Tecnológico de Monterrey

Diseño de Compiladores

(Gpo 1)

Febrero - Julio 2022

Ing. Elda Quiroga

Proyecto Final

Antonio González Menchaca A01194261

Gerardo Peart Reynoso A01194337

FIRMA ELECTRIONICA

Monterrey, Nuevo León

DESCRIPCIÓN DEL PROYECTO 1
Propósito y Alcance del Proyecto. 1
Análisis de Requerimientos y descripción de los principales Test Cases. 2
Requerimientos 2
Test Cases 2
Descripción del Proceso 3
Bitacoras 3
Commitments 3
Reflexion 4
DESCRIPCIÓN DEL LENGUAJE 4
Nombre del lenguaje. 4
Descripción genérica del lenguaje 4
Posibles Errores 4
DESCRIPCIÓN DEL COMPILADOR: 4
Equipo, Lenguaje y Librerías utilizadas. 4
Descripción del Análisis de Léxico 5
Patrones de Construcción 5
Enumeración de los "tokens" 6
Descripción del Análisis de Sintaxis 7
Descripción de Generación de Código Intermedio y Análisis Semántico. Debe incluir: 7
Descripción detallada del proceso de Administración de Memoria usado en la compilación o Especificación gráfica de CADA estructura de datos usada y su JUSTIFICACIÓN. (Dir.Func., Tablas de Var's, Cuádruplos, Pilas, etc...) 7

1. DESCRIPCIÓN DEL PROYECTO

a. Propósito y Alcance del Proyecto.

El propósito de este proyecto es implementar los conocimientos adquiridos en clase de Diseño de Compiladores con el fin de crear un compilador funcional e integrarle a este compilador un funcionamiento que lo distinga sobre los demás.

El alcance de este proyecto, como se mencionó anteriormente es diseñar y construir un compilador funcional. En este caso se va a crear un estilo dashboard donde el usuario va a poder escribir y compilar su código. En este dashboard el usuario va tener la opción de compilar y ejecutar su código ahí mismo, o igualmente podrá descargar un OBJ y ejecutar el código en otro momento. Para distinguirse de los otros compiladores estamos buscando implementar una visualización gráfica sobre los elementos del compilador (Memoria Virtual, Directorio de Funciones, Cuadruplos)

b. Análisis de Requerimientos y descripción de los principales Test Cases.

Requerimientos

Código a compilar y ejecutar	El programa va a requerir un código a ejecutar, esto se podrá hacer ahí mismo en la plataforma o ser un documento OBJ ingresado por el usuario.
Limites de memoria	Debemos tener una idea del tamaño de los programas a ejecutar para implementar una memoria del tamaño ideal.

Test Cases

Test Case	Nombre Archivo	Proposito
TC # 1	tc1.txt	Verificar el funcionamiento completo del compilador: ● Compilacion ● Ejecución ● Descarga OBJ ● Representación visual de datos
TC # 2	tc2.txt	Verificar que en caso que no pueda compilar se ejecute un error e informe al usuario
TC # 3	tc3.txt	Verificar que en caso que se presente un error durante la ejecución se detenga e informe al usuario.
TC # 4	tc1.obj	Verificar el funcionamiento completo del compilador desde un documento OBJ: ● Compilacion

		• Ejecución • Representación visual de datos
TC # 5	tc3.obj	En caso que se haya descargado un OBJ con error se debe desplegar el error al momento de ejecutar.

c. Descripción del Proceso

Bitacoras

Semana 0:
Antonio Gonzalez
Geraro Peart:
Semana 1:
Antonio Gonzalez
Geraro Peart:
Semana 2:
Antonio Gonzalez
Geraro Peart:
Semana 3:
Antonio Gonzalez
Geraro Peart:
Semana 4:
Antonio Gonzalez
Geraro Peart:
Semana 5:
Antonio Gonzalez
Geraro Peart:
Semana 6:
Antonio Gonzalez
Geraro Peart:
Semana 7:
Antonio Gonzalez
Geraro Peart:

Commitments

1. El equipo va a trabajar de manera equitativa.

2. Ambos integrantes deben estar familiarizados con el contenido.
3. Habrá reuniones de manera semanal.
    a. Se monitorea el avance
    b. Se definen nuevos trabajos
    c. Se aclaran dudas
4. Cada quien hará el avance que tenga que hacer a tiempo y de manera correcta.
5. En caso de no poder solucionar un problema nos apoyamos entre nosotros.

Reflexion
Antonio Gonzalez

Geraro Peart:

2. DESCRIPCIÓN DEL LENGUAJE

a. Nombre del lenguaje.

El lenguaje se ha nombrado: **VizComp**

b. Descripción genérica del lenguaje

Este lenguaje se está creando como un lenguaje básico, que tendrá la capacidad de resolver expresiones, ciclos condicionales y funciones. De igual manera vamos a ofrecer al usuario un “Insight” de manera visual sobre el proceso de compilación.

c. Posibles Errores

3. DESCRIPCIÓN DEL COMPILADOR:

a. Equipo, Lenguaje y Librerías utilizadas.

Equipo de computo	Lenguaje	Librerias
PC	Python	Lex - Lexer Yacc - Parser Pandas - DataFrames (Estructura de datos) Dash - Rep Visual

b. Descripción del Análisis de Léxico

Patrones de Construcción

`t_ignore = " \t\n"`	`t_PLUS = r'\+'`	`t_MINUS = r'-'`
`t_MULT = r'\*'`	`t_MOD = r'%'`	`t_DIV = r'/'`
`t_EQUALS = r'='`	`t_SEMI = r';'`	`t_COLON = r':'`
`t_COMMA = r'\,'`	`t_LPAREN = r'\('`	`t_RPAREN = r'\)'`
`t_LBRACES = r'\{'`	`t_RBRACES = r'\}'`	`t_LBRACKET = r'\['`
`t_RBRACKET = r'\]'`	`t_LT = r'<'`	`t_GT = r'>'`
`t_LE = r'<='`	`t_GE = r'>='`	`t_EQ = r'=='`
`t_NE = r'!='`	`t_LOR = r'||'`	`t_LAND = r'&&'`
`t_CTE_I = r'\d+'`	`t_CTE_F = r'((\d*\.\d+)(E[\+-]?\d+)?|([1-9]\d*E[\+-]?\d+))'`	`t_CTE_S = r'\'.*?\''`
`t_TP = r'..'`		

```
def t_ID(t):

r'[a-zA-Z_][a-zA-Z_0-9]*
'
t.type =
reserved.get(t.value,'ID
')
     return t
```

```
def t_error(t):
print("Illegal character
%s" % repr(t.value[0]))
    t.lexer.skip(1)
```

```
def t_newline(t):
    r'\n+'
    t.lexer.lineno +=
t.value.count("\n")
```

Enumeración de los "tokens"

Reserved Tokens

```
    'if' : 'IF',
    'else' : 'ELSE',
    'elif' : 'ELSEIF',
    'for' : 'FOR',
    'while' : 'WHILE',
    'do' : 'DO',
    'to' : 'TO',
    'programa' : 'PROGRAMA',
    'principal' : 'PRINCIPAL',
    'return' : 'RETURN',
    'lee' : 'LEE',
    'escribir' : 'ESCRIBIR',
    'int'  : 'INT',
    'float' : 'FLOAT',
    'char' : 'CHAR',
    'funcion' : 'FUNCION',
    'void'  : 'VOID',
    'bool' : 'BOOL',
    'atributos': 'ATRIBUTOS',
    'metodos' :'METODOS'
```

Tokens

```
'ID', 'PLUS' , 'MINUS' , 'MULT', 'MOD',
'DIV', 'EQUALS', 'COLON', 'SEMI', 'COMMA',
'LPAREN','RPAREN', 'LBRACKET', 'RBRACKET',
'LBRACES', 'RBRACES', 'LT', 'LE', 'GT',
'GE', 'EQ', 'NE', 'RANGE', 'LOR', 'LAND',
'CTE_I', 'CTE_F', 'CTE_S'
```

c. Descripción del Análisis de Sintaxis

Nombre	Gramatica Formal
< Programa >	Programa → programa ID ; Programa2 Programa2 → *ATRIBUTOS* Programa2 | Programa3 Programa3 → *METODOS* Programa3 | *PRINCIPAL*
< Atributos >	Atributos → atributos *VAR*
< Metodos >	Metodos → metodos *FUNCIONES*
< Principal >	Principal → principal (*ESTATUTOS* Principal2 Principal2 → *ESTATUTOS* Principal2 |)
< Funciones >	Funciones → *TIPO* ID Funciones2 \ void ID Funciones2 Funciones2 → (*PARAMETROS*) { *ESTATUTOS* }
< Var >	Var → *TIPO* : Var2 Var2 → ID Var3 | ID [Cte-i] Var3 | ID [Cte-i] [Cte-i] Var3 Var3 → , Var2 | ; Var | ;
< Parametros >	Parametros → *TIPO* ID , Parametros | *TIPO* ID
< Tipo>	Int | Float | Char | Bool
< EXP >	EXP → *T-EXP* | *T-EXP* ||
< T-EXP >	T-EXP → *G-EXP* | *G-EXP &&*
< G-EXP >	G-EXP → *M-EXP* G-EXP2 *M-EXP* G-EXP2 → > | < | >= | <= | == | !+
< M-EXP >	M-EXP → *T* M-EXP2 | *T* M-EXP2 → + M-EXP | - M-EXP
< T >	T → *F* T2 | *F* T2 → * T | /T
< F >	F → (*EXP*) | Cte-i | Cte-f | Cte-c| *VAR* | *LLAMADA*
< Estatutos >	Estatutos → *ASIGNA* | *LLAMADA* | *LEE* | *ESCRIBE* | *CONDICION* | *CICLO-W* | *CICLO-F*
< Asigna >	Asigna → *VAR* = *EXP*
< Llamada >	Llamada → ID (*EXP* Llamada2 Llamada2 → , *EXP* Llamada2 |)

< Lee >	Lee → lee (Lee2 Lee2 → Cte-i) | Cte-f) | Cte-c)
< Escribe >	Escribe → escribir (*EXP* Escribe2 Escribe2 → , *EXP* |)
< Condicion >	Condicion → if (*EXP*) Condicion2 | if (*EXP*) Condicion3 Condicion2 → elif (*EXP*) Condicion2 | Condicion3 Condicion3 → { *ESTATUTO* } | else { *ESTATUTO* }
< Ciclo-W >	Ciclo-W → while Ciclo-W2 | do Ciclo-W3 Ciclo-W2 → (*EXP*) { *ESTATUTOS* } Ciclo-W2 | (*EXP*) { *ESTATUTOS* } Ciclo-W3 → { *ESTATUTOS* } while (*EXP*) Ciclo-W3 | { *ESTATUTOS* } while (*EXP*)
< Ciclo-F >	Ciclo-F → (for Ciclo-F2 Ciclo-F2 → Ciclo-F3 Ciclo-F2 | Ciclo-F3 Ciclo-F3 → *VAR* = *EXP* to do) { *ESTATUTOS* }
< llamarArr>	FALTA LLAMAR ARREGLOS???

d. Descripción de Generación de Código Intermedio y Análisis Semántico. Debe incluir:
   i. Código de operación y direcciones virtuales asociadas a los elementos del código.
   ii. Diagramas de Sintaxis con las acciones correspondientes marcadas sobre ellos (puntos neurálgicos).
   iii. Breve descripción de cada una de las acciones semánticas y de generación de código (no más de 2 líneas).
   iv. Tabla de consideraciones semánticas (combinaciones factibles y errores de tipo)
e. Descripción detallada del proceso de Administración de Memoria usado en la compilación o Especificación gráfica de CADA estructura de datos usada y su JUSTIFICACIÓN. (Dir.Func., Tablas de Var's, Cuádruplos, Pilas, etc...)

4. DESCRIPCION DE LA MÁQUINA VIRTUAL:
    a. Equipo de cómputo, lenguaje y utilerías especiales usadas (en caso de ser diferente que el compilador).
    b. Descripción detallada del proceso de Administración de Memoria en ejecución (Arquitectura). Incluir:
        i. Especificación gráfica de CADA estructura de datos usada para manejo de scopes y su JUSTIFICACIÓN. (Memoria Local, global, etc..)
        ii. Asociación hecha entre las direcciones virtuales (compliación) y las reales (ejecución).

5. PRUEBAS DEL FUNCIONAMIENTO DEL LENGUAJE :
    a. Incluir pruebas que "comprueben" el funcionamiento del proyecto:
        i. Codificación de la prueba (en su lenguaje).
        ii. Resultados arrojados por la generación de código intermedio y por la ejecución.

6. DOCUMENTACIÓN DEL CÓDIGO DEL PROYECTO:
    a. Incluir comentarios de Documentación, es decir: para cada sección/módulo, una pequeña explicación de qué hace, qué parámetros recibe, qué genera como salida y cuáles son los módulos más importantes que hacen uso de él.
    b. Dentro de los módulos principales se esperan comentarios de Implementación, es decir: pequeña descripción de cuál es la función de algún estatuto que sea importante de ese módulo.