



Universidad Tecnológica Centroamericana

Compiladores I

Manual de Usuario de

Ing. Carlos Vallejo

Andre Velasquez | 11611355

Carlos Rivera | 11541261

29 de septiembre de 2020

Lenguaje:

Tipo de Variables

- **int:** tipos de enteros, positivos o negativos.
- **bool:** tipos booleanos este puede ser Verdadero o Falso.
- **char:** Tipo carácter, estos tipos están formados por una sola letra
- **arr[]:** Tipo arreglo, es un arreglo de una dimensión en el cual en ella se pueden definir una lista de palabras, números o caracteres.

Operadores Aritméticos

- **Suma:** Estos están representados por el carácter +
- **Resta:** Estos están representados por el carácter -
- **División:** Estos están representados por el carácter /
- **Multipliación:** Estos están representados por el carácter *
- **Residuo:** Estos están representados por el carácter %

Operadores Lógicos

- **Operador And:** Éstos están representados por &&
- **Operador Or:** Estos están representados por ||

Operadores Relacionales

- **Símbolo de Asignación:** Está representado por ~
 - Declaracion de variables `Id(type) ~ value`
 - `number(int)~10;`
 - `letter(char) ~ a;`
 - `ttrue(bool)~true;`

- **Símbolo de Igualdad:** Está representado por ==
- **Símbolo de diferencia:** Está representado por !=
- **Símbolo de mayor igual:** Está representado por >=
- **Símbolo de menor igual:** Está representado por <=
- **Símbolo de mayor:** Está representado por >
- **Símbolo de menor:** Está representado por <

Declaración del MAIN

```
int main () {  
  
endmain }
```

Bloque de decisión IF

```
if(expresion){
```

```
//statement
```

```
}endif
```

```
if(expresion){
```

```
//statement
```

```
}or(expresion){
```

```
//statement
```

```
}else{}
```

Bloque de decision FOR

```
for(int a; 0<1>10){
```

```
//0<1>10 significa que vamos desde 0 hasta 10 con un step de 1}
```

Bloque de decisión WHILE

```
run{  
  
}while(exp)
```

Bloque SWITCH

```
switch(var){  
    case a:///do something  
        end;  
    case x:  
        end;  
    default:  
        end;  
  
}
```

LLamada a FUNCIONES

```
function_name(params);
```

Asignación

```
varname(type) ~ value;
```

ejemplo:

```
counter(int) ~ c1 + 10;
```

Declaración de FUNCIONES

```
func int nombrefuncion(var x, var y){
```

```
    Return int
```

```
}
```

```
int thefunction(x,y){
```

```
    if(foo){
```

```
        X += y;
```

```
    }or(bar){
```

```
        thefunction(x+2, y+1);
```

```
    }or(asdasd){
```

```
        //do something
```

```
    }else{
```

```
    }endif
```

```
        Return 0;
```

```
}
```

Comentario

comentario para línea de código

#26

Comentario para bloque de código

/#

Comment

#/

Herramientas Adicionales

Para generar el AST se tiene que tener instalado graphviz

Ejemplos

```
int main(){
```

```
    c1(int) ~ 10;
```

```
    c2(int) ~ 25;
```

```
    if(c1==c2 and c2<30){
```

```
        c2 ~ 14;
```

```
    }endif;
```

```
    if(11>=c1){
```

```
        switch(id2){
```

```
            case 23:
```

```
                id2 ~ 45;
```

```
            end;
```

```
            case 45:
```

```
                end;
```

```
            default:
```

```
                end;
```

```
        }
```

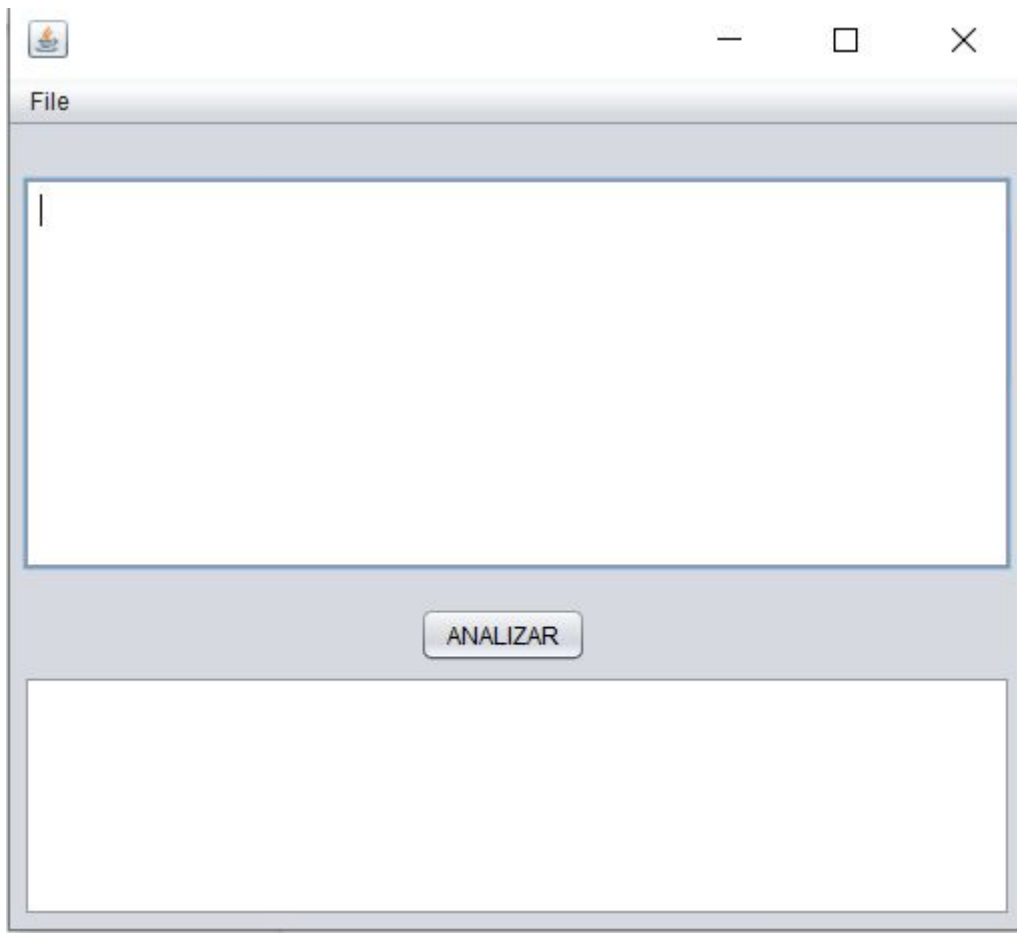
```
    }or(c2){
```



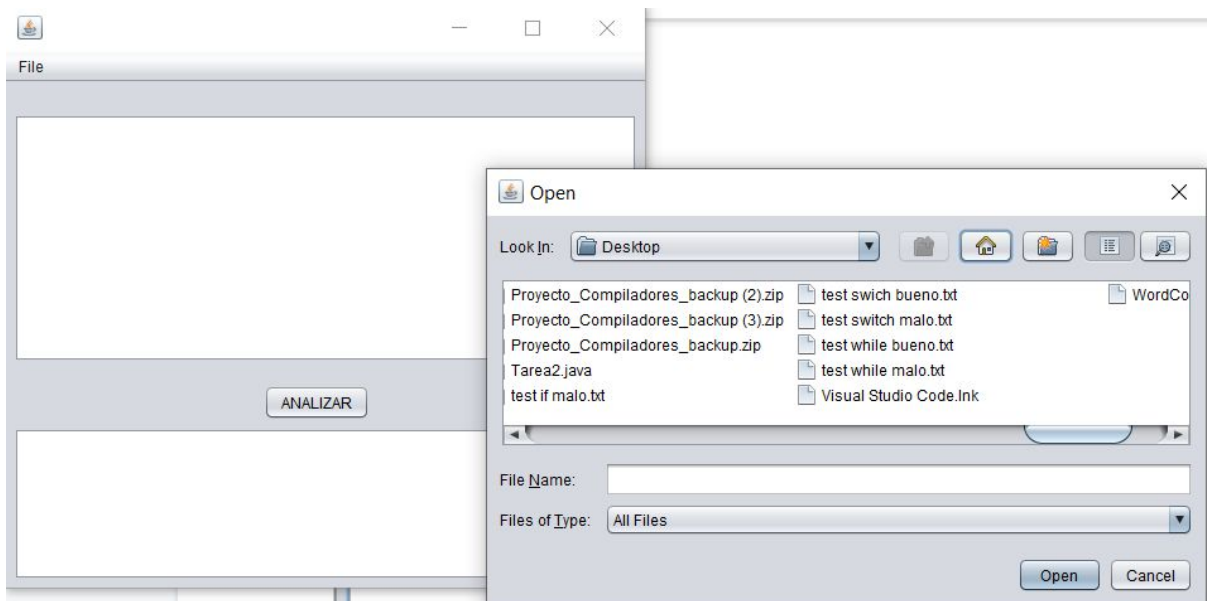
```
for(int a2; 10<1>21){  
  
    a3 ~ a3+a2;  
  
}  
  
}or(c1){  
  
    a2 ~ (10+(a3*b2)/23);  
  
}else{  
  
    run{  
  
        if(a2 != a3){  
  
            print(a2);  
  
        }endif;  
  
    }while (a3 > a3);  
  
}endif;  
  
endmain}
```

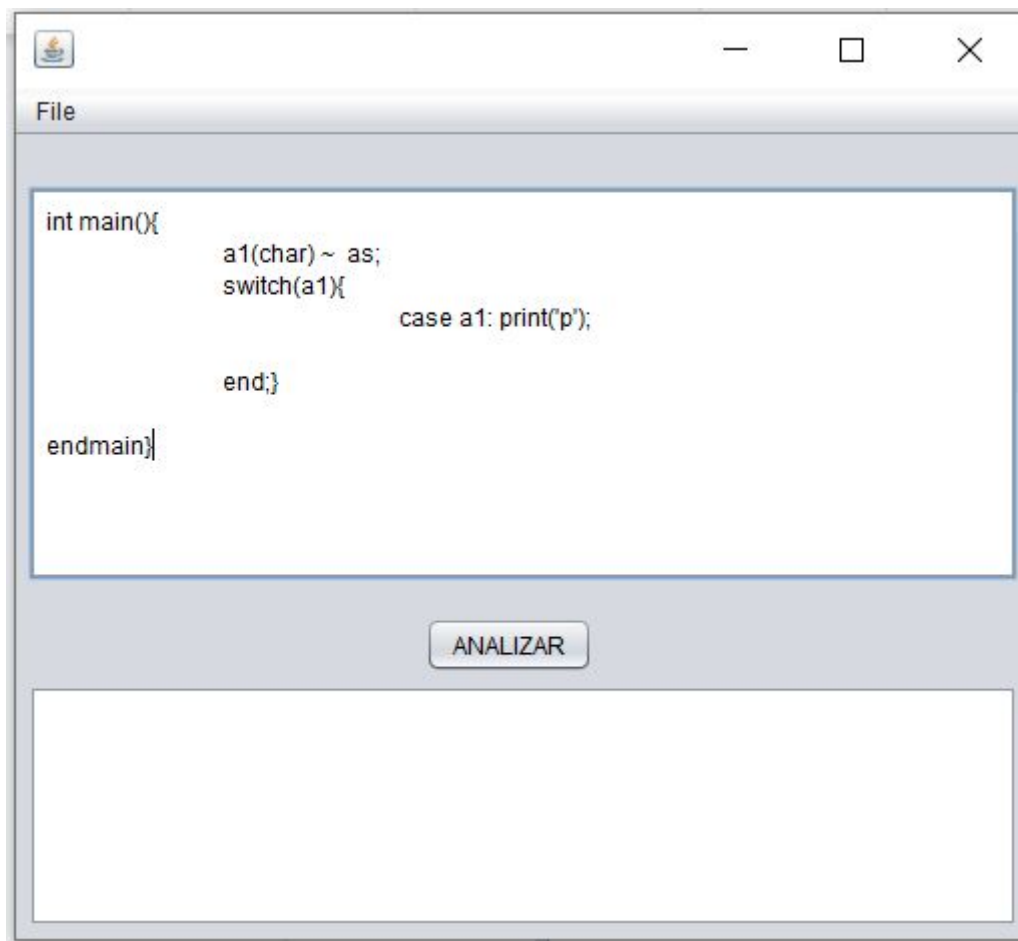
OutPut

Paso 1: aparecerá una ventana en cual se puede ingresar código o se puede buscar el archivo

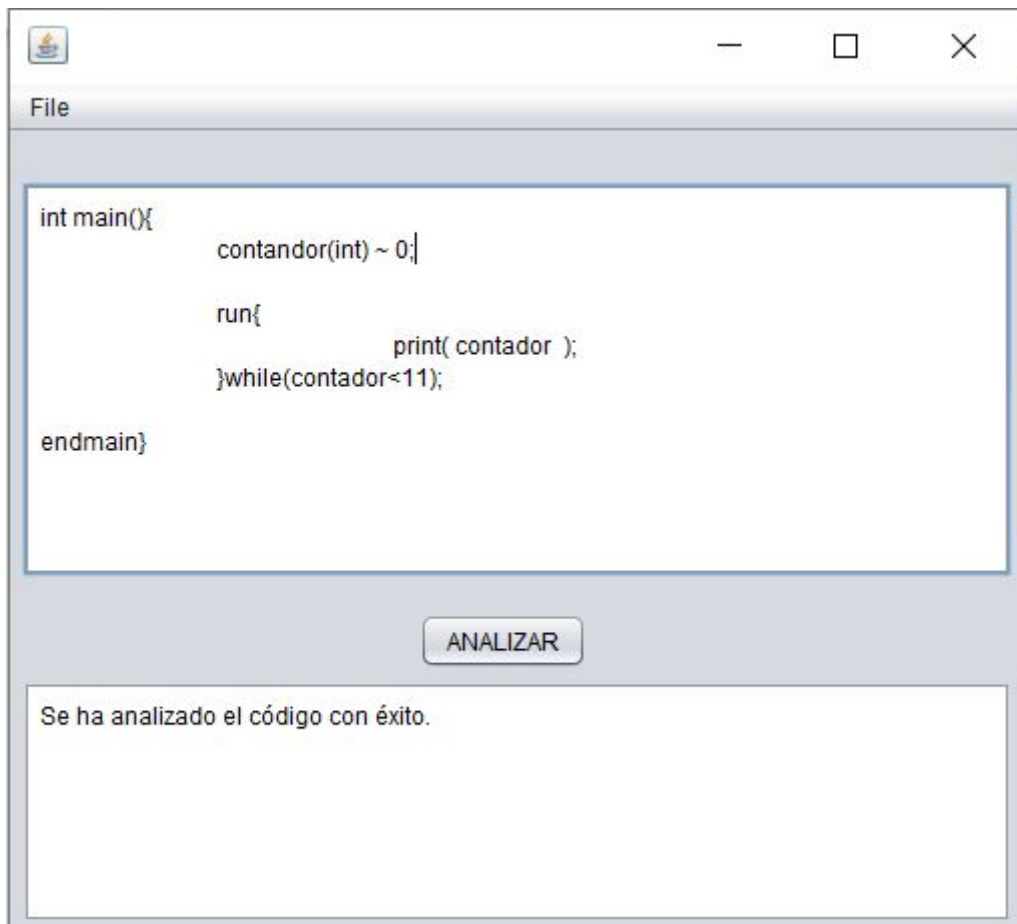


Paso 2: buscar el archivo que se desea analizar

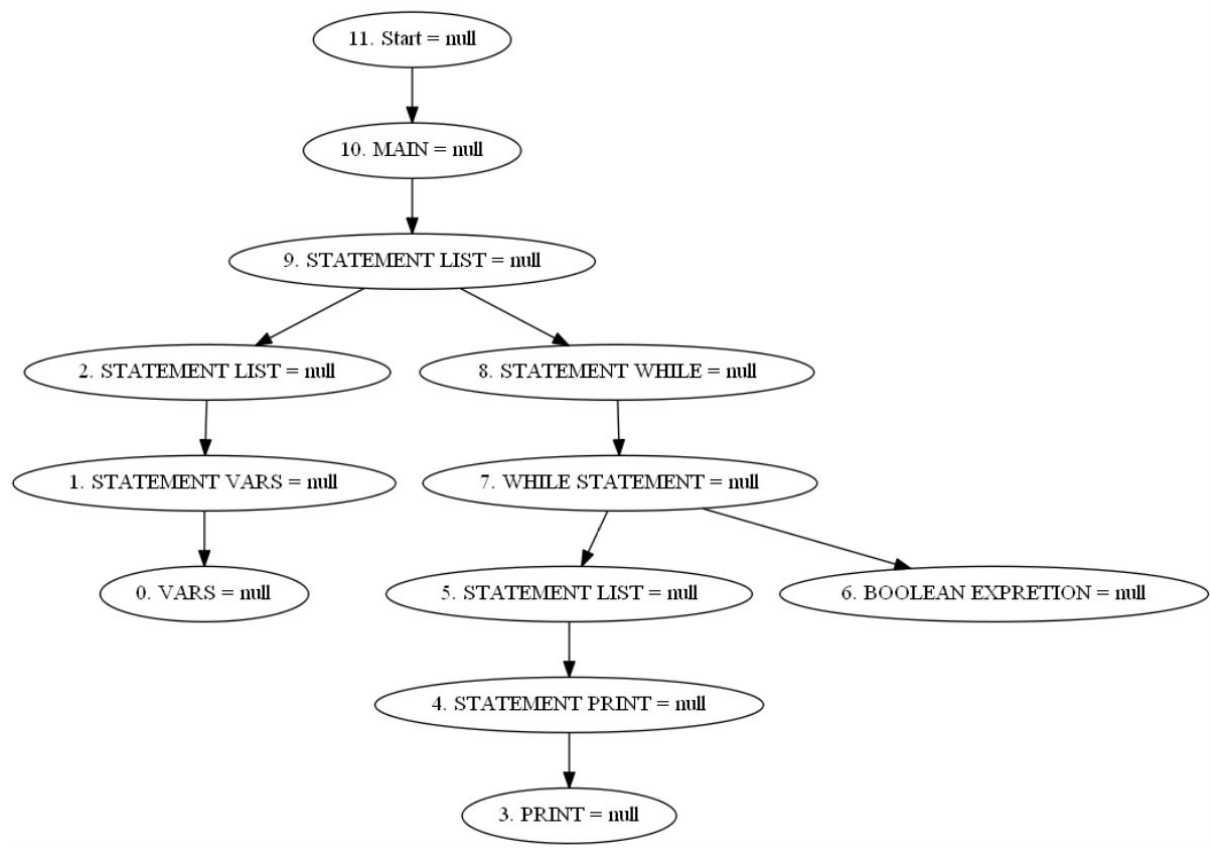




Paso 4: Analizar el segmento de código



Al momento de presionar el botón de analizar, este genera el AST automáticamente



Paso 5: Al momento de encontrar errores, este indica en qué parte fueron encontrados

