ITD **TecNM**

Ing. en Sistemas Computacionales

Prof. Elda Rivera Sauceda

Lenguajes y Autómatas II

Grupo: 7Y

Manual Técnico y de Usuario

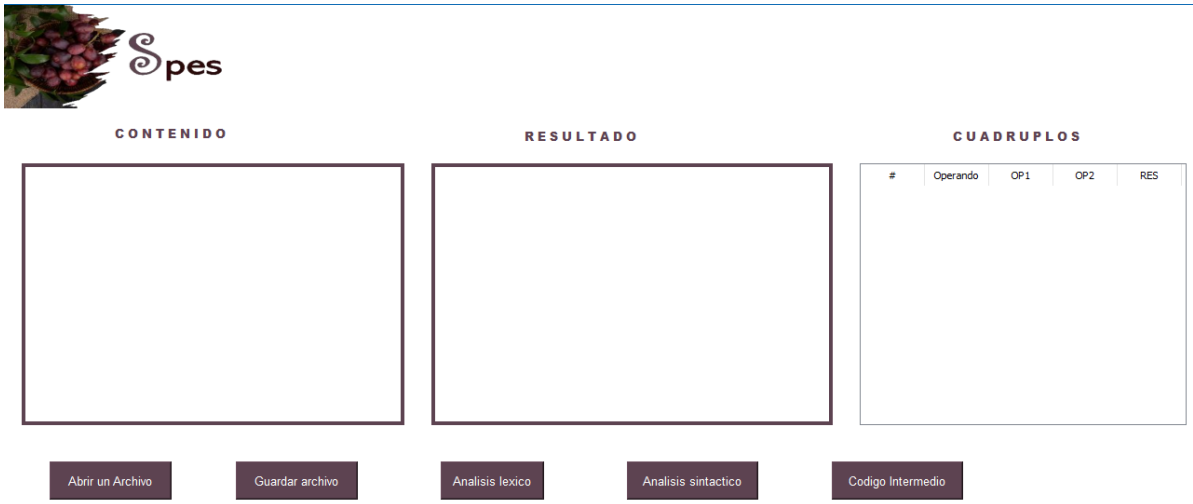
Diciembre 3 del 2021

Luis Alberto Jiménez Soto – 18041271

Carlos Alberto Soto Molina – 18041314

Manual de Usuario

Este software hace uso de un solo interfaz gráfico.



Dentro de este se tienen dos cajas de texto, una tabla y cinco botones.

Cajas de Texto

Contenido

La caja de texto denominada contenido, es donde se introduce el código del lenguaje a compilar.

Resultados

La caja de texto denominado resultados es donde se lista los resultados de las etapas de análisis léxico y sintáctico, aunque también se usa para el código intermedio (imprime los errores que se presenten).

Tabla

Cuádruplos

La tabla de la derecha se encarga de imprimir los diversos cuádruplos que se generan en el código intermedio.

Abrir archivo

Botones

Abrir Archivo

El uso de este botón es opcional, dado que se puede introducir código de manera manual en la caja de texto “Contenido”, de lo contrario se puede cargar un archivo con este botón.

Guardar Archivo

Este botón genera un archivo con el contenido de la caja de texto “Contenido”; el cual se almacena.

Análisis Léxico

Este botón ejecuta la etapa del análisis léxico, posteriormente imprime los resultados de este en la caja de texto “Resultado”; esto siendo los tokens reconocidos y su tipo.

Análisis Sintáctico

Al igual que botón previo, este ejecuta un análisis, solo que este es el sintáctico. Posteriormente te notifica del tipo de terminación (correcta/ incorrecta) con una ventaja y te va imprimiendo los tokens que analizar hasta terminar. Dado el caso que llegue a un error, termina el proceso y te indica lo que se esperaba para completar la estructura

Código Intermedio

Este botón ejecuta las últimas dos etapas, el análisis semántico y el código intermedio. En base a dicho análisis imprime los errores de la etapa semántica; dentro de la caja de texto “Resultados”. Con respecto a la generación de código intermedio, imprime los errores entre operaciones y también lista los cuádruplos en la tabla.

Manual Técnico

Objetivo

La creación de un software que sea una continuación del previo proyecto de análisis léxico y sintáctico; y sea capaz de encargarse de las etapas de análisis semántico y generación de código intermedio. De tal manera que te pueda identificar errores entre los tipos de datos que tenemos y generar los cuádruplos de código intermedio.

Requerimientos de Hardware (de Windows 10)

- Procesador de 1GHz o mas
- 1GB de memoria para 32 bit (2GB para 64 bit)
- 50GB de almacenamiento
- Tarjeta gráfica (con manejadores de DirectX 9)

Software Usado

QT Creator

Descripción Código

void MainWindow::**on_pushButton_clicked**()

Este método es el evento del botón de abrir un archivo

void MainWindow::**on_pushButton_2_clicked**()

Método de evento del boton de guardar un archivo

int **relaciona**(char c)

Método que toma un carácter y en base a su valor retorna su tipo (la cadena del tipo).

int **Token**(int e,string token)

Método que toma un token y en base a su valor retorna su tipo (el número del tipo).

int **Error**(int e)

Toma un número y en base a tal te retorna el tipo de error (número de error) que se presenta.

void MainWindow::**on_btnAnaliza_clicked**()

Método de evento del botón que empieza el análisis léxico.

void **GetToken**()

Método que toma estado y columna y en base a tales devuelve el siguiente estado; esto se va repitiendo mientras siga el análisis.

int **RelacionaToken**(string token)

Retorna el estado en el que está, en base a la cadena que se le pasa.

void MainWindow::**on_btnAnaliza_2_clicked**()

Método de evento del botón que empieza el análisis sintáctico.

void MainWindow::**on_btnAnaliza_3_clicked**()

Método de evento del botón que empieza el análisis semántico y la generación del código intermedio.

QString **CuadruplosOperaciones**(QString Op1, QString Op2,QString Oper)

Se encarga de la validación de operaciones, es decir revisa los tipos de datos y que sea factible dicha operación.

void MainWindow::**Semantico**()

método encargado de hacer el análisis semántico. Este va recorriendo la cadena de código y en base al token actual va haciendo la operación correcta. En si la mayoría de la lógica se encuentra dentro de este método.

void MainWindow::**estatutoAsig**(QString Token, QString TokenTexto)

Se encarga de la generación de los cuádruplos de asignación.

void MainWindow::**generarCuadruplo**()

Método que se enfoca en la generación de los cuádruplos de comparación y operaciones.

void MainWindow::**cuadrupoSaltoFalso**()

Generación del cuádruplo del salto en falso.

void MainWindow::**cuadrupoSaltoVerdadero**()

Generación del cuádruplo de salto verdadero.

void MainWindow::**cuadruploSaltoIncondicional**()

Generación del cuádruplo del salto incondicional.

void MainWindow::**llenarSaltoTope**()

Se encarga de actualizar los saltos verdaderos, falsos e incondicional en base a la pila de saltos.

QString **existeVariable**(string variable)

Método que hace la valida que no exista la variable que se está tratando de definir.

Código

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QFileDialog>
#include <QMessageBox>
#include <QDir>
#include <QFile>
#include <QTextStream>
#include <QPixmap>
#include <QStack>
#include <iostream>
```

```
#include <stdio.h>
#include <string.h>
#include <cstdio>

using namespace std;
MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}
void Relaciona(char a);
int Token(int e,string token);
int Error(int t);
void  GetToken();
void Semantico();
QString cadenaAnalizar;
int RelacionaToken(string token);
void generarCuadruplo();
void realizarOperacion();
void estatutoAsig(QString Token, QString TokenTexto);
QString existeVariable(string variable);
static int cont_cadena_posicion = 0;
static QStack<int> Pila,ResultadosTokens,pilaResultadoTokens;
static QStack<string> Tokens,pilaTokens,pilaResultadoTokensTexto;
static QStack<string> Variables,VariablesTipo;

//global var
QString cuadroResultado;
QString typeOp1,typeOp2,typeRes;
QStack<QString> operandos,operandosTipos;
QStack<QString> operadores;
QStack<int> saltos;
string cToken;
int cont=0;
static int contCuad=1;
static int contR=1;
static bool ban1=0;
static bool banwhile= 0;
static bool banfor=0;
static bool bando=0;
static bool semantico=0;

int producciones[123][123]={{1,2},
                            {1000,1004,1002, 1},
                            {-1},
                            {1003,1004,1002,3,4,1005},
                            {1006,56,12,1008,55,1002,3},
                            {-1},
                            {6,5},
                            {1022,1023,7,1024,8,17,1025},
                            {1026,1004,1008,55,1023,7,1024,8,17,1027,6},
                            {-1},
                            {12,1008,55,1002,7},
                            {-1},
                            {1028,9,1029},
                            {-1},
                            {12,1008,55,1002,9},
                            {-1},
                            {1004,11},
                            {14},
                            {-1},
```

```
{10,13},
{1007,10,13},
{-1},
{1009,15,1010},
{1012,1011,1012,16},
{1007,1012,1011,1012,16},
{-1},
{18},
{57,1002,18},
{21,1002,18},
{31,1002,18},
{20,1002,18},
{19,1002,18},
{24,1002,18},
{32,1002,18},
{27,1002,18},
{30,1002,18},
{-1},
{1038,17,1042,1023,40,1024,1043},
{1035,1004,1036,40,1037,40,1038,17,1039},
{1044,1023,40,1024,17,22,1047},
{1045,1023,40,1024,17,22,23},
{-1},
{1046,17},
{-1},
{1033,40},
{1034,40},
{40,25},
{1033},
{1034},
{34,54,40},
{1030,1023,28,1024},
{1004,29},
{-1},
{1007,1004,29},
{1031,1023,42,1024},
{1040,1023,40,1024,17,1041},
{1032,40},
{1004,34},
{35},
{-1},
{1009,42,1010},
{38,37},
{1059,38,37},
{1060,38,37},
{1061,38,37},
{-1},
{1004,39},
{1012},
{1062},
{1063},
{1064},
{1065},
{1023,40,1024},
{44,41},
{1048,44,41},
{-1},
{40,43},
{1007,40,43},
{-1},
{46,45},
```

```
                                {1049,46,45},
                                {-1},
                                {1050,47},
                                {47},
                                {49,48},
                                {53,49},
                                {-1},
                                {36,50},
                                {1057,36,50},
                                {1058,36,50},
                                {-1},
                                {1023,51,1024},
                                {-1},
                                {1004,52},
                                {-1},
                                {-1},
                                {1007,1004,52},
                                {1051},
                                {1052},
                                {1053},
                                {1054},
                                {1055},
                                {1056},
                                {1036},
                                {1066},
                                {1067},
                                {1068},
                                {1069},
                                {1070},
                                {1013},
                                {1014},
                                {1015},
                                {1016},
                                {1017},
                                {1018},
                                {1019},
                                {1020},
                                {1021},
                                {-1},
                                {1004,58},
                                {26},
                                {25}};

int M[27][32]=
{{1,1,3,511,22,1,1,13,14,15,23,26,16,19,21,17,18,12,20,131,129,137,138,139,
140,0,0,0,11,9,511,511},

{1,1,1,2,100,1,1,100,100,100,100,100,100,100,100,100,100,100,100,100,100,10
0,100,100,100,100,100,100,100,100,100,100},

{1,1,1,2,500,1,1,500,500,500,500,500,500,500,500,500,500,500,500,500,500,50
0,500,500,500,500,500,500,500,500,500,500},

{101,101,3,101,4,101,101,101,101,101,101,101,101,101,101,101,101,101,101,10
1,101,101,101,101,101,101,101,101,101,101,101,101},

{501,501,5,501,501,501,501,501,501,501,501,501,501,501,501,501,501,501,501,
501,501,501,501,501,501,501,501,501,501,501,501,501},

{102,102,5,102,102,6,6,102,102,102,102,102,102,102,102,102,102,102,102,102,
102,102,102,102,102,102,102,102,102,102,102,102},
```

{502,502,8,502,502,502,502,7,7,502,502,502,502,502,502,502,502,502,502,502,
502,502,502,502,502,502,502,502,502,502,502,502,502},

{503,503,8,503,503,503,503,503,503,503,503,503,503,503,503,503,503,503,503,
503,503,503,503,503,503,503,503,503,503,503,503,503,503,503},

{103,103,8,103,103,103,103,103,103,103,103,103,103,103,103,103,103,103,103,
103,103,103,103,103,103,103,103,103,103,103,103,103,103,103},

{10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
,504,504,10,10,507,504,504},

{505,505,505,505,505,505,505,505,505,505,505,505,505,505,505,505,505,505,505,50
5,505,505,505,505,505,505,505,505,505,505,505,104,505,505},

{11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11
,11,11,11,105,11,11,506},

{106,106,106,106,106,106,106,106,106,106,106,106,106,106,106,106,106,107,10
6,106,106,106,106,106,106,106,106,106,106,106,106,106,106},

{108,108,108,108,108,108,108,110,108,108,108,108,108,108,108,108,108,109,10
8,108,108,108,108,108,108,108,108,108,108,108,108,108,108},

{111,111,111,111,111,111,111,111,113,111,111,111,111,111,111,111,111,112,11
1,111,111,111,111,111,111,111,111,111,111,111,111,111,111},

{114,114,114,114,114,114,114,114,114,116,114,114,114,114,114,114,114,115,11
4,114,114,114,114,114,114,114,114,114,114,114,114,114,114},

{117,117,117,117,117,117,117,117,117,117,117,117,117,117,117,117,117,118,11
7,117,117,117,117,117,117,117,117,117,117,117,117,117,117},

{119,119,119,119,119,119,119,119,119,119,119,119,119,119,119,119,119,120,11
9,119,119,119,119,119,119,119,119,119,119,119,119,119,119},

{121,121,121,121,121,121,121,121,121,121,121,121,121,121,121,121,121,122,12
1,121,121,121,121,121,121,121,121,121,121,121,121,121,121},

{512,512,512,512,512,512,512,512,512,512,512,512,512,124,512,512,512,512,51
2,512,512,512,512,512,512,512,512,512,512,512,512,512,512},

{513,513,513,513,513,513,513,513,513,513,513,513,513,513,513,513,513,513,12
6,513,513,513,513,513,513,513,513,513,513,513,513,513,513},

{127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,128,12
7,127,127,127,127,127,127,127,127,127,127,127,127,127,127},

{508,508,508,508,130,508,508,508,508,508,508,508,508,508,508,508,508,508,50
8,508,508,508,508,508,508,508,508,508,508,508,508,508,508},

{132,132,132,132,132,132,132,132,132,24,132,132,132,132,132,132,132,133,132
,132,132,132,132,132,132,132,132,132,132,132,132,132,132},

{24,24,24,24,24,24,24,24,24,25,24,24,24,24,24,24,24,24,24,24,24,24,24,24,24
,24,24,24,24,24,24,510},

{24,24,24,24,24,24,24,24,24,24,134,24,24,24,24,24,24,24,24,24,24,24,24,24,2
4,24,24,24,24,24,24,510},

```
{26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26
,26,136,26,26,26,26,136}};
string tokenExitoso[]={"Identificador",
                       "Constantes numérica entera",
                       "Constantes numérica real",
                       "Constantes numérica de notación científica",
                       "Constante carácter",
                       "Constante string",
                       "Operador de asignacion  = Igual",
                       "Operador Relacional == Es igual",
                       "Operador aritmético + Suma",
                       "Operador de asignacion += Adicion",
                       "Operador aritmético ++ Incremento",
                       "Operador aritmético - Resta",
                       "Operador de asignacion -= Sustraccion",
                       "Operador aritmético -- Decremento",
                       "Operador aritmético * Multiplicacion",
                       "Operador de asignacion *= ",
                       "Operador aritmético  ** ",
                       "Operador aritmético % Porcentaje",
                       "Operador de asignacion %=",
                       "Operador Relacional < Menor que",
                       "Operador Relacional <= Menor igual",
                       "Operador Relacional > Mayor que",
                       "Operador Relacional >= Mayor que",
                       "Operador lógico && And",
                       "Operador lógico || Or",
                       "Operador lógico ! Not",
                       "Operador Relacional != Diferente",
                       "Signo de puntacion ; Punto y coma",
                       "Signo de puntacion .. Punto punto",
                       "Signo de puntacion , coma",
                       "Operador aritmético / División",
                       "Operador de asignacion /=",
                       "Comentario /* */",
                       "Comentario # Simple",
                       "Signo de agrupacion [ Corchete que abre",
                       "Signo de agrupacion ] Corchete que cierra",
                       "Signo de agrupacion ( Parentesis que abre",
                       "Signo de agrupacion ) Parentesis que cierra",
                       "Fin del archivo"};
string tokenError[]={"Identificador no puede terminar en '_' ",
                     "Dato numerico  no puede terminar en '.' esta se
esperaba otro numero",
                     "Dato numerico  no puede terminar en 'E' o 'e' se
esperaba otro numero o algún signo + o -",
                     "Dato numerico  no puede terminar en '+' o '-' se
esperaba algun numero",
                     "Constante carácter debe contenter un elemento y
terminar en comilla simple",
                     "Constante carácter debe en terminar en comilla
simple",
                     "Constante string debe de terminar en  comillas dobles
",
                     "Constante carácter debe de contener al menos un
carácter",
                     "Signo de puntacion esperaba un '.'",
                     "El comentario esperaba un '*/' para terminar",
                     "Carácter no reconocido",
                     "Operedor logico esperaba &",
```

```
                        "Operedor logico esperaba |"};
string palabrasResevadas[]={"class",
                            "endclass",
                            "int",
                            "float",
                            "char",
                            "string",
                            "bool",
                            "if",
                            "else",
                            "elseif",
                            "endif",
                            "do",
                            "to",
                            "eval",
                            "enddo",
                            "while",
                            "endwhile",
                            "read",
                            "write",
                            "def",
                            "as",
                            "for",
                            "endfor",
                            "private",
                            "public",
                            "void",
                            "protected",
                            "library",
                            "func",
                            "endfunc",
                            "main",
                            "endmain",
                            "true",
                            "false"};
//Matriz predictiva

int MP [72][72] =
{{0,1500,1500,0,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500
,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500
,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500
,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500
,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500},

{1,1500,1500,2,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,
1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,
1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,
1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,
1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500},

{1500,1500,1500,3,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,15
00,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,15
00,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,15
00,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,15
00,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500},

{1502,1502,1502,1502,1502,1502,4,1502,1502,1502,1502,1502,1502,1502,1502,1502,15
02,1502,1502,1502,1502,1502,1502,5,1502,1502,1502,5,1502,1502,1502,1502,1502,150
2,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,150
2,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,1502,150
2,1502,1502,1502,1502,1502,1502,1502,1502,1502,5},
```

{1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503
,1503,1503,1503,1503,1503,1503,1503,6,1503,1503,1503,6,1503,1503,1503,1503,
1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,
1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,
1503,1503,1503,1503,1503,1503,1503,1503,1503,1503,1503},

{1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504
,1504,1504,1504,1504,1504,1504,1504,7,1504,1504,1504,1504,1504,1504,1504,15
04,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,15
04,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,15
04,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504,1504},

{1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505
,1505,1505,1505,1505,1505,1505,1505,9,1505,1505,1505,8,1505,1505,1505,1505,
1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,
1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,
1505,1505,1505,1505,1505,1505,1505,1505,1505,1505,9},

{1506,1506,1506,1506,10,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,1
506,1506,1506,1506,1506,1506,1506,1506,1506,11,1506,1506,1506,1506,1506,150
6,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,150
6,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,150
6,1506,1506,1506,1506,1506,1506,1506,1506,1506,1506,11},

{1507,1507,1507,1507,13,1507,1507,1507,1507,1507,1507,1507,13,1507,1507,150
7,1507,1507,1507,1507,1507,1507,1507,13,1507,13,1507,13,12,1507,13,13,13,13
,13,13,1507,1507,13,13,13,13,13,1507,13,1507,13,13,1507,1507,13,1507,1507,1
507,1507,1507,1507,1507,1507,1507,1507,1507,13,13,13,13,1507,1507,1507,1507
,1507,13},

{1507,1507,1507,1507,14,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1
507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,15,150
7,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,150
7,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,150
7,1507,1507,1507,1507,1507,1507,1507,1507,1507,1507,15},

{1508,1508,1508,1508,16,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1
508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1
508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1
508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1
508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508},

{1508,1508,1508,1508,1508,1508,1508,18,18,17,1508,1508,1508,1508,1508,1508,
1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,
1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,
1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,
1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,18},

{1508,1508,1508,1508,19,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1
508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1
508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1
508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1
508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508},

{1508,1508,1508,1508,1508,1508,1508,20,21,1508,1508,1508,1508,1508,1508,150
8,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,150
8,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,150
8,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,150
8,1508,1508,1508,1508,1508,1508,1508,1508,1508,1508,21},

{1509,1509,1509,1509,1509,1509,1509,1509,1509,22,1509,1509,1509,1509,1509,1
509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1
509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1
509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1
509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509},

{1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,23,1509,1509,1509,1
509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1
509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1
509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1
509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509,1509},

{1510,1510,1510,1510,1510,1510,1510,24,1510,1510,25,1510,1510,1510,1510,151
0,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,151
0,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,151
0,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,151
0,1510,1510,1510,1510,1510,1510,1510,1510,1510,1510,25},

{1511,1511,1511,1511,26,1511,1511,1511,1511,1511,1511,1511,26,1511,1511,151
1,1511,1511,1511,1511,1511,1511,1511,26,1511,26,1511,26,1511,1511,26,26,26,
26,26,26,1511,1511,26,26,26,26,26,1511,26,26,26,26,1511,1511,26,1511,1511,1
511,1511,1511,1511,1511,1511,1511,1511,1511,26,26,26,26,1511,1511,1511,1511
,1511,26},

{1511,1511,1511,1511,27,1511,1511,1511,1511,1511,1511,1511,32,1511,1511,151
1,1511,1511,1511,1511,1511,1511,1511,32,1511,36,1511,36,1511,1511,34,35,33,
32,32,30,1511,1511,31,36,29,36,36,1511,28,40,36,36,1511,1511,32,1511,1511,1
511,1511,1511,1511,1511,1511,1511,1511,1511,32,32,32,32,1511,1511,1511,1511
,1511,36},

{1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512
,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512
,1512,1512,1512,1512,1512,1512,1512,1512,37,1512,1512,1512,1512,1512,1512,1
512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1
512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512,1512},

{1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513
,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513
,1513,1513,1513,1513,1513,1513,38,1513,1513,1513,1513,1513,1513,1513,1513,1
513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1
513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513,1513},

{1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514
,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514
,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,39,1
514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1
514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514,1514},

{1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515
,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515
,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515
,40,42,41,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,
1515,1515,1515,1515,1515,1515,1515,1515,1515,1515,41},

{1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516
,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516
,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516
,1516,42,43,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,151
6,1516,1516,1516,1516,1516,1516,1516,1516,1516,1516,43},

{1517,1517,1517,1517,46,1517,1517,1517,1517,1517,1517,1517,46,1517,1517,151
7,1517,1517,1517,1517,1517,1517,1517,46,1517,1517,1517,1517,1517,1517,1517,
1517,1517,44,45,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517
,1517,1517,1517,46,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,4
6,46,46,46,1517,1517,1517,1517,1517,1517},

{1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517
,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517
,1517,1517,1517,47,48,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,151
7,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,151
7,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517,1517},

{1518,1518,1518,1518,1518,1518,1518,1518,1518,49,1518,1518,1518,1518,1518,1
518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1
518,1518,1518,1518,1518,1518,49,1518,1518,1518,1518,1518,1518,1518,1518,151
8,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,151
8,1518,1518,1518,1518,1518,49,49,49,49,49,49},

{1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519
,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519
,50,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1
519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1
519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519},

{1519,1519,1519,1519,51,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1
519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1
519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1
519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1
519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519},

{1519,1519,1519,1519,1519,1519,1519,53,1519,1519,1519,1519,1519,1519,1519,1
519,1519,1519,1519,1519,1519,1519,1519,1519,52,1519,1519,1519,1519,1519,151
9,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,151
9,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,151
9,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519,1519},

{1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520
,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520
,1520,54,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1
520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1
520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520,1520},

{1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521
,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521
,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,55,1521,1521,1521,1521,1
521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1
521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521,1521},

{1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522
,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522
,1522,1522,56,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1
522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1
522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522,1522},

{1518,1518,1518,1518,57,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1
518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1
518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1
518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1
518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518},

{1518,1518,1518,1518,1518,1518,1518,1518,1518,58,1518,1518,1518,1518,1518,1
518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1
518,1518,1518,1518,1518,1518,59,1518,1518,1518,1518,1518,1518,1518,1518,151
8,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,1518,151
8,1518,1518,1518,1518,1518,59,59,59,59,59,59},

{1523,1523,1523,1523,1523,1523,1523,1523,1523,60,1523,1523,1523,1523,1523,1
523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1
523,1523,1523,1523,1523,1523,49,1523,1523,1523,1523,1523,1523,1523,1523,152
3,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,152
3,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,152
3,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523,1523},

{1524,1524,1524,1524,61,1524,1524,1524,1524,1524,1524,1524,61,1524,1524,152
4,1524,1524,1524,1524,1524,1524,1524,61,1524,1524,1524,1524,1524,1524,1524,
1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,
1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,1524,
1524,61,61,61,61,1524,1524,1524,1524,1524,1524},

{1524,1524,65,1524,65,1524,1524,65,1524,1524,65,1524,65,1524,1524,1524,1524
,1524,1524,1524,1524,1524,1524,65,65,1524,1524,1524,1524,1524,1524,1524,152
4,1524,1524,1524,1524,65,65,1524,1524,1524,1524,1524,1524,1524,1524,1524,65
,65,65,65,65,65,65,65,65,65,62,63,64,65,65,65,65,1524,1524,1524,1524,152
4,65},

{1525,1525,1525,1525,66,1525,1525,1525,1525,1525,1525,1525,67,1525,1525,152
5,1525,1525,1525,1525,1525,1525,1525,72,1525,1525,1525,1525,1525,1525,1525,
1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,
1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,1525,
1525,68,69,70,71,1525,1525,1525,1525,1525,1525},

{1526,1526,92,1526,1526,1526,1526,92,1526,1526,92,1526,92,1526,1526,1526,15
26,1526,1526,1526,1526,1526,1526,91,92,1526,1526,1526,1526,1526,1526,1526,1
526,1526,1526,1526,1526,92,92,1526,1526,1526,1526,1526,1526,1526,1526,1526,
92,92,92,92,92,92,92,92,92,92,92,92,92,92,92,92,92,92,1526,1526,1526,1526,1
526,92},

{1527,1527,1527,1527,73,1527,1527,1527,1527,1527,1527,1527,73,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,73,1527,1527,1527,1527,1527,1527,1527,
1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,
1527,1527,1527,1527,73,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,15
27,73,73,73,73,1527,1527,1527,1527,1527,1527},

{1527,1527,75,1527,1527,1527,1527,75,1527,1527,75,1527,1527,1527,1527,1527,
1527,1527,1527,1527,1527,1527,1527,1527,75,1527,1527,1527,1527,1527,1527,15
27,1527,1527,1527,1527,1527,75,75,1527,1527,1527,1527,1527,1527,1527,1527,1
527,74,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,75},

{1527,1527,1527,1527,76,1527,1527,1527,1527,1527,1527,1527,76,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,76,1527,1527,1527,1527,1527,1527,1527,
1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,
1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,
1527,76,76,76,76,1527,1527,1527,1527,1527,1527},

{1527,1527,1527,1527,1527,1527,1527,77,1527,1527,78,1527,1527,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,78,1527,1527,1527,1527,1527,1527,
1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,
1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,
1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,79},

{1527,1527,1527,1527,79,1527,1527,1527,1527,1527,1527,1527,79,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,152
7,1527,1527,1527,1527,79,1527,1527,1527,1527,1527,1527,1527,1527,1527,
1527,79,79,79,79,1527,1527,1527,1527,1527,1527},

{1527,1527,81,1527,81,1527,1527,81,1527,1527,81,1527,81,1527,1527,1527,1527
,1527,1527,1527,1527,1527,1527,81,81,1527,1527,1527,1527,1527,1527,1527,152
7,1527,1527,1527,1527,81,81,1527,1527,1527,1527,1527,1527,1527,1527,1527,81
,80,81,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,81,81,81,81,1
527,1527,1527,1527,1527,81},

{1527,1527,1527,1527,83,1527,1527,1527,1527,1527,1527,1527,83,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,83,1527,1527,1527,1527,1527,1527,1527,
1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,
1527,1527,1527,1527,82,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,15
27,83,83,83,83,1527,1527,1527,1527,1527,1527},

{1527,1527,1527,1527,84,1527,1527,1527,1527,1527,1527,1527,84,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,152
7,1527,84,84,84,84,1527,1527,1527,1527,1527,1527},

{1527,1527,86,1527,86,1527,1527,86,1527,1527,86,1527,86,1527,1527,1527,1527
,1527,1527,1527,1527,1527,1527,86,86,1527,1527,1527,1527,1527,1527,1527,152
7,1527,1527,1527,1527,86,86,1527,1527,1527,1527,1527,1527,1527,1527,1527,86
,86,86,85,85,85,85,85,85,1527,1527,1527,1527,1527,86,86,86,86,1527,1527,152
7,1527,1527,86},

{1527,1527,1527,1527,87,1527,1527,1527,1527,1527,1527,1527,87,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,152
7,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,1527,152
7,1527,87,87,87,87,1527,1527,1527,1527,1527,1527},

{1527,1527,90,1527,90,1527,1527,90,1527,1527,90,1527,90,1527,1527,1527,1527
,1527,1527,1527,1527,1527,1527,90,90,1527,1527,1527,1527,1527,1527,1527,152
7,1527,1527,1527,1527,90,90,1527,1527,1527,1527,1527,1527,1527,1527,1527,90
,90,90,90,90,90,90,90,90,88,89,1527,1527,1527,90,90,90,90,1527,1527,1527,15
27,1527,89},

{1526,1526,1526,1526,93,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1
526,1526,1526,1526,1526,1526,1526,1526,1526,94,1526,1526,1526,1526,1526,152
6,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,152
6,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,152
6,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,1526,94},

{1526,1526,1526,1526,1526,1526,1526,96,1526,1526,1526,1526,1526,1526,1526,1
526,1526,1526,1526,1526,1526,1526,1526,1526,95,1562,1562,1562,1562,1562,156
2,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,156
2,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,156
2,1562,1562,1562,1562,1562,1562,1562,1562,1562,1562,95},

{1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528
,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528
,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528
,1528,1528,1528,1528,1528,1528,97,98,99,100,101,102,1528,1528,1528,1528,152
8,1528,1528,1528,1528,1528,1528,1528,1528,1528,1528},

```
{1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529
,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529
,1529,1529,1529,1529,1529,1529,103,1529,1529,1529,1529,1529,1529,1529,1529,
1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,1529,
1529,1529,1529,1529,1529,1529,104,105,106,107,108,1529},

{1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,109,110,1
11,112,113,114,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,
1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,
1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,
1530,1530,1530,1530,1530,1530,1530,1530,1530,1530,1530},

{1531,1531,1531,1531,118,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,
1531,1531,1531,1531,115,116,117,1531,1531,1531,1531,1531,1531,1531,1531,153
1,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,153
1,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,153
1,1531,1531,1531,1531,1531,1531,1531,1531,1531,1531,118},

{1511,1511,1511,1511,119,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,
1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,
1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,
1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,
1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511},

{1511,1511,120,1511,1511,1511,1511,1511,1511,120,1511,1511,1511,1511,1511,1
511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1
511,1511,1511,121,121,1511,120,1511,1511,1511,1511,1511,1511,1511,1511,1511
,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511,1511
,1511,1511,1511,1511,1511,120,120,120,120,120,120}};

string ErroresAnalisis[] = {"Inicio invalido, revise su entrada",
                            "Archivo esta vacio",
                            "Declaracion incorrecta, revise su sintaxis",
                            "Debe existir un main",
                            "Main, parametros incorrectos",
                            "Declaracion de funcion incorecta",
                            "Parametros incorrectos, revise su sintaxis",
                            "Declaracion de varibles locales incorrecta",
                            "Declaracion de id incorrecta",
                            "Declaracion de arreglo incorrecto",
                            "Declaracion incorrecta de la matriz",
                            "Declaracion incorrecta de estatutos",
                            "Declaracion incorrecta del estatuto do",
                            "Declaracion incorrecta del estatuto for",
                            "Declaracion incorrecta del estatuto if",
                            "Declaracion incorrecta del elseif",
                            "Declaracion incorrecta del else",
                            "Declaracioin incorrecta del estatuto unit",
                            "Declaracion incorrecta del estatuto asig",
                            "Declaracion incorrecta del estatuto read",
                            "Declaracion incorrecta del estatuto write",
                            "Declaracion incorrecta del estatuto while",
                            "Declaracion incorrecta del estatuto return",
                            "Declaracion incorrecta de estatuto dimasig",
                            "Declaracion incorrecta de term",
                            "Declaracion Fact incorrecta",
                            "Declaracion de llamada a funcion incorrecta",
                            "Declaracion de expresion incorrecta",
                            "Se esperaba un operador relacional",
                            "Se esperaba un operador de asignacion",
```

```
                                "Declaracion de acceso a variable incorrecta"};

string columnas[72][72] ={{"1000","library"},
                          {"1001","@este se me fue"},
                          {"1002",";"},
                          {"1003","class"},
                          {"1004","id"},
                          {"1005","endclass"},
                          {"1006","def"},
                          {"1007",","},
                          {"1008","as"},
                          {"1009","["},
                          {"1010","]"},
                          {"1011",".."},
                          {"1012","cteentera"},
                          {"1013","int"},
                          {"1014","float"},
                          {"1015","char"},
                          {"1016","string"},
                          {"1017","bool"},
                          {"1018","void"},
                          {"1019","public"},
                          {"1020","private"},
                          {"1021","protected"},
                          {"1022","main"},
                          {"1023","("},
                          {"1024",")"},
                          {"1025","endmain"},
                          {"1026","func"},
                          {"1027","endfunc"},
                          {"1028","local"},
                          {"1029","endlocal"},
                          {"1030","read"},
                          {"1031","write"},
                          {"1032","return"},
                          {"1033","++"},
                          {"1034","--"},
                          {"1035","for"},
                          {"1036","="},
                          {"1037","to"},
                          {"1038","do"},
                          {"1039","endfor"},
                          {"1040","while"},
                          {"1041","endwhile"},
                          {"1042","eval"},
                          {"1043","enddo"},
                          {"1044","if"},
                          {"1045","elseif"},
                          {"1046","else"},
                          {"1047","endif"},
                          {"1048","||"},
                          {"1049","&&"},
                          {"1050","!"},
                          {"1051","=="},
                          {"1052","!="},
                          {"1053","<"},
                          {"1054","<="},
                          {"1055",">"},
                          {"1056",">="},
                          {"1057","+"},
```

```
                                {"1058","-"},
                                {"1059","*"},
                                {"1060","/"},
                                {"1061","%"},
                                {"1062","ctereal"},
                                {"1063","ctenotacion"},
                                {"1064","ctecaracter"},
                                {"1065","ctestring"},
                                {"1066","+="},
                                {"1067","-="},
                                {"1068","*="},
                                {"1069","/="},
                                {"1070","%="},
                                {"1071","$"}};

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_clicked()
{
    //Tipo de archivo a abrir
    QString filter = "Spes files (*.spes)";
    QString file_name = QFileDialog::getOpenFileName(this, "Abrir archivo",
QDir::homePath(), filter);

    QFile file(file_name);

    if (!file.open(QFile::ReadOnly | QFile::Text))
    {
        QMessageBox::warning(this, "Advertencia", "Archivo no abierto");
    }
    QTextStream in(&file);
    QString text = in.readAll();
    ui->plainTextEdit->setPlainText(text);
    file.close();
    ui->plainTextEdit_2->setPlainText("");
}

void MainWindow::on_pushButton_2_clicked()
{
    //Tipo de archivo a guardar
    QString filter = "Spes files (*.spes)";
    QString file_name = QFileDialog::getSaveFileName(this, "Guardar
Archivo", QDir::homePath(), filter);
    QFile file(file_name);

    if (!file.open(QFile::WriteOnly | QFile::Text))
    {
        QMessageBox::warning(this, "Advertencia", "Archivo no guardado");
    }
    QTextStream out(&file);
    QString text = ui->plainTextEdit->toPlainText();
    out << text;
    file.flush();
}

int relaciona(char c)
{
    //Caso numero
```

```
    if (c >= '0' && c <= '9'){
        return 2;
    }
    //Caso caracteres conocidos
    switch (c)
    {
        case '_': return 3;
        case '.': return 4;
        case 'E' : return 5;
        case 'e': return 6;
        case '+': return 7;
        case '-': return 8;
        case '*': return 9;
        case '/': return 10;
        case '#': return 11;
        case '%': return 12;
        case '&': return 13;
        case '!': return 14;
        case '<': return 15;
        case '>': return 16;
        case '=': return 17;
        case '|': return 18;
        case ',': return 19;
        case ';': return 20;
        case '[': return 21;
        case ']': return 22;
        case '(': return 23;
        case ')': return 24;
        case '"': return 28;
        case ' ': return 27;
        case '\n': return 26;
        case '\t': return 25;
        case '\0': return 31;
    }
    //Caso letras
    if ((int(c) >= 97 && int(c) <= 122 )|| (int(c) >= 65 && int(c) <= 90))
    {
        return 0;
    }
    //Caso comilla simple
    switch (int(c)) {
    case 39: return 29;
    }
    //Caso diferente
    return 30;
}

int Token(int e,string token)
{
    switch(e)
    {
        case 100: return  0;
        case 101: return  1;
        case 102: return  2;
        case 103: return  3;
        case 104: return  4;
        case 105: return  5;
        case 106: return  6;
        case 107: return  7;
        case 108: return  8;
        case 109: return  9;
```

```
        case 110: return  10;
        case 111: return  11;
        case 112: return  12;
        case 113: return  13;
        case 114: return  14;
        case 115: return  15;
        case 116: return  16;
        case 117: return  17;
        case 118: return  18;
        case 119: return  19;
        case 120: return  20;
        case 121: return  21;
        case 122: return  22;
        case 124: return  23;
        case 126: return  24;
        case 127: return  25;
        case 128: return  26;
        case 129: return  27;
        case 130: return  28;
        case 131: return  29;
        case 132: return  30;
        case 133: return  31;
        case 134: return  32;
        case 136: return  33;
        case 137: return  34;
        case 138: return  35;
        case 139: return  36;
        case 140: return  37;
        case 144: return  38;
    }
    return 100;
} //fin de token
int Error(int e) //Esta es la Cuadruplos de Errores
{
    switch (e)
    {
    case 500:  return 0;
    case 501:  return 1;
    case 502:  return 2;
    case 503:  return 3;
    case 504:  return 4;
    case 505:  return 5;
    case 506:  return 6;
    case 507:  return 7;
    case 508:  return 8;
    case 510:  return 9;
    case 511:  return 10;
    case 512:  return 11;
    case 513:  return 12;
    }
    return 100;
} // fin de error

void MainWindow::on_btnAnaliza_clicked()
{
    int edo, col;
    char car;
    int cont_cadena = 0;
    QString a = ui->plainTextEdit->toPlainText();
    string cadena =a.toUtf8().constData();
    char *cstr = &cadena[cont_cadena];
```

```
    string resultado = "";
    string cadenaResultante,identificador;
    QString r;
    int T=0;
    bool AnalisisCorrecto=true;
    if(a.length()>0){
        while ( cont_cadena <= (a.length()))
        {
            //Se reinician variables
            edo = 0;
            cadenaResultante="",identificador="";
            while (edo <= 26)
            {
                car = (char)cstr[cont_cadena];
                col = relaciona (car);
                edo = M[edo][col];
                //Terminaron con la letra correspondiente

if(edo==104||edo==105||edo==107||edo==109||edo==110||edo==112||edo==113||ed
o==115||edo==116||edo==120||edo==122||edo==123||edo==124||edo==126||edo==12
6||edo>127){
                    cadenaResultante=cadenaResultante+""+car;
                }//Terminaron con dif, deja el elemento que lo termino para
volver a analizarlo
                else
if(edo==100||edo==101||edo==102||edo==103||edo==106||edo==108||edo==111||ed
o==114||edo==117||edo==119||edo==121||edo==127){
                    cont_cadena--;;
                }
                if(edo<27){
                    //Omitimos a los delimitadores
                    if(edo!=0&&car!='\n'&&car!='\b'&&car!='\t'&&car!='
'&&car!='\0'){
                        cadenaResultante=cadenaResultante+""+car+"";
                    }
                    cont_cadena++;
                }

            }
            //concatenamos resultado de analisis
            resultado+=cadenaResultante;
            if (edo >= 100 && edo <= 144)
            {
                //Respaldamos el valor de nuestra cadena antes de asiganrle
el resultado de su token correspondiente
                identificador=cadenaResultante;
                T =Token(edo,identificador);
                cadenaResultante = tokenExitoso[T]; //Obtiene la cadena
correspondiente al estado al que llego
                //Analisis para detectar si el identificador pertenece a
las palabras reservadas
                if(edo==100){
                    for(int rPR=0;rPR<32;rPR++){

if(identificador.compare(palabrasResevadas[rPR])==0)
                        {
                            cadenaResultante="Palabra Reservada";
                            rPR=33;
                        }
                    }
```

```
                }
            }
            else
            {
                T =Error(edo);
                cadenaResultante = tokenError[T]; //Obtiene la cadena
correspondiente al estado al que llego
                AnalisisCorrecto=false;
                cont_cadena=a.size();
            }
            resultado+=" -> "+cadenaResultante+"\n";
            cont_cadena++;
            if(!AnalisisCorrecto){
                resultado+=" - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - \n"
                           "      Análisis Finalizado se encontraron \n"
                           "                          Errores\n"
                           " - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - ";
                //Ponemos resultados en la pantalla
                r =QString::fromStdString(resultado);
                ui->plainTextEdit_2->setPlainText(r);
                break;
            }
            else if(AnalisisCorrecto&&cont_cadena>=a.length()){
                resultado+=" - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - \n"
                           "      Análisis Finalizado Correctamente\n"
                           " - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - ";
                //Ponemos resultados en la pantalla
                r =QString::fromStdString(resultado);
                ui->plainTextEdit_2->setPlainText(r);
                break;
            }
            //Ponemos resultados en la pantalla
            r =QString::fromStdString(resultado);
            ui->plainTextEdit_2->setPlainText(r);
        }
    }else{
        resultado+=" - - - - - - - - - - - - - - - - - - - - - - - - - - - -
\n"
                   "           Sin datos para analizar\n"
                   " - - - - - - - - - - - - - - - - - - - - - - - - - - - -
";
        //Ponemos resultados en la pantalla
        r =QString::fromStdString(resultado);
        ui->plainTextEdit_2->setPlainText(r);
    }
}

void GetToken(){
    int edo, col;
    char car;
    string cadena =cadenaAnalizar.toUtf8().constData();
    char *cstr = &cadena[cont_cadena_posicion];
    string resultado = "";
    string cadenaResultante,identificador;
    QString r;
    int T=0;
    bool AnalisisCorrecto=true;
```

```
    if(cadenaAnalizar.length()>0){
        while ( cont_cadena_posicion <= (cadenaAnalizar.length()))
        {
            //Se reinician variables
            edo = 0;
            cadenaResultante="",identificador="";
            while (edo <= 26)
            {
                car = (char)cstr[cont_cadena_posicion];
                col = relaciona (car);
                edo = M[edo][col];
                //Terminaron con la letra correspondiente

if(edo==104||edo==105||edo==107||edo==109||edo==110||edo==112||edo==113||ed
o==115||edo==116||edo==120||edo==122||edo==123||edo==124||edo==126||edo==12
6||edo>127){
                    cadenaResultante=cadenaResultante+""+car;
                }//Terminaron con dif, deja el elemento que lo termino para
volver a analizarlo
                else
if(edo==100||edo==101||edo==102||edo==103||edo==106||edo==108||edo==111||ed
o==114||edo==117||edo==119||edo==121||edo==127){
                    cont_cadena_posicion--;;
                }
                if(edo<27){
                    //Omitimos a los delimitadores
                    if(edo!=0&&car!='\n'&&car!='\b'&&car!='\t'&&car!='
'&&car!='\0'){
                        cadenaResultante=cadenaResultante+""+car+"";
                    }
                    cont_cadena_posicion++;
                }

            }
            //concatenamos resultado de analisis
            resultado+=cadenaResultante;
            if (edo >= 100 && edo <= 144)
            {
                //Respaldamos el valor de nuestra cadena antes de asiganrle
el resultado de su token correspondiente
                identificador=cadenaResultante;
                Tokens.push_front(identificador);
                T =Token(edo,identificador);
                cadenaResultante = tokenExitoso[T]; //Obtiene la cadena
correspondiente al estado al que llego
                //Analisis para detectar si el identificador pertenece a
las palabras reservadas
                if(edo==100){
                    for(int rPR=0;rPR<32;rPR++){

if(identificador.compare(palabrasResevadas[rPR])==0)
                        {

                            cadenaResultante=palabrasResevadas[rPR];
                            rPR=33;
                        }
                    }

                }
                pilaResultadoTokens.push_front(edo);
```

```
pilaResultadoTokensTexto.push_front(cadenaResultante);
            }
            else
            {
                T =Error(edo);
                cadenaResultante = tokenError[T]; //Obtiene la cadena
correspondiente al estado al que llego
                AnalisisCorrecto=false;
                cont_cadena_posicion=cadenaAnalizar.size();
            }

            ResultadosTokens.push_front(edo);

            resultado+=" -> "+cadenaResultante+"\n";
            cont_cadena_posicion++;
            if(!AnalisisCorrecto){
                break;
            }
            else
if(AnalisisCorrecto&&cont_cadena_posicion>=cadenaAnalizar.length()){
                break;
            }
        }
    }
}
int RelacionaToken(string token){
    if(ResultadosTokens.top()<500){
        for(int x=0;x<72;x++){
            if(token.compare(columnas[x][1])==0)
            {
                int r=stoi(columnas[x][0]);
                return r;
            }
        }
        if(ResultadosTokens.top()==100){
            return 1004;
        }else if (ResultadosTokens.top()==101) {
            return 1012;
        }else if (ResultadosTokens.top()==102) {
            return 1062;
        }else if (ResultadosTokens.top()==103) {
            return 1063;
        }else if (ResultadosTokens.top()==104) {
            return 1064;
        }else if (ResultadosTokens.top()==105) {
            return 1065;
        }else if (ResultadosTokens.top()==134) {
            return 1080;
        }else if (ResultadosTokens.top()==136) {
            return 1080;
        }
    }else
    {
        return -1;
    }
}
void MainWindow::on_btnAnaliza_2_clicked()
{
    string resultado="";
    cont_cadena_posicion=0;
```

```
    cadenaAnalizar= ui->plainTextEdit->toPlainText();
    while(!Pila.empty()){
        Pila.pop();
    }
    while(!ResultadosTokens.empty()){
        ResultadosTokens.pop();
    }
    while(!Tokens.empty()){
        Tokens.pop();
    }
    Pila.push(999);
    Pila.push(0);

    GetToken();

    ResultadosTokens.push_front(0);
    Tokens.push_front("$");
    string token = Tokens.pop();
    resultado+=token+",\n";
    bool continua=true;
    while (continua) {
        qDebug()<<QString::fromStdString(token)<<"<-  token";
        qDebug()<<RelacionaToken(token)<<"<- Relaciona";
        qDebug()<<Pila.top()<<"<-  pila";
        if(RelacionaToken(token)==-1){
            ui->plainTextEdit_2->setPlainText(
QString::fromStdString(tokenError[Error(ResultadosTokens.top())]));
            continua=false;
            QMessageBox::about(this, "SPES", "Análisis detenido, error
léxico encontrado");
            break;
        }
        while(RelacionaToken(token)==1080){
            token=Tokens.pop();
            ResultadosTokens.pop();
        }

        if(Pila.top()>998){
            if(Pila.top()==999){
                qDebug()<<"Pila acabo";
                resultado+=" - - - - - - - - - - - - - - - - - - - - - - -
- - - - \n"
                           "      Análisis Finalizado Correctamente\n"
                           " - - - - - - - - - - - - - - - - - - - - - - -
- - - - ";
                QMessageBox::about(this, "SPES", "Análisis finalizado
correctamente");
                continua=false;
                break;
            }else{
                if(Pila.top()==RelacionaToken(token)&&Pila.top()!=999){
                    qDebug()<<"Saco de pila";
                    Pila.pop();
                    token=Tokens.pop();
                    resultado+=token+",\n";
                    ResultadosTokens.pop();
                }else{
                    qDebug()<<"Error "<<Pila.top();
                    if(Pila.top()>1499){
                        int x =Pila.top()-1500;
```

```
qDebug()<<QString::fromStdString(ErroresAnalisis[x])<<"    ";
                        resultado+=" - - - - - - - - - - - - - - - - - - - -
- - - - - - - - \n"
                           "Error de análisis :\n"+ErroresAnalisis[x]+",\n"
                           " - - - - - - - - - - - - - - - - - - - - - - -
- - - - - ";
                        QMessageBox::about(this, "SPES", "Análisis
detenido, error sintáctio encontrado");
                        continua=false;
                        break;
                    }else {
                        string a =columnas[Pila.top()-1000][1];
                        qDebug()<<QString::fromStdString(a)<<" <<<";
                        resultado+=" - - - - - - - - - - - - - - - - - - - -
- - - - - - - - \n"
                           "Error se esperaba -> "+a+"\n"
                           " - - - - - - - - - - - - - - - - - - - - - - -
- - - - - ";
                        QMessageBox::about(this, "SPES", "Análisis
detenido, error sintáctico encontrado");
                        continua=false;
                        break;
                    }
                    break;
                }
            }
        }
        else {

            if(MP[Pila.top()][RelacionaToken(token)-1000]<123){
                int Salio=Pila.top();
                Pila.pop();
                qDebug()<<" llenar pila "+MP[Salio][RelacionaToken(token)-
1000];
                for(int x = 11; x >= 0; x--){
                    if(producciones[MP[Salio][RelacionaToken(token)-
1000]][x] != 0){

Pila.push(producciones[MP[Salio][RelacionaToken(token)-1000]][x]);
                        qDebug()<<Pila.top();
                    }
                }
                if(Pila.top() == -1){
                    qDebug()<<"saco de pila, queda esto "<<Pila.top();
                    Pila.pop();
                }
                //imprimePila();
            }else {
                qDebug()<<"Error- - - -
"<<MP[Pila.top()][RelacionaToken(token)-1000];
                string a
=ErroresAnalisis[MP[Pila.top()][RelacionaToken(token)-1000]-1500];
                resultado+=" - - - - - - - - - - - - - - - - - - - - - - - -
- - - - \n"
                           "Error -> "+a+"\n"
                           " - - - - - - - - - - - - - - - - - - - - - - -
- - - - - ";
                QMessageBox::about(this, "SPES", "Análisis detenido, error
sintáctico encontrado");
```

```
                continua=false;
                break;
            }
        }
    }

    ui->plainTextEdit_2->setPlainText(QString::fromStdString(resultado));
}


void MainWindow::on_btnAnaliza_3_clicked()
{
    semantico=1;
    string resultado="";
    cuadroResultado="";
    cont_cadena_posicion=0;
    contCuad=1;
    contR=1;
    ban1=0;
    banwhile= 0;
    banfor=0;
    ui->Cuadruplos->clear();
    cadenaAnalizar= ui->plainTextEdit->toPlainText();
    while(!Pila.empty()){
        Pila.pop();
    }
    while(!Variables.empty()){
        Variables.pop();
    }
    while(!operadores.empty()){
        operadores.pop();
    }
    while(!operandos.empty()){
        operandos.pop();
    }
    while(!ResultadosTokens.empty()){
        ResultadosTokens.pop();
    }
    while(!Tokens.empty()){
        Tokens.pop();
    }
    while(!pilaTokens.empty()){
        pilaTokens.pop();
    }
    while(!pilaResultadoTokens.empty()){
        pilaResultadoTokens.pop();
    }
    while(!pilaResultadoTokensTexto.empty()){
        pilaResultadoTokensTexto.pop();
    }
    Pila.push(999);
    Pila.push(0);

    GetToken();

    pilaTokens=Tokens;


    ResultadosTokens.push_front(0);
    Tokens.push_front("$");
    string token = Tokens.pop();
```

```
    resultado+=token+",\n";
    bool continua=true;
    bool SintaxisCorrecta=true;
    while (continua) {
        qDebug()<<QString::fromStdString(token)<<"<-  token";
        qDebug()<<RelacionaToken(token)<<"<- Relaciona";
        qDebug()<<Pila.top()<<"<-  pila";
        if(RelacionaToken(token)==-1){
            ui->plainTextEdit_2->setPlainText(
QString::fromStdString(tokenError[Error(ResultadosTokens.top())]));
            continua=false;
            QMessageBox::about(this, "SPES", "Análisis detenido, error
léxico encontrado");
            break;
        }
        while(RelacionaToken(token)==1080){
            token=Tokens.pop();
            ResultadosTokens.pop();
        }

        if(Pila.top()>998){
            if(Pila.top()==999){
                qDebug()<<"Pila acabo";
                resultado+=" - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - \n"
                           "      Análisis Finalizado Correctamente\n"
                           " - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - ";

                continua=false;
                Semantico();
                break;
            }else{
                if(Pila.top()==RelacionaToken(token)&&Pila.top()!=999){
                    qDebug()<<"Saco de pila";
                    Pila.pop();
                    token=Tokens.pop();
                    resultado+=token+",\n";
                    ResultadosTokens.pop();
                }else{
                    qDebug()<<"Error "<<Pila.top();
                    if(Pila.top()>1499){
                        int x =Pila.top()-1500;

qDebug()<<QString::fromStdString(ErroresAnalisis[x])<<"   ";

                        resultado+=" - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - \n"
                           "Error de análisis :\n"+ErroresAnalisis[x]+",\n"
                           " - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - ";
                        QMessageBox::about(this, "SPES", "Análisis
detenido, error sintáctio encontrado");
                        continua=false;
                        break;
                    }else {
                        string a =columnas[Pila.top()-1000][1];
                        qDebug()<<QString::fromStdString(a)<<" <<<";
                        resultado+=" - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - \n"
                           "Error se esperaba -> "+a+"\n"
```

```
                          " - - - - - - - - - - - - - - - - - - - - - - -
- - - - ";
                       QMessageBox::about(this, "SPES", "Análisis
detenido, error sintáctico encontrado");
                       continua=false;
                       break;
                   }
                   break;
               }
           }
       }else{

           if(MP[Pila.top()][RelacionaToken(token)-1000]<123){
               int Salio=Pila.top();
               Pila.pop();
               qDebug()<<" llenar pila "+MP[Salio][RelacionaToken(token)-
1000];
               for(int x = 11; x >= 0; x--){
                   if(producciones[MP[Salio][RelacionaToken(token)-
1000]][x] != 0){

Pila.push(producciones[MP[Salio][RelacionaToken(token)-1000]][x]);
                       qDebug()<<Pila.top();
                   }
               }
               if(Pila.top() == -1){
                   qDebug()<<"saco de pila, queda esto "<<Pila.top();
                   Pila.pop();
               }
               //imprimePila();
           }else {
               qDebug()<<"Error- - - -
"<<MP[Pila.top()][RelacionaToken(token)-1000];
               string a
=ErroresAnalisis[MP[Pila.top()][RelacionaToken(token)-1000]-1500];
               resultado+=" - - - - - - - - - - - - - - - - - - - - - - - -
- - - - \n"
                          "Error -> "+a+"\n"
                          " - - - - - - - - - - - - - - - - - - - - - - -
- - - - ";
               QMessageBox::about(this, "SPES", "Análisis detenido, error
sintáctico encontrado");
               continua=false;
               SintaxisCorrecta=false;
               break;
           }
       }
   }
   if(SintaxisCorrecta){
       if(semantico){
           resultado=" - - - - - - - - - - - - - - - - - - - - - - - - - -
-  - - - - - - - - - - - - - - - - -\n"
                  "      Generación de Codigo Intermedio Finalizada
Correctamente\n"
                  " - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - -";
       }else {
           resultado=" - - - - - - - - - - - - - - - - - - - - - - - - - -
-  - - - - - - - - - - - - - - - - - - - - -  \n"
                  "      Generación de Codigo Intermedio Finalizada Se
encontraron Errores\n"
```

```
                    " - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - -  ";
        }
        cuadroResultado+=QString::fromStdString(resultado);
        ui->plainTextEdit_2->setPlainText(cuadroResultado);
    }else {
        ui->plainTextEdit_2-
>setPlainText(QString::fromStdString(resultado));
    }
}

QString CuadruplosOperaciones(QString Op1, QString Op2,QString Oper){
string e;
string n="hola";
n;
n=std::to_string(contCuad);
    if(typeOp1=="Identificador"){
        Op1=existeVariable(Op1.toStdString());
    }else if (typeOp1 == "Constantes numérica entera"){
        Op1="E";
    }else if (typeOp1=="Constantes numérica real") {
        Op1="R";
    }else if (typeOp1=="Constantes numérica de notación científica") {
        Op1="N";
    }else if (typeOp1=="Constante carácter") {
        Op1="C";
    }else if (typeOp1=="Constante string") {
        Op1="S";
    }else if(typeOp1=="resultado"){
        Op1 = existeVariable(Op1.toStdString());
    }

    if(typeOp2=="Identificador"){
        Op2 = existeVariable(Op2.toStdString());
    }else if (typeOp2 == "Constantes numérica entera"){
        Op2="E";
    }else if (typeOp2=="Constantes numérica real") {
        Op2="R";
    }else if (typeOp2=="Constantes numérica de notación científica") {
        Op2="N";
    }else if (typeOp2=="Constante carácter") {
        Op2="C";
    }else if (typeOp2=="Constante string") {
        Op2="S";
    }else if(typeOp2=="resultado"){
         Op2 = existeVariable(Op2.toStdString());
    }


    if (Op1 == "E" && Op2 == "E") {
        if (Oper == "*") {
            return "E";
        } else if (Oper == "+") {
            return "E";
        } else if (Oper == "-") {
            return "E";
        } else if (Oper == "/") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "R";
        } else if (Oper == "%") {
```

```
            return "R";
        } else if (Oper == "||") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "B";
        } else if (Oper == "&&") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "B";
        } else if (Oper == "!") {
            return "B";
        } else if (Oper == "==") {
            return "B";
        } else if (Oper == "!=") {
            return "B";
        } else if (Oper == "<") {
            return "B";
        } else if (Oper == "<=") {
            return "B";
        } else if (Oper == ">") {
            return "B";
        } else if (Oper == ">=") {
            return "B";
        } else if (Oper == "**") {
            return "E";
        } else if (Oper == "=") {
            return "E";
        } else if (Oper == "+=") {
            return "E";
        } else if (Oper == "-=") {
            return "E";
        } else if (Oper == "*=") {
            return "E";
        } else if (Oper == "/=") {
            return "R";
        } else if (Oper == "%=") {
            return "R";
        }
    } else if ((Op1 == "E" && Op2 == "R")) {
        if (Oper == "*") {
            return "R";
        } else if (Oper == "+") {
            return "R";
        } else if (Oper == "-") {
            return "R";
        } else if (Oper == "/") {
            return "R";
        } else if (Oper == "%") {
            return "R";
        } else if (Oper == "||") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "B";
        } else if (Oper == "&&") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "B";
        } else if (Oper == "!") {
            return "B";
        } else if (Oper == "==") {
            return "B";
```

```
        } else if (Oper == "!=") {
            return "B";
        } else if (Oper == "<") {
            return "B";
        } else if (Oper == "<=") {
            return "B";
        } else if (Oper == ">") {
            return "B";
        } else if (Oper == ">=") {
            return "B";
        } else if (Oper == "**") {
            return "R";
        } else if (Oper == "=") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "E";
        } else if (Oper == "+=") {
            return "R";
        } else if (Oper == "-=") {
            return "R";
        } else if (Oper == "*=") {
            return "R";
        } else if (Oper == "/=") {
            return "R";
        } else if (Oper == "%=") {
            return "R";
        }
    } else if (Op1 == "R" && Op2 == "E") {
        if (Oper == "*") {
            return "R";
        } else if (Oper == "+") {
            return "R";
        } else if (Oper == "-") {
            return "R";
        } else if (Oper == "/") {
            return "R";
        } else if (Oper == "%") {
            return "R";
        } else if (Oper == "||") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "B";
        } else if (Oper == "&&") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "B";
        } else if (Oper == "!") {
            return "B";
        } else if (Oper == "==") {
            return "B";
        } else if (Oper == "!=") {
            return "B";
        } else if (Oper == "<") {
            return "B";
        } else if (Oper == "<=") {
            return "B";
        } else if (Oper == ">") {
            return "B";
        } else if (Oper == ">=") {
            return "B";
        } else if (Oper == "**") {
```

```
            return "R";
        } else if (Oper == "=") {
            return "R";
        } else if (Oper == "+=") {
            return "R";
        } else if (Oper == "-=") {
            return "R";
        } else if (Oper == "*=") {
            return "R";
        } else if (Oper == "/=") {
            return "R";
        } else if (Oper == "%=") {
            return "R";
        }
    } else if ((Op1 == "R" && Op2 == "R")) {
        if (Oper == "*") {
            return "R";
        } else if (Oper == "+") {
            return "R";
        } else if (Oper == "-") {
            return "R";
        } else if (Oper == "/") {
            return "R";
        } else if (Oper == "%") {
            return "R";
        } else if (Oper == "||") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "B";
        } else if (Oper == "&&") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "B";
        } else if (Oper == "!") {
            return "B";
        } else if (Oper == "==") {
            return "B";
        } else if (Oper == "!=") {
            return "B";
        } else if (Oper == "<") {
            return "B";
        } else if (Oper == "<=") {
            return "B";
        } else if (Oper == ">") {
            return "B";
        } else if (Oper == ">=") {
            return "B";
        } else if (Oper == "**") {
            return "R";
        } else if (Oper == "=") {
            return "R";
        } else if (Oper == "+=") {
            return "R";
        } else if (Oper == "-=") {
            return "R";
        } else if (Oper == "*=") {
            return "R";
        } else if (Oper == "/=") {
            return "R";
        } else if (Oper == "%=") {
            return "R";
```

```
        }
    } else if ((Op1 == "C" && Op2 == "C") || (Op1 == "C" && Op2 == "S")) {
        if (Oper == "==") {
            return "B";
        } else if (Oper == "!=") {
            return "B";
        } else if (Oper == "<") {
            return "B";
        } else if (Oper == "<=") {
            return "B";
        } else if (Oper == ">") {
            return "B";
        } else if (Oper == ">=") {
            return "B";
        } else {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "C";
        }
    }else if((Op1 == "S" && Op2 == "C") || (Op1 == "S" && Op2 == "S")||
(Op1 == "S" && Op2 == "E")){
        if (Oper != "=") {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
        }
        return "S";
    } else if (Op1 == "B" && Op2 == "B") {
        if (Oper == "||") {
            return "B";
        } else if (Oper == "&&") {
            return "B";
        } else if (Oper == "!") {
            return "B";
        } else if (Oper == "==") {
            return "B";
        } else if (Oper == "!=") {
            return "B";
        } else if (Oper == "<") {
            return "B";
        } else if (Oper == "<=") {
            return "B";
        } else if (Oper == ">") {
            return "B";
        } else if (Oper == ">=") {
            return "B";
        } else {
            e = "Error entre tipos en el cuadruplo "+n+" \n";
            cuadroResultado+=QString::fromStdString(e);
            return "B";
        }
    } else {
        e = "Error entre tipos en el cuadruplo "+n+" \n";
        cuadroResultado+=QString::fromStdString(e);
        return "E";
    }
}

void MainWindow::Semantico(){
    cuadroResultado="";
    string token="",tokenTexto;
    int tipoToken = 0;
```

```
    bool finDeclaracion=false;
    while(!pilaTokens.empty()){
        token=pilaTokens.pop();
        tokenTexto=pilaResultadoTokensTexto.pop();
        tipoToken=pilaResultadoTokens.pop();
        if(!finDeclaracion){
            //Llena Cuadruplos de tipos
            if(token.compare("def")==0){
                token=pilaTokens.pop();
                tokenTexto=pilaResultadoTokensTexto.pop();
                tipoToken=pilaResultadoTokens.pop();

                while(token.compare("as")){
                    bool disponible=true;
                    if(token.compare(",")){
                        for(int v=0;v<Variables.count();v++){
                            if(!token.compare(Variables[v])){
                                disponible=false;
                                string e=" Error la variable "+token+" ya
fue definida\n";
                                cuadroResultado+=QString::fromStdString(e);
                                semantico=0;
                            }
                        }
                        if(disponible){
                            Variables.push(token);
                            VariablesTipo.push("x");
                        }
                    }
                    token=pilaTokens.pop();
                    tokenTexto=pilaResultadoTokensTexto.pop();
                    tipoToken=pilaResultadoTokens.pop();
                }
                token=pilaTokens.pop();
                tokenTexto=pilaResultadoTokensTexto.pop();
                tipoToken=pilaResultadoTokens.pop();

                static QStack<int> pilaVariablesTipoTemp;
                static QStack<string>
VariablesTemp,pilaVariablesTipoTextoTemp;



while(!VariablesTipo.empty()&&VariablesTipo.top().compare("x")==0){
                    if(!token.compare("int")){
                        token="E";
                    }
                    if(!token.compare("float")){
                        token="R";
                    }
                    if(!token.compare("char")){
                        token="C";
                    }
                    if(!token.compare("string")){
                        token="S";
                    }
                    if(!token.compare("bool")){
                        token="B";
                    }
                    VariablesTemp.push(Variables.pop());
                    VariablesTipo.pop();
```

```
                    pilaVariablesTipoTextoTemp.push(token);
                }
                while(!VariablesTemp.empty()){
                    Variables.push(VariablesTemp.pop());
                    VariablesTipo.push(pilaVariablesTipoTextoTemp.pop());
                }

            }
            if(!(token.compare("main"))){
                finDeclaracion=true;
            }
        }
        //Operaciones dentro del main
        if(finDeclaracion){
            //Estatuto asignacion
            if(!(tokenTexto.compare(tokenExitoso[0]))){
                qDebug()<<"Hola";

estatutoAsig(QString::fromStdString(token),QString::fromStdString(tokenTexto));

            }
            if(!(tokenTexto.compare("if"))){
                qDebug()<<"Hola if";
                operadores.push("/MFF");
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
                operandos.push(QString::fromStdString(pilaTokens.pop()));

operandosTipos.push(QString::fromStdString(pilaResultadoTokensTexto.pop()));
                operadores.push(QString::fromStdString(pilaTokens.pop()));
                pilaResultadoTokensTexto.pop();
                operandos.push(QString::fromStdString(pilaTokens.pop()));

operandosTipos.push(QString::fromStdString(pilaResultadoTokensTexto.pop()));
                generarCuadruplo();
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
                cuadrupoSaltoFalso();
            }
            if(!(tokenTexto.compare("else"))){
                qDebug()<<"Hola else";
                //llena el salto en falso pendiente del if y genera el
salto incondicional hasta el elseif
                ban1=1;
                    llenarSaltoTope();//llena el ultimo salto pendiente
                    cuadruploSaltoIncondicional();
            }
            if(!(tokenTexto.compare("endif"))){
                qDebug()<<"Hola endif";
                llenarSaltoTope();// llena el salto en el tope de la fila
de saltos con el contador de l semantico
            }
            if(!(tokenTexto.compare("while"))){
                operadores.push("/MFF");
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
                operandos.push(QString::fromStdString(pilaTokens.pop()));
```

```
operandosTipos.push(QString::fromStdString(pilaResultadoTokensTexto.pop()))
;
                operadores.push(QString::fromStdString(pilaTokens.pop()));
                pilaResultadoTokensTexto.pop();
                operandos.push(QString::fromStdString(pilaTokens.pop()));

operandosTipos.push(QString::fromStdString(pilaResultadoTokensTexto.pop()))
;
                generarCuadruplo();
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
                saltos.push(contCuad-1);
                cuadrupoSaltoFalso();
                operadores.pop();
            }
            if(!(tokenTexto.compare("endwhile"))){
                qDebug()<<"Hola endwhile";

                ban1=1;
                llenarSaltoTope();
                cuadruploSaltoIncondicional();
                banwhile=1;
                llenarSaltoTope();

            }
            if(!(tokenTexto.compare("for"))){
                qDebug()<<"Hola for";

                Variables.push(pilaTokens.top());
                VariablesTipo.push("E");


operandosTipos.push(QString::fromStdString(pilaResultadoTokensTexto.pop()))
;
                operandos.push(QString::fromStdString(pilaTokens.pop()));
                pilaResultadoTokensTexto.pop();
                operadores.push(QString::fromStdString(pilaTokens.pop()));

operandosTipos.push(QString::fromStdString(pilaResultadoTokensTexto.pop()))
;
                operandos.push(QString::fromStdString(pilaTokens.pop()));

                QString Op1,Asig,Ope,Res,Op2;
                typeOp2= operandosTipos.pop();
                typeOp1= operandosTipos.pop();
                Res= operandos.pop();
                Op1= operandos.pop();
                Ope= operadores.pop();
                Asig= CuadruplosOperaciones(Op1,Res,Ope);

                ui->Cuadruplos->insertRow(ui->Cuadruplos->rowCount());
                QString num=QString::number(contCuad);
                ui->Cuadruplos->setItem(contCuad-1,0,new
QTableWidgetItem(num)); //num
                ui->Cuadruplos->setItem(contCuad-1,1,new
QTableWidgetItem(Ope)); //oper
                ui->Cuadruplos->setItem(contCuad-1,2,new
QTableWidgetItem(Op1)); //op1
                ui->Cuadruplos->setItem(contCuad-1,3,new QTableWidgetItem("
- ")); //op2
```

```
                ui->Cuadruplos->setItem(contCuad-1,4,new
QTableWidgetItem(Res)); //res

                qDebug()<<num<<"     "<<Ope<<"     "<<Op1<<"     "<<"-"<<"
"<<Res<<"     ";
                contCuad++;

                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();

                operandos.push(QString::fromStdString(pilaTokens.pop()));

operandosTipos.push(QString::fromStdString(pilaResultadoTokensTexto.pop()))
;

                Op2= operandos.pop();
                Asig= CuadruplosOperaciones(Op1,Op2,Ope);

                num=QString::number(contCuad);
                ui->Cuadruplos->setItem(contCuad-1,0,new
QTableWidgetItem(num)); //num
                ui->Cuadruplos->setItem(contCuad-1,1,new
QTableWidgetItem(">")); //oper
                ui->Cuadruplos->setItem(contCuad-1,2,new
QTableWidgetItem(Op1)); //op1
                ui->Cuadruplos->setItem(contCuad-1,3,new
QTableWidgetItem(Op2)); //op2
                QString r=QString("R%1").arg(contR);
                ui->Cuadruplos->setItem(contCuad-1,4,new
QTableWidgetItem(r)); //res
                qDebug()<<num<<"     "<<">"<<"     "<<Op1<<"     "<<Op2<<"
"<<r<<"     ";
                contR++;

                Variables.push(r.toStdString());
                VariablesTipo.push(Asig.toStdString());

                operandos.push(r);
                operandosTipos.push("B");
                saltos.push(contCuad-1);
                saltos.push(contCuad);
                                contCuad++;
                cuadrupoSaltoVerdadero();
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
            }
            if(!(tokenTexto.compare("endfor"))){
                qDebug()<<"Hola endfor";
                qDebug() << "Salto for";
                //
                ban1=1;
                qDebug() << "antes del sv/sf";

                llenarSaltoTope();
               qDebug() << "antes de salto Incondicional";

                cuadruploSaltoIncondicional();
               banfor=1;

              //
                llenarSaltoTope();
```

```
            }
            if(!(tokenTexto.compare("do"))){
                qDebug()<<"Hola do";
                saltos.push(contCuad);
            }
            if(!(tokenTexto.compare("eval"))){
                qDebug()<<"Hola eval";
                bando=1;
                operadores.push("/MFF");
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
                operandos.push(QString::fromStdString(pilaTokens.pop()));

operandosTipos.push(QString::fromStdString(pilaResultadoTokensTexto.pop()))
;
                operadores.push(QString::fromStdString(pilaTokens.pop()));
                pilaResultadoTokensTexto.pop();
                operandos.push(QString::fromStdString(pilaTokens.pop()));

operandosTipos.push(QString::fromStdString(pilaResultadoTokensTexto.pop()))
;
                generarCuadruplo();
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
                cuadrupoSaltoFalso();
                operadores.pop();
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
                llenarSaltoTope();
            }

            if(!(tokenTexto.compare("read"))){
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
                if(pilaResultadoTokensTexto.top()=="Identificador"){
                    existeVariable(pilaTokens.top());
                }else {
                    string e=" Error el estatuto read solo admite
identificadores: "+pilaTokens.top()+" no es valido\n";
                    cuadroResultado+=QString::fromStdString(e);
                }
                pilaResultadoTokensTexto.pop();

                QString leer=QString::fromStdString(pilaTokens.pop());

                ui->Cuadruplos->insertRow(ui->Cuadruplos->rowCount());
                QString num=QString::number(contCuad);
                ui->Cuadruplos->setItem(contCuad-1,0,new
QTableWidgetItem(num)); //num
                ui->Cuadruplos->setItem(contCuad-1,1,new
QTableWidgetItem("read")); //oper
                ui->Cuadruplos->setItem(contCuad-1,2,new QTableWidgetItem("
-  ")); //op1
                ui->Cuadruplos->setItem(contCuad-1,3,new QTableWidgetItem("
-  ")); //op2
                ui->Cuadruplos->setItem(contCuad-1,4,new
QTableWidgetItem(leer)); //res

                qDebug()<<num<<"     "<<"read"<<"     "<<" - "<<"     "<<"
- "<<"     "<<leer<<"     ";
                contCuad++;
```

```
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
            }
            if(!(tokenTexto.compare("write"))){
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();

                QString Op1,Asig,Ope,Res,Op2;
                typeOp2= "Constante string";
                typeOp1 =
QString::fromStdString(pilaResultadoTokensTexto.pop());
                Op1= QString::fromStdString(pilaTokens.pop());
                Op2= "";
                Ope= "=";

                Asig= CuadruplosOperaciones(Op1,Op2,Ope);

                QString escribir=Op1;
                ui->Cuadruplos->insertRow(ui->Cuadruplos->rowCount());
                QString num=QString::number(contCuad);
                ui->Cuadruplos->setItem(contCuad-1,0,new
QTableWidgetItem(num)); //num
                ui->Cuadruplos->setItem(contCuad-1,1,new
QTableWidgetItem("write")); //oper
                ui->Cuadruplos->setItem(contCuad-1,2,new QTableWidgetItem("
-  ")); //op1
                ui->Cuadruplos->setItem(contCuad-1,3,new QTableWidgetItem("
-  ")); //op2
                ui->Cuadruplos->setItem(contCuad-1,4,new
QTableWidgetItem(escribir)); //res

                qDebug()<<num<<"     "<<"write"<<"     "<<" - "<<"     "<<"
- "<<"     "<<escribir<<"     ";
                contCuad++;

                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
            }
        }
    }

}
//Cuadruplo asig
void MainWindow::estatutoAsig(QString Token, QString TokenTexto){
    QString token=Token,tokenTexto=TokenTexto;
    int tipoToken = 0;
    operandos.push(token);
    operandosTipos.push(tokenTexto);
    token = QString::fromStdString(pilaTokens.pop());
    tokenTexto = QString::fromStdString(pilaResultadoTokensTexto.pop());
    operadores.push(token);

    typeOp1=QString::fromStdString(pilaResultadoTokensTexto.top());
    while (pilaTokens.top()!=";") {
        string temp = pilaTokens.top();
        /*a*/     if
(pilaTokens.top()=="*"||pilaTokens.top()=="/"||pilaTokens.top()=="+"||pilaT
okens.top()=="-") {
            operadores.push(QString::fromStdString(pilaTokens.pop()));
            pilaResultadoTokensTexto.pop();
```

```
            /*b*/     }else if
(pilaTokens.top()!="("&&pilaTokens.top()!=")") {
            operandos.push(QString::fromStdString(pilaTokens.pop()));

operandosTipos.push(QString::fromStdString(pilaResultadoTokensTexto.pop()))
;
            temp = pilaTokens.top();
            if(pilaTokens.top()=="*"||pilaTokens.top()=="/"){
                if (operadores.top()=="*"||operadores.top()=="/") {
                    generarCuadruplo();
                    if ((operadores.top()=="+"||operadores.top()=="-
")&&pilaTokens.top()!="*"&&pilaTokens.top()!="/") {
                        generarCuadruplo();
                    }
                }else{
                    // No hace nada
                }
            }else{
                if (operadores.top()!="/MFF"&&operadores.top()!="=") {
                    generarCuadruplo();
                    if(operadores.top()=="+"||operadores.top()=="-"){
                        generarCuadruplo();
                    }
                }else {
                    //nada
                }
            }
            /*c*/   }else{
            if(pilaTokens.top()=="("){
                operadores.push("/MFF");
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
            }else {
                operadores.pop();
                pilaTokens.pop();
                pilaResultadoTokensTexto.pop();
                if
(operadores.top()=="*"||operadores.top()=="/"||operadores.top()=="+"||opera
dores.top()=="-") {
                    generarCuadruplo();
                }
            }
        }
    }
    pilaTokens.pop();
    pilaResultadoTokensTexto.pop();
    QString Op1,Asig,Ope,Res;
    typeOp2= operandosTipos.pop();
    typeOp1= operandosTipos.pop();
    Res= operandos.pop();
    Op1= operandos.pop();
    Ope= operadores.pop();
    Asig= CuadruplosOperaciones(Op1,Res,Ope);

    ui->Cuadruplos->insertRow(ui->Cuadruplos->rowCount());
    QString num=QString::number(contCuad);
    ui->Cuadruplos->setItem(contCuad-1,0,new QTableWidgetItem(num)); //num
    ui->Cuadruplos->setItem(contCuad-1,1,new QTableWidgetItem(Ope)); //oper
    ui->Cuadruplos->setItem(contCuad-1,2,new QTableWidgetItem(Op1)); //op1
    ui->Cuadruplos->setItem(contCuad-1,3,new QTableWidgetItem(" - "));
//op2
```

```
    ui->Cuadruplos->setItem(contCuad-1,4,new QTableWidgetItem(Res)); //res

    qDebug()<<num<<"     "<<Ope<<"     "<<Op1<<"     "<<"-"<<"
"<<Res<<"     ";
    contCuad++; //agregar a la pila el resultado
}
void MainWindow::generarCuadruplo()
{
    QString Op1,Op2,Ope,Res;
    typeOp2= operandosTipos.pop();
    typeOp1= operandosTipos.pop();
    Op2= operandos.pop();
    Op1= operandos.pop();
    Ope= operadores.pop();
    Res= CuadruplosOperaciones(Op1,Op2,Ope);



    ui->Cuadruplos->insertRow(ui->Cuadruplos->rowCount());
    QString num=QString::number(contCuad);
    ui->Cuadruplos->setItem(contCuad-1,0,new QTableWidgetItem(num)); //num
    ui->Cuadruplos->setItem(contCuad-1,1,new QTableWidgetItem(Ope)); //oper
    ui->Cuadruplos->setItem(contCuad-1,2,new QTableWidgetItem(Op1)); //op1
    ui->Cuadruplos->setItem(contCuad-1,3,new QTableWidgetItem(Op2)); //op2
    QString r=QString("R%1").arg(contR);
    ui->Cuadruplos->setItem(contCuad-1,4,new QTableWidgetItem(r)); //res

    qDebug()<<num<<"     "<<Ope<<"     "<<Op2<<"     "<<Op1<<"     "<<r<<"
";
    operandos.push(r);
    operandosTipos.push("resultado");
    Variables.push(r.toStdString());
    VariablesTipo.push(Res.toStdString());
    contCuad++; //agregar a la pila el resultado
    contR++;
}

void MainWindow::cuadrupoSaltoFalso()
{
    QString op = operandos.pop();// saca operando para el sf
    operandosTipos.pop();
    ui->Cuadruplos->insertRow(ui->Cuadruplos->rowCount());
    QString num=QString::number(contCuad);
    //crea en la Cuadruplos el cuadruplo del salto en falso
    ui->Cuadruplos->setItem(contCuad-1,0,new QTableWidgetItem(num)); //num
    ui->Cuadruplos->setItem(contCuad-1,1,new QTableWidgetItem("sf"));
//oper
    ui->Cuadruplos->setItem(contCuad-1,3,new QTableWidgetItem(" - "));
    ui->Cuadruplos->setItem(contCuad-1,2,new QTableWidgetItem(op)); //op1
    ui->Cuadruplos->setItem(contCuad-1,4,new QTableWidgetItem("")); //res

    contCuad++; //agregar a la pila el resultado

    qDebug() << "aqui hizo el salto";
    saltos.push(contCuad-1); // ignorar

}
void  MainWindow::cuadrupoSaltoVerdadero()
{
    QString op = operandos.pop();
```

```
    ui->Cuadruplos->insertRow(ui->Cuadruplos->rowCount());
    QString num=QString::number(contCuad);

    ui->Cuadruplos->setItem(contCuad-1,0,new QTableWidgetItem(num)); //num
    ui->Cuadruplos->setItem(contCuad-1,1,new QTableWidgetItem("sv"));
//oper
    ui->Cuadruplos->setItem(contCuad-1,3,new QTableWidgetItem(" - "));
    ui->Cuadruplos->setItem(contCuad-1,2,new QTableWidgetItem(op)); //op1
    ui->Cuadruplos->setItem(contCuad-1,4,new QTableWidgetItem("")); //res

    contCuad++; //agregar a la pila el resultado

    saltos.push(contCuad-1);
}
void  MainWindow::cuadruploSaltoIncondicional()
{

    ui->Cuadruplos->insertRow(ui->Cuadruplos->rowCount());
    QString num=QString::number(contCuad);

    ui->Cuadruplos->setItem(contCuad-1,0,new QTableWidgetItem(num)); //num
    ui->Cuadruplos->setItem(contCuad-1,1,new QTableWidgetItem("si"));
//oper
    ui->Cuadruplos->setItem(contCuad-1,2,new QTableWidgetItem(" - "));
//op1
    ui->Cuadruplos->setItem(contCuad-1,3,new QTableWidgetItem(" - "));
//op1
    ui->Cuadruplos->setItem(contCuad-1,4,new QTableWidgetItem("")); //res

    contCuad++; //agregar a la pila el resultado
    qDebug() << "si lo metio";
    saltos.push(contCuad-1);
}

void  MainWindow::llenarSaltoTope()
{
    //imprimeSem();
    int aux=saltos.pop();
    qDebug() << "Tope de saltos"<< aux;
    QString num;

    if(banfor)
    {
        qDebug() << "Llenando SI for";
        //imprimeSem();
        int numa=(contCuad);
        qDebug ()<< "pila saltos" << saltos.size();
        int aux2=saltos.pop();
        qDebug() << saltos.size();


        qDebug() << "este es el cuadruplo a llenar";
        qDebug() << "auxiliar2"<< aux2 << "{"<<aux << "},{"<<4<<"}";

        ui->Cuadruplos->setItem(aux-1,4,new
QTableWidgetItem(QString::number(aux2)));
        banfor=0;
    }else if(ban1){
            num=QString::number(contCuad+1);
            ui->Cuadruplos->setItem(aux-1,4,new QTableWidgetItem(num));
            ban1=0;
```

```
        } else if(!banwhile&&!bando){
            num=QString::number(contCuad);
            ui->Cuadruplos->setItem(aux-1,4,new QTableWidgetItem(num));
        }
    if(banwhile)
    {
        qDebug() << "Llenando SI while";
        int numa=(contCuad);
        int aux2=saltos.pop();

        qDebug() << "auxiliar2"<< aux2 << "{"<<contCuad << "},{"<<4<<"}";

        ui->Cuadruplos->setItem(contCuad-2,4,new
QTableWidgetItem(QString::number(aux2)));
        banwhile=0;
    }

    if(bando)
    {
        qDebug() << "Llenando SF do-enddo";
        int aux2=saltos.pop();

        ui->Cuadruplos->setItem(aux-1,4,new
QTableWidgetItem(QString::number(aux2)));
        bando=0;
    }
}

QString existeVariable(string variable){
    bool existe = false;
    for(int x=0;x<Variables.count();x++){
        if(!variable.compare(Variables[x])){
            existe= true;
            return QString::fromStdString(VariablesTipo[x]);
        }
    }
    if (!existe) {
        string e=" Error la variable "+variable+" no fue definida\n";
        cuadroResultado+=QString::fromStdString(e);
        semantico=0;
        Variables.push(variable);
        VariablesTipo.push("E");
    }
    return "E";
}
```