81           //Con esto el programa es mas simple y puedo escribir sin(5)
82           //en el cmd y obtener la respuesta sin tener que escribir print
83           | STATEMENT ASSIGNMENT ';' ; {}
84           | STATEMENT EXIT_TOKEN ';' ; {exit(0);}
85           | STATEMENT PRINT_TOKEN EXPR ';' ; {printf("%.2f\n", $3);}
86           | STATEMENT PRINT_TOKEN STRING_TOKEN ';' ; {printf("%s\n", $3);}
87           | STATEMENT EXPR ';' ; {printf("%.2f\n", $2);}
88           ;
89
90 ASSIGNMENT : DATA_TYPE_TOKEN VAR_TOKEN '=' EXPR
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113

```

```

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; }

```

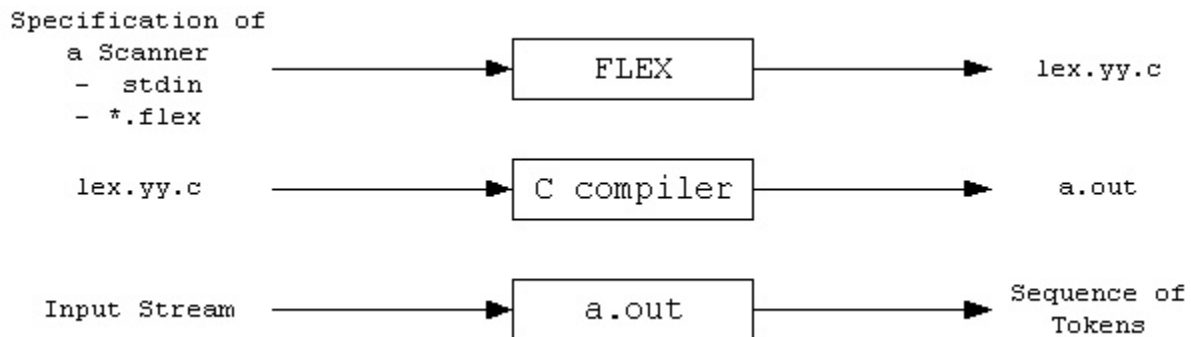
```

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 | ;

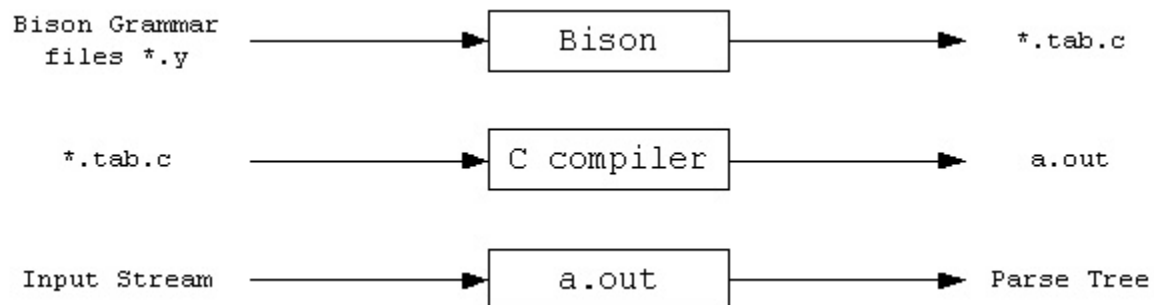
```

Interfaces between modules

Flex - (Fast LEXical analyzer generator) is a tool for generating scanners. Instead of writing a scanner from scratch, you write a specification of patterns using regular expressions, and Flex will construct a scanner.



Now with our sequence of tokens, we use **Yacc/Bison** to find the hierarchical structure of the program using context free grammar, and at the same time translate the code into executable c code.



Both figures were taken from: <http://alumni.cs.ucr.edu>

To run our programming language Nero, we have to do the following:

- Write a lexical analyzer to process input and pass tokens to the parser (nero.l)
- Write the grammar specification for bison (nero.y), including grammar rules.
- Run Bison on the grammar to produce the parser. (Makefile or using the app Flex Windows (Yacc and Lex)
- Compile the code output by Bison, as well as any other source files.
- Link the object files to produce the finished product (an executable of our programming language Nero.exe).

The **makefile** would look like:

```
all: nero
```

```
nero.tab.c nero.tab.h:  nero.y  
    bison -t -v -d nero.y
```

```
lex.yy.c: nero.l nero.tab.h  
    flex nero.l
```

```
nero: lex.yy.c nero.tab.c nero.tab.h  
    gcc -o nero nero.tab.c lex.yy.c
```

```
clean:  
    rm nero nero.tab.c lex.yy.c nero.tab.h nero.output
```

As an alternative, we can use the app Flex Windows (Yacc and Lex) to compile the programming language. Once the executable has been created, then we can just call it through the command prompt: nero and start typing code, or using the nero < <filename> call in Windows or ./ciel <filename> in Linux environments to run a piece of code in a text file.

Software Development Environment

To develop this programming language we used Visual Studio Code for all of our development needs and Flex Windows (Lex and Yacc) to compile the programming language and convert it to an executable.

Testing Methodology

For this project we mostly focused on unit testing, making sure every little thing that we added we tested it out and didn't break the program, we had a few test programs that we ran every time a change was made and we ran those programs to see if we got the expected results. A little bit of integrating testing was also done when we integrated the Lexer and the Parser.

Example of a Nero program that we used for testing:

```
print "Program started";
print "Statement: print 5; - Expecting a value of 5";
print 5;
print "Statement: float x = 6; - No value expected";
float x = 6;
print "Statement: print x; - Expecting a value of 6";
print x;
print "Statement: print sin(90); - Expecting a value of .89";
print sin(90);
print "Statement: float h = 3^2 + 5 + (3/2.5) - 4; - No value expected";
float h = 3^2 + 5 + (3/2.5) - 4;
print "Statement: print h + x; - Expecting a value of 17.2";
print h + x;
print "Finding roots of x^2 + 9x + 20";
print "Statement: float a = 1; - No value expected";
float a = 1;
print "Statement: float b = 9; - No value expected";
float b = 9;
print "Statement: float c = 20; - No value expected";
float c = 20;
print "Statement: find_roots(a, b, c); - Expected roots are: -4, -5";
find_roots(a, b, c);
print "Statement: square_root((3*5 + 10/2)/4); - Expected Value 2.24:";
square_root((3*5 + 10/2)/4);
print "Program completed";
exit;
```

Github Link

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

Conclusion

Overall we feel extremely successful with this project, as completely new technologies were learned to be able to accomplish everything that we wanted to accomplish. I sincerely think that a simple programming language like this one can help a lot of early students to enjoy what programming is, and all the benefits that it can give them. See how powerful they are, and how many amazing things they can make with it. For next steps, if I had the chance, I would demo this to a group of young students, and get their input and talk about what new functions and features they would like to have in Nero.