**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**Compiladores**

**“Práctica 3: Tabla de símbolos”**

**Por:**

**Ramírez Arce Edwin Ricardo**

**Grupo:**

**3CM9**

Profesor:

Tecla Parra Roberto

Junio 2017

**PLANTEAMIENTO DEL PROBLEMA**

* *Implementar un compilador básico capaz de realizar operaciones booleanas (AND, OR y NOT) con una tabla de símbolos y variables igualandolas con los operadores binarios.*

**IMPLEMENTACIÓN DE LA SOLUCIÓN**

%{

#include <stdio.h>

#include <ctype.h>

//#include <malloc.h>

#include <stdlib.h>

#include <string.h>

#include "booleano.h"

#include <math.h>

#include "y.tab.h"

**void** yyerror(**char** \*s);

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

**int** yylex(**void**);

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

%}

%**union** {

NodoArb\* val;

Symbol \*sym;

}

%left '|'

%left '&'

%left UNARYMINUS

%token <val> NUMBER

%token <sym> VAR BLTIN INDEF

%type <val> expr asgn

%%

list:

| list '\n'

| list asgn '\n'

| list expr '\n' {

imprimeArb($2, imprimeCadena);

printf("%d \n",evalua($2));

}

;

asgn: VAR '=' expr {

$$=$1->u.val=$3;

//$1->type=VAR; hubo cambio en yylex

}

;

expr: NUMBER { $$=$1; }

| VAR {

//printf("VAR nombre=(%s)\n",$1->name);

$$=$1->u.val;

}

| expr '&' expr { $$ = creaNodoArb(strdup("&&"),$1, $3); }

| expr '|' expr { $$ = creaNodoArb(strdup("||"),$1, $3); }

|'!' expr %prec UNARYMINUS { $$ =creaNodoArb(strdup("!"), $2, NULL); }

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

;

%%

//ejemplo cadena: T&F&T&F&T&T&T&T

//ejemplo cadena: a&b&c&d&e&g

**char** \*progname;

**int** lineno = 1;

**int** NumVar=0;

**int** contador=0;

**void** imprimeCadena(**void** \*p){

puts((**char**\*)p);

}

**int** yylex() {

**int** c;

**int** val;

**while**((c=getchar()) == ' ' || c == '\t')

;

**if**(c == EOF)

**return** 0;

// si no se permiten numeros

**if**(isdigit(c)) {

**char** sbuf[100];

ungetc(c, stdin);

scanf("%d",&val);

sprintf(sbuf,"%d", val);

yylval.val=creaNodoArb(strdup(sbuf), NULL, NULL);

**return** NUMBER;

}

**if**(isalpha(c)){

Symbol \*s;

NodoArb\* n;

**char** sbuf[200], \*p=sbuf;

**if**(c=='F' || c=='f'){

yylval.val=creaNodoArb(strdup("0"), NULL, NULL);

**return** NUMBER;

} **else** **if**(c=='T' || c=='t'){

yylval.val=creaNodoArb(strdup("1"), NULL, NULL);

**return** NUMBER;

}

//puts("Antes del do");

**do** {

\*p++=c;

} **while** ((c=getchar())!=EOF && isalnum(c));

ungetc(c, stdin);

\*p='\0';

//printf("lex sbuf=(%s) ",sbuf);

**if**((s=lookup(sbuf))==(Symbol \*)NULL){

//printf("lex IF sbuf=(%s) ",sbuf);

n= creaNodoArb(strdup(sbuf), NULL, NULL);

s=install(sbuf, VAR, n);// VAR era INDEF

contador++;

//printf("Varibables: %d\n",contador);

}

yylval.sym=s;// cambio importante

//en el lab lo teniamos como yylval.val=n;

**return** s->type;

}

**if**(c == '\n')

lineno++;

**return** c;

}

**void** yyerror(**char** \*s) {

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

}

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

warning(s, t);

}

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

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

**if**(t)

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

fprintf(stderr, " near line %d\n", lineno);

}

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

progname = argv[0];

yyparse();

printf("Varibables: %d\n",contador);

}

NodoArb \*creaNodoArb(**void** \*dato,NodoArb \*izq,NodoArb \*der){

NodoArb \*nvo;

nvo=(NodoArb \*)malloc(**sizeof**(NodoArb));

**if**(!nvo){

puts("no hay memoria para crear NodoL");

**return** (NodoArb \*)NULL;

}

nvo->izq=izq;

nvo->der=der;

nvo->info=dato;

**return** nvo;

}

**void** imprimeArb(NodoArb \*raiz, **void** (\*f)(**void** \*)){

**if**(!raiz)

**return**;

imprimeArb(raiz->izq,f);

(\*f)(raiz->info);

imprimeArb(raiz->der,f);

}

**int** evalua(NodoArb \*r){

**char** info=((**char** \*)(r->info))[0];

**if**(isdigit(info)) **return** info-'0';

**if**(info == '|') **return** evalua(r->izq) || evalua(r->der);

**else** **if**(info == '&') **return** evalua(r->izq) && evalua(r->der);

**else** **if**(info == '!') **return** !evalua(r->izq);

}

**FUNCIONAMIENTO DE LA SOLUCIÓN**

