Nero Programming Language

COMP 4036-070 - Programming Languages
Cindy M Mendez Aviles

Intro

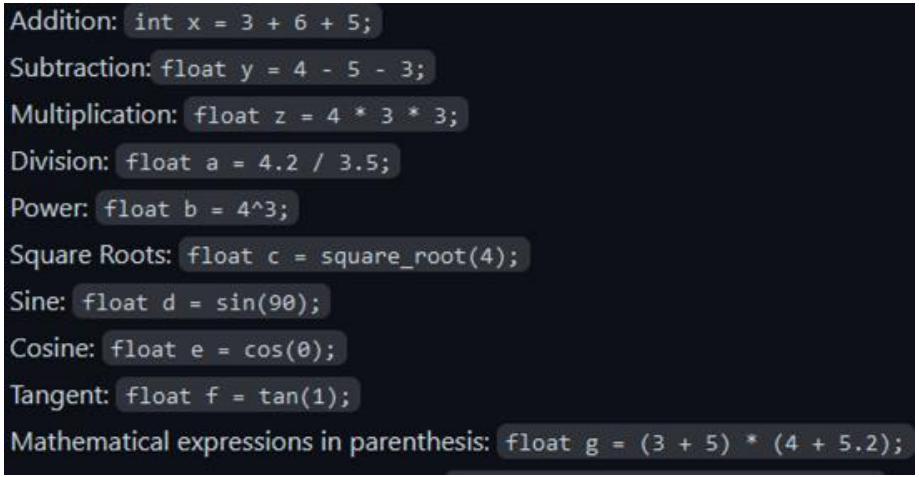
- Nero es un lenguaje de programación sencillo, pero increíblemente útil.
- Construido usando Flex y Bison en C.
- Estará enfocado en jóvenes de séptimo a duodécimo grado.
- El propósito del lenguaje de programación es que sea divertido, y útil para mucha de las clases de ciencia y matemáticas que los jóvenes hoy en día tienen que tomar.
- Tendrá muchas funciones que los usuarios podrán llamar para resolver sus asignaciones.
- A la vez también hacer y resolver expresiones matemáticas.

Motivación para crear a Nero

- Una gran cantidad de estudiantes que aman las computadoras nunca se le exponen a que es programación.
- La programación es un tema que le asusta a muchas personas, porque parece complicado.
- La tecnología aumenta día a día, y queremos que nuestra juventud sea expuesta a nuestro simple lenguaje de programación.
- Con la idea de que los ayudemos con Nero con sus tareas, y a la vez crezcamos la cantidad de programadores que tengamos aquí en PR y en el mundo
- Con Nero lo vamos introduciendo a la programación y esperamos que le guste tanto que empiecen a usar otros lenguajes de programación más poderosos.

¿Qué podemos hacer con Nero?

- Uno de los features más importantes de Nero es la función de help, para que los usuarios puedan ver todo lo que podemos hacer con Nero.
- Podemos imprimir strings, y expresiones en la pantalla.
- Asignar tipos y crear variables.
- Usar las variables en otras expresiones o funciones matemáticas.
- Realizar una gama de expresiones matemáticas.



¿Qué más podemos hacer?

- Tenemos una gran cantidad de funciones que podemos usar y la lista seguirá creciendo.

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);`

Más funciones!

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);`

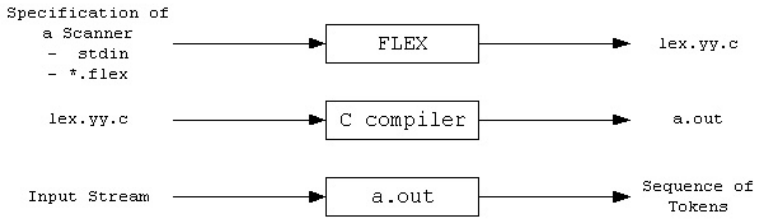
¿Porque esas funciones?

- Tratando de quedar con la temática de que nuestro programa estará enfocado para niños, las funciones son relativamente simples.
- Claro podíamos importar librerías y resolver problemas complejos, pero eso no es lo que queríamos con Nero.
- Queremos un lenguaje de programación sencillo y útil para nuestra juventud.
- En un futuro, si decidimos expandir el proyecto, buscaremos feedback de jóvenes y añadiríamos más funciones que ellos piensen que les serían útiles.

¿Cómo creamos a Nero?

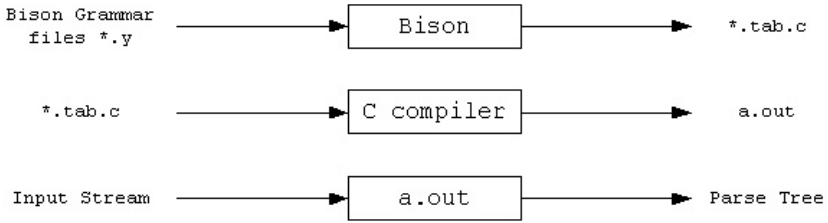
Lex / Flex

- Usamos Flex para generar nuestro scanner, y tener nuestra sequencia de tokens.
- Usamos patrones y expresiones regulares y Flex nos crea el scanner.



Yacc / Bison

- Con nuestra secuencia de tokens ya generada, usamos a Bison para encontrar nuestra estructura jerárquica usando nuestra gramática definida en el parser y con esto podemos traducir el código a un ejecutable en C.



¿Como creamos a Nero?

Ya con las tecnologías que usamos explicadas, esto son los pasos que hacemos para crear a Nero

- Creamos el lexer (nero.l) para processar inputs y pasar los tokens al parser (nero.y).
- Con nuestra gramática ya definida corremos a Bison encima de la gramática para producir el parser.
- Compilamos el output de bison y cualquier otro código.
- Y finalmente, linkeamos los files de tipo objeto para producir nuestro lenguaje de programación - Nero.

¿Cómo corremos el programa?

- También podemos usar un programa para Windows que nos crea, compila, linkea el ejecutable de Nero, llamado: Flex Windows (Flex and Yacc).
- Otra alternative, es usando un makefile ->
- Luego de tener el ejecutable creado, podemos llamar al ejecutable usando nero en el command prompt y darle instrucciones, o nero < filename para correr un programa de un archivo de texto.

```
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
```

Nero's Lexer

nero.l

Todos los archivos están

en el github!

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

```
%{
#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

nero.y

Todos los archivos están en el github!

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

```
int getVarIndex(char* aVar);
float findRoots(float a, float b, float c);
void helptext();
char * getAllButFirstAndLast(char *input);
float pi = 3.14159265359;
%}

/* Yacc/Tokens definitions */
//La union se utiliza para asignarle tipos a los tokens que se llaman con $ en la gramati
//Todas las variables que se accesan con $, necesitan un tipo, o C no va a
//saber que hacer
%union {
            int num;
            float fl;
            char* myString;
            char* myType;
            char* stringText;
        }

%start STATEMENT //donde comienza la gramatica
%token PRINT_TOKEN
%token EXIT_TOKEN
%token HELP_TOKEN
%token T_AREA_TOKEN
%token R_AREA_TOKEN
%token C_AREA_TOKEN
%token C_CIRC_TOKEN
%token S_ROOT_TOKEN
%token R_VOL_TOKEN
%token S_VOL_TOKEN
%token C_VOL_TOKEN
%token SIN_TOKEN
%token COS_TOKEN
%token TAN_TOKEN
%token ROOTS_TOKEN
%token FORCE_TOKEN
%token VOLTAGE_TOKEN
%token <stringText> STRING_TOKEN //string type
%token <num> INT_TOKEN //int type
%token <fl> FLOAT_TOKEN //float type
%token <myString> VAR_TOKEN //string type
%token <myType> DATA_TYPE_TOKEN //string type
```

Parte de la Gramatica!

nero.y

Todos los archivos están en el github!

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

```
//GRAMMAR//

STATEMENT       : ASSIGNMENT ';'
                | EXIT_TOKEN ';'
                | HELP_TOKEN ';'
                | PRINT_TOKEN EXPR ';'
                | PRINT_TOKEN STRING_TOKEN ';'
                | EXPR ';'
                | STATEMENT ASSIGNMENT ';'
                | STATEMENT EXIT_TOKEN ';'
                | STATEMENT HELP_TOKEN ';'
                | STATEMENT PRINT_TOKEN EXPR ';'
                | STATEMENT PRINT_TOKEN STRING_TOKEN ';'
                | STATEMENT EXPR ';'
                ;

ASSIGNMENT : DATA_TYPE_TOKEN VAR_TOKEN '=' EXPR
           | VAR_TOKEN '=' EXPR
           ;

EXPR    : TERM
        | EXPR '+' EXPR
        | EXPR '-' EXPR
        | EXPR '*' EXPR
        | EXPR '/' EXPR
        | EXPR '^' EXPR
        | '(' EXPR ')'
        | T_AREA_TOKEN '(' EXPR ',' EXPR ')'
        | R_AREA_TOKEN '(' EXPR ',' EXPR ')'
        | C_AREA_TOKEN '(' EXPR ')'
        | C_CIRC_TOKEN '(' EXPR ')'
        | S_ROOT_TOKEN '(' EXPR ')'
        | R_VOL_TOKEN '(' EXPR ',' EXPR ',' EXPR ')'
        | S_VOL_TOKEN '(' EXPR ')'
        | C_VOL_TOKEN '(' EXPR ',' EXPR ')'
        | SIN_TOKEN '(' EXPR ')'
        | COS_TOKEN '(' EXPR ')'
        | TAN_TOKEN '(' EXPR ')'
        | ROOTS_TOKEN '(' EXPR ',' EXPR ',' EXPR ')'
        | FORCE_TOKEN '(' EXPR ',' EXPR ')'
        | VOLTAGE_TOKEN '(' EXPR ',' EXPR ')'
```

Ejemplos de código en Nero y su output

```
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);
print "Program completed";
```

```
C:\nero>nero < nero.n
5.00
6.00
Value of sin(1):
0.84
Value of cos(2):
-0.42
Finding roots of x^2 + 9x + 20
First Root = -4.00 and Second Root = -5.00
Amount of real roots: 2.00
Complex Math Equation:
2.24
Program completed

C:\nero>
```

Conclusión

- Fue un proyecto muy divertido donde aprendí muchas tecnologías nuevas.
- Un proyecto retante, que espero seguir expandiendo en el futuro.
- Pienso que un lenguaje de programación así sencillo, y claro con muchas mejoras, podría ser considerado muy útil para la juventud de hoy en día.
- Me gustaría dar un demo de esto en un futuro a nuestra juventud, ya que podría ser de ayuda a motivar más jóvenes a estudiar programación.

Gracias por su atención!

Cindy Mendez

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