**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

PRÁCTICA 3

Calculadora científica con tabla de símbolos.

**Profesor:**

• **Tecla Parra Roberto**

**Alumno:**

* **Pacheco Delgado José Jacobeth**

**Grupo:**

• **3CM8**

**Materia:**

• **Compiladores**

**Introducción**

Esta practica involucra variables mediante una tabla de símbolos, para realizar operaciones arítmeticas más avanzadas que la practica 1. Es un mejora de la practica 1, también involura funciones como **sen, cos** etc…

**Objetivos**

* Tabla de símbolos para almacenar números racionales en una variable con un nombre con una longitud de más de una letra.
* Agregar builtins si es posible, con la función **seno**, **coseno**, **logaritmo**.

**Desarrollo**

Primero se debe modificar la gramática para poder relizar la asignación de variables, además se debe agregar una unión, principalmente para poder manejar las variables y los valores para los números racionales

**Hoc1.y**

%{

#include "racional.h"

#include "hoc.h"

#include <math.h>

#define MSDOS

void yyerror (char \*s);

*// int yylex ();*

void warning(char \*s, char \*t);

void execerror(char \*s, char \*t);

void fpecatch();

**extern** double Pow(double, double);

*// typedef double (\*apf)(double);*

*// static struct { Predefinidos*

*// char \*name;*

*// double (\*func)();*

*// } builtins\_[] = {*

*// "sin", sin,*

*// "cos" , cos,*

*// "atan", atan,*

*// "log", log,*

*// "log10", log10,*

*// "exp", exp,*

*// "sqrt", sqrt, /\* verifica rango \*/*

*// "int" , integer,*

*// //"abs", fabs,*

*// 0, 0*

*// };*

%}

%union {

double val;

char \* bltin;

Symbol \*sym;

RacionalAP racional\_;

}

%token <racional\_> racionalnum

%token <val> NUMBER

%token <sym> VAR BLTIN INDEF

%type <racional\_> asgn expr

%right '='

%left '+' '-'

%left '\*' '/'

%left UNARYMINUS

%right '^'

%% */\* A continuaci󮠬as reglas gramaticales y las acciones \*/*

list:

| list '\n'

| list asgn '\n' { printf("%s**\n**", "assing" ); }

| list expr '\n' { imprimirR( $2 );}

| list error '\n' { yyerrok; }

;

asgn: VAR '=' expr {$$=$1->u.val=$3; $1->type=VAR;}

;

expr: VAR { **if**($1->type == INDEF)

execerror("variable no definida ",$1->name);

$$=$1->u.val;

}

| asgn

| BLTIN '(' expr ')' {

printf("sen() = %lf**\n**",sin( $3 -> dec ) );

$$= creaRacional( 1,1,(\*($1->u.ptr))( $3 -> dec ), 0 );

}

| '(' expr ')' { $$ = $2;}

*//| '-' expr %prec UNARYMINUS { $$= -$2; }*

| racionalnum {

**if**($1 -> den == 0.0)

execerror("division por cero", "");

$$ = $1;

}

| expr '+' expr { $$ = racionalSuma( $1, $3 ); }

| expr '-' expr { $$ = racionalResta( $1, $3 ); }

| expr '\*' expr { $$ = racionalMultiplicar( $1, $3 ); }

| expr '/' expr { $$ = racionalDividir( $1, $3 ); }

;

%%

#include <stdio.h>

#include <ctype.h>

#include <signal.h>

#include <setjmp.h>

jmp\_buf begin;

char \*progname;

int lineno = 1;

int main (int argc, char \*argv[]){

progname=argv[0];

init();

setjmp(begin);

signal(SIGFPE, fpecatch);

yyparse ();

}

void execerror(char \*s, char \*t){

warning(s, t);

longjmp(begin, 0);

}

void fpecatch(){

execerror("excepcion de punto flotante", (char \*)0);

}

*// apf busca(){*

*// }//end busca*

void yyerror (char \*s) */\* Llamada por yyparse ante un error \*/*

{

warning(s, (char \*) 0);

}

void warning(char \*s, char \*t)

{

fprintf (stderr, "%s: %s", progname, s);

**if**(t)

fprintf (stderr, " %s", t);

fprintf (stderr, "cerca de la linea %d**\n**", lineno);

}

En el código de init.c se muestra la inicialización y los elementos con los que cuenta la tabla de símbolos.

Init.c

#include "racional.h"

#include "hoc.h"

#include "y.tab.h"

#include <math.h>

**extern** integer();

**static** **struct** { */\* Constantes \*/* char \*name; double cval;

} consts[] = {

"PI", 3.14159265358979323846,

"E", 2.71828182845904523536,

"GAMMA", 0.57721566490153286060, */\* Euler \*/*

"DEG", 57.29577951308232087680, */\* grado/radian \*/*

"PHI", 1.6180339887498948\*820, */\* proporcion dorada \*/*

0, 0

};

**static** **struct** { */\* Predefinidos \*/*

char \*name;

double (\*func)();

} builtins[] = {

"sin", sin,

"cos" , cos,

"atan", atan,

"log", log,

"log10", log10,

"exp", exp,

"sqrt", sqrt, */\* verifica rango \*/*

"int" , integer,

*//"abs", fabs,*

0, 0

};

init( ) */\* instalar constantes y predefinidos en la tabla \*/*

{

int i;

Symbol \*s;

RacionalAP \* r;

**for** (i = 0; consts[i].name; i++) {

r = creaRacional( 0, 1, consts[i].cval, 0 );

installR(consts[i].name, VAR, r );

} *// end for*

**for** (i = 0; builtins[i].name; i++) {

r = creaRacional( 0, 1, 0.0, 0 );

s = installR(builtins[i].name, BLTIN, r);

s->u.ptr = builtins[i].func;

}

}

Código símbolo.c para instalar variables e inicializar las funciones predifinidas en el siguiente código.

#include <math.h>

#include "racional.h"

#include "hoc.h"

#include "y.tab.h"

**static** Symbol \*symlist=0; */\* tabla de simbolos: lista ligada \*/*

Symbol \*lookup(char \*s) */\* encontrar s en la tabla de solos \*/*

{

Symbol \*sp;

**for** (sp = symlist; sp != (Symbol \*)0; sp = sp->next)

**if** (strcmp(sp->name, s)== 0)

**return** sp;

**return** 0; */\* 0 ==> no se encontr󠪯*

}

Symbol \*installR(char \*s,int t, RacionalAP r) */\* instalar s en la tabla de solos \*/*

{

Symbol \*sp;

char \*emalloc();

sp = (Symbol \*) emalloc(**sizeof**(Symbol));

sp->name = emalloc(strlen(s)+ 1) ; */\* +1 para '\0' \*/*

strcpy(sp->name, s);

sp->type = t;

sp->u.val = r;

sp->next = symlist; */\* poner al frente de la lista \*/*

symlist = sp;

**return** sp;

}

char \*emalloc(unsigned n) */\* revisar el regreso desde malloc \*/*

{

char \*p, \*malloc();

p = malloc(n);

**if**(p == 0)

execerror("out of memory", (char \*) 0);

**return** p;

}

Se hicieron modificaciones al archivo **racional**\_**calc**.c para obtener que las variables y funciones soportaran la estructura **racional.**

**E**n la **figura 1** se muestra la ejecución del programa, de igual forma se tuvo que modificar el main, para inicializar la tabla de símbolos con los built-ins y flex se agregó una expresión regular para el soporte de variables y almacenar los números racionales en la tabla de símbolos.

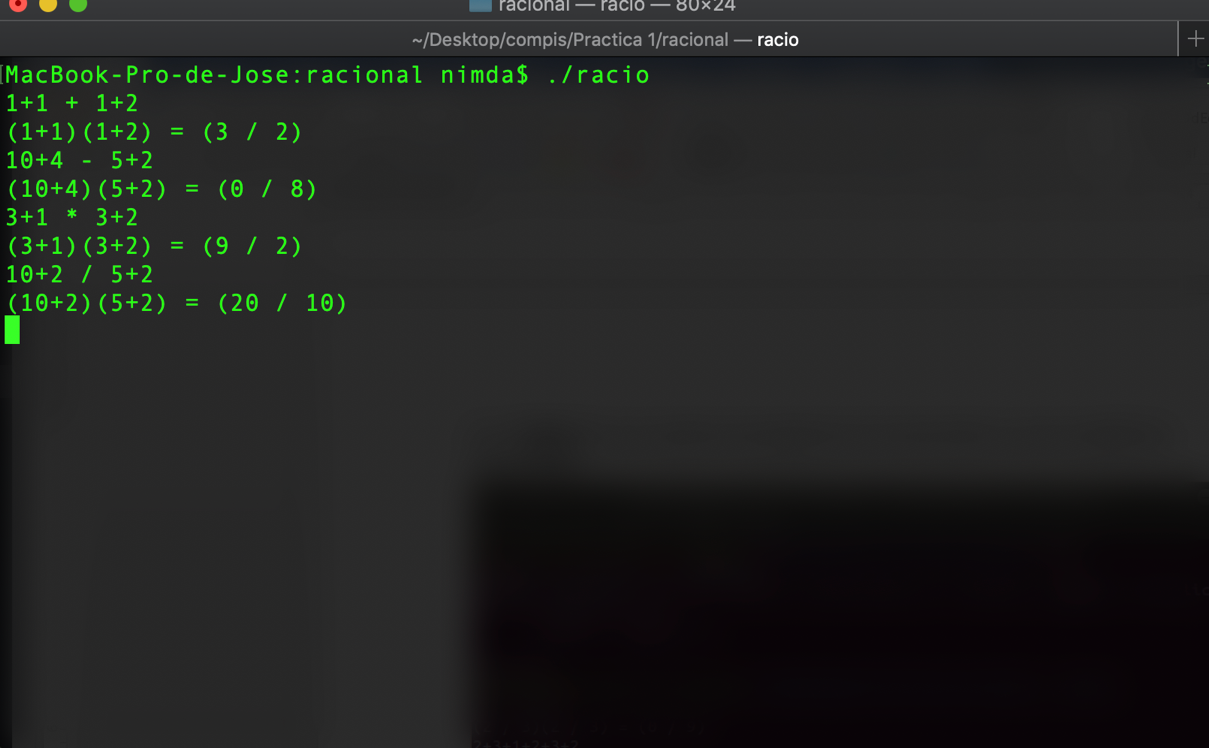
****

Figura 1 Ejecución de calculadora científica con tabla de símbolos.

**Conclusión**

Se demotro el éxito de implementar una tabla de símbolos para poder almacenar variables con números racionales, adémas de funciones mejor llamadas como “built-ins” para funciones trigonométricas como **seno** y **coseno.**