

**Noobscript**

Diseño de Compiladores

Dr. José Icaza / Ing. Elda Quiroga

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Víctor Manuel Clemente López Jorge Antonio Carrillo Curiel

A00800053 A0113698

Monterrey Nuevo León, a 6 de mayo 2014

1. **Descripción y documentación del proyecto**
   1. **Visión**

Definir como objeto de estudio la creación del proyecto de programación, consistente en crear un nuevo lenguaje de programación orientado a objetos el cual considera todos los aspectos y elementos básicos de un lenguaje de una manera más sencilla para que, de esta manera, se agilice el entendimiento de los mismos y dar una visión más clara a el estudio abstracto del desarrollo computacional.

* 1. **Objetivo de el lenguaje y alcance**

El estudio y desarrollo de lenguajes de programación es un estudio abstracto y complicado desde el punto de vista de lógica, ya que al contener elementos como ciclos o condiciones su aprendizaje puede llegar a ser altamente complicado. Se pretende diseñar un lenguaje de programación básico orientado objetos que permita la definición de clases con atributos y métodos de una manera más sencilla, para de este modo, agilizar el proceso de aprendizaje de los mismos en un ambiente de programación más amistoso orientado a la educación.

* 1. **Análisis de requerimientos y casos de uso**
  2. **Casos de Prueba**
  3. **Proceso General**

1. **Descripción del lenguaje**
   1. **Nombre de el Lenguaje**

Noobscript

* 1. **Descripción genérica de características de el lenguaje**

Noobscript nos permite, por medio de un léxico sencillo, la capacidad de realizar operaciones básicas, esto incluye la capacidad de crear estatutos de asignación, condición, ciclo y escritura. Estos realizados con una aritmética de tipos fácil de entender por el lenguaje diseñado que permite realizar operaciones aritméticas sin que el usuario final se pierda.

De igual manera, el lenguaje permite manejo de funciones, las cuales manejan tipos de variables como globales, locales o temporales dependiendo ya de la necesidad de el programador, asimismo el manejo de datos estructurados.

* 1. **Posibles errores**

Los posibles errores principales que pueden ocurrir durante la compilación de noobscript yacen principalmente en la sintaxis de el lenguaje, ya que al ser un lenguaje de bajo nivel y enfocado ala educación, muestra los errores mala sintaxis para que el programador este consciente de lo que sucede y le permita mejorar en el entendimiento de su error fácilmente.

AGREGAR MAS

1. **Descripción del compilador** 
   1. **Hardware y Software**

Se considera como plataforma de desarrollo el lenguaje de programación python 3.4.3 , ya que es un lenguaje fácil de aprender y de igual manera hace hincapié en una sintaxis que favorezca a un código legible. Este, al ser un lenguaje de alto nivel, es orientado a objetos, interpretado e interactivo y esto se considera esto como base y filosofía del lenguaje que se pretende crear. Este, será desarrollado en el sistema operativo Mac 10.10.3 (Yosemite) que es el mas actual y nos permite mayor capacidad de procesamiento para evitar errores de cualquier tipo.

* 1. **Descripción del análisis léxico**

Principales Componentes de Léxico

Dígitos

D = {1,2,3,4,5,6,7,8,9}

Abecedario

L = {a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,,v,w,x,y,z,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z}

Operadores

O = { + , - , \* , / , < , >, =, == }

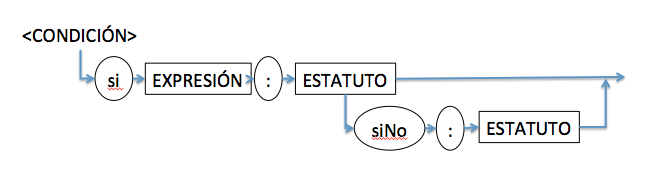
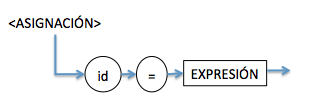
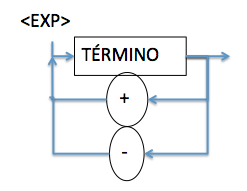
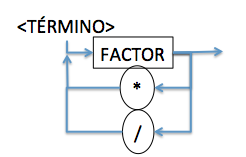
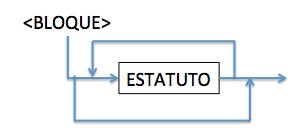
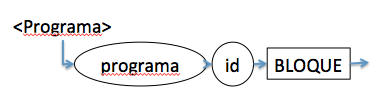
Delimitadores

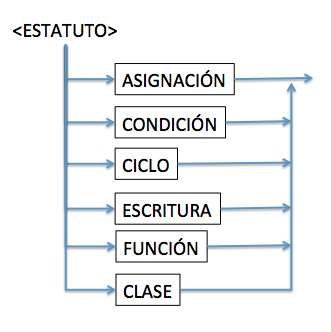
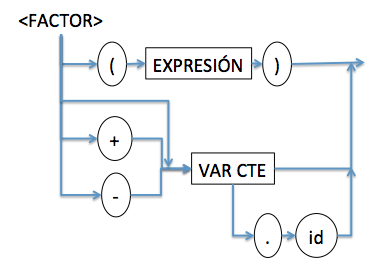
De = { ( , ) , [ , ] , ; , : , : ,. , “}”, “{“ }

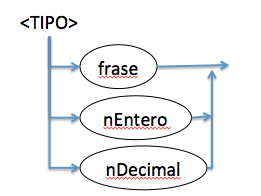
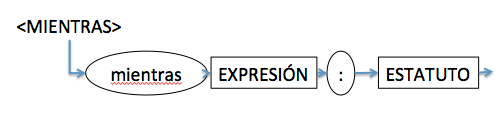
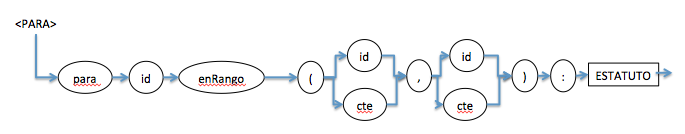
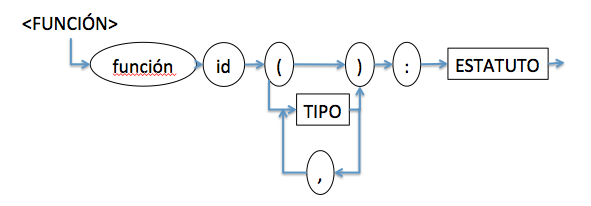
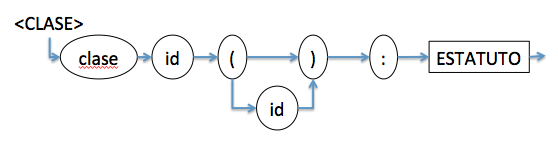
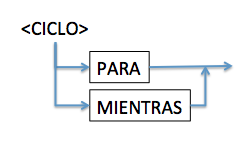
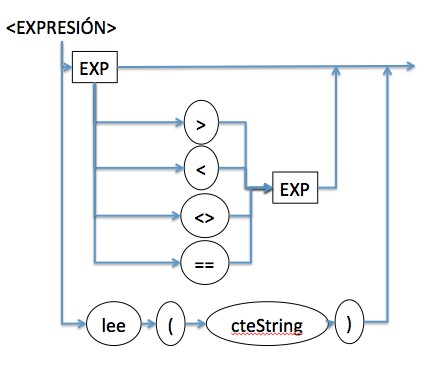
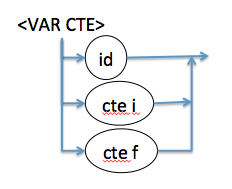
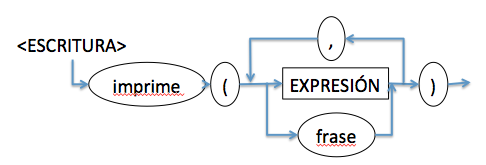
Palabras reservadas

|  |  |
| --- | --- |
| palabra | Explicación - Uso |
| entero | Número Entero |
| decimal | Número Decimal (float) |
| frase | Cadena de caracteres |
| imprime | Impresión en pantalla |
| lee | Lectura de captura en pantalla |
| clase | class |
| vGlobal | Variable global |
| esVerdad | Valor verdadero (boolean) |
| regresa | Return de la función |
| programa | Inicio del código de programación |
| funcion | Inicio de una función |
| enRango | Formato de ciclo para denotar rango |
| ciclo | Inicio de un ciclo tipo for |
| mientras | Inicio de un ciclo tipo while |
| si | Inicio de una condición |
| siNo | Equivalente a else |

* 1. **Descripción de el análisis de sintaxis**







* 1. **Descripción de el análisis semántico FALTAN DIAGRAMS**

Basamos las consideraciones semánticas en las operaciones básicas aritméticas restringiendo su compatibilidad respecto a los lenguajes mas comunes para su aprendizaje.

Operadores

+ 1

- 2

\* 3

/ 4

< 5

> 6

== 7

= 8

( 9

) 10

GotoF 20

GotoV 21

Goto 22

ERA 30

Param 31

GOsub 32

Write 33

TEMPFRASE =6500

//constantes

CONSTENTERO = 7000

CONSTDECIMAL = 7500

CONSTesVerdad = 8000

CONSTFRASE = 8500

Tabla de consideraciones semánticas

entero + entero entero

decimal decimal

esVerdad E

frase E

entero - entero entero

decimal decimal

esVerdad E

frase E

entero \* entero entero

decimal decimal

esVerdad E

frase E

entero / entero decimal

decimal decimal

esVerdad E

frase E

entero = entero entero

decimal E

esVerdad E

frase E

entero == entero esVerdad

decimal esVerdad

esVerdad E

frase E

entero < entero esVerdad

> decimal esVerdad

esVerdad E

frase E

decimal + entero decimal

decimal decimal

esVerdad E

frase E

decimal - entero decimal

decimal decimal

esVerdad E

frase E

decimal \* entero decimal

decimal decimal

esVerdad E

frase E

decimal / entero decimal

decimal decimal

esVerdad E

frase E

decimal = entero decimal

decimal decimal

esVerdad E

frase E

decimal == entero esVerdad

decimal esVerdad

esVerdad E

frase E

decimal < entero esVerdad

> decimal esVerdad

esVerdad E

frase E

esVerdad + entero E

decimal E

esVerdad E

frase E

esVerdad - entero E

decimal E

esVerdad E

frase E

esVerdad \* entero E

decimal E

esVerdad E

frase E

esVerdad / entero E

decimal E

esVerdad E

frase E

esVerdad = entero E

decimal E

esVerdad esVerdad

frase E

esVerdad == entero E

decimal E

esVerdad esVerdad

frase E

esVerdad < entero E

> decimal E

esVerdad E

frase E

frase + entero E

decimal E

esVerdad E

frase E

frase - entero E

decimal E

esVerdad E

frase E

frase \* entero E

decimal E

esVerdad E

frase E

frase / entero E

decimal E

esVerdad E

frase E

frase = entero E

decimal E

esVerdad E

frase frase

frase == entero E

decimal E

esVerdad E

frase E

frase < entero E

> decimal E

== esVerdad E

frase E

Pseudo Código Cubo Semantico

char cubo[4][4][4 ]=

{

{

//Suma +

//entero + op2

//{entero,decimal,esVerdad,Frase}

{en, d, e, e},

//decimal + op2

{d, d, e, e},

//esVerdad + op2

{e, e, e, e},

//frase + op2

{e, e, e, e}

},

{

//Resta -

//entero - op2

{en, d, e, e},

//decimal - op2

{d, d, e, e},

//esVerdad - op2

{e, e, e, e},

//frase - op2

{e, e, e, e},

},

{

//Multiplicacion \*

//entero \* op2

{en, d, e, e},

//decimal \* op2

{d, d, e, e},

//esVerdad \* op2

{e, e, e, e},

//frase \* op2

{e, e, e, e},

},

{

//Division /

//entero / op2

{d, d, e, e},

//decimal / op2

{d, d, e, e},

//esVerdad / op2

{e, e, e, e},

//frase / op2

{e, e, e, e},

},

{

//Igualdad =

//int = op2

{en, en, e, e},

//decimal = op2

{d,d, e, e},

//esVerdad = op2

{e, e, es, e},

//frase = op2

{e, e, e, e}

},

{

//comparativo ==

//entero == op2

{es, es, e, e},

//decimal == op2

{es, es, e, e},

//esVerdad == op2

{e, e, es, e},

//frase == op2

{e, e, e, e}

},

{

//comparativos < >

//entero <> op2

{es, es, e, e},

//decimal <> op2

{es, es, e, e},

//esVerdad <> op2

{e, e, e, e},

//frase <> op2

{e, e, e, e}

},

};

* 1. **Administración de memoria de el compilador**

Direcciones Virtuales asociadas con elementos de el código

//Globales

GLOBENTERO = 1000

GLOBDECIMAL =1500

GLOBesVerdad =2000

GLOBFRASE =2500

//Locales

LOCALENTERO =3000

LOCALDECIMAL =3500

LOCALesVerdad =4000

LOCALFRASE =4500

//Temporales

TEMPENTERO =5000

TEMPDECIMAL =5500

TEMPesVerdad =6000

1. **Descripción de la maquina virtual**
   1. **Equipo**

La maquina virtual fue desarrollada en el lenguaje Python de igual manera por su sencillez en administración de listas y lectura de archivos, lo cual permitió la fácil lectura de el archivo de cuádruplos

* 1. **Administración de memoria**

1. **Pruebas de funcionamiento de el lenguaje**

Primordialmente se muestran pruebas que aseguran el funcionamiento optimo de el lenguaje.

Compatibilidad de tipos

1. **Documentación de el proyecto**

Quad.py

Codigo que realiza las operaciones aritméticas y las deposita en el archivo de texto para que sean

from cubo import check, tipotemp

cont=1

contemp=0

cuadruplos = [[0 for x in range(5)] for x in range(300)]

#Crear las pilas e inicializarlas

#pila de operadores

PilaO = [] #pila de operandos

PSaltos = [] #pila de saltos

POper = [] #pila de operadores

cuadf = None

def init():

# PilaO=createStack(100); #pila de operandos

# PSaltos=createStack(100); #pila de saltos

# POper=createStack(100); #pila de operadores

global cuadf

cuadf = open("cuadruplos.txt", 'w')

if cuadf == None:

print( "Archivo cuadruplos.txt no se ha podido abrir");

exit(1);

def printcuadruplos():

for i in range(300):

for j in range(5):

if cuadruplos[i][j]== 0:

exit(1)

global cuadf

cuadf.write(repr(cuadruplos[i][j]) + " ")

cuadf.write("\n") ;

#impresion de de direccion en memoria y de valor asignado de la variable dependiendo de su tipo

def escribe\_ctei(dire, val):

cuadf.write("%d %d %d\n",s,dire,val)

def escribe\_ctef(dire, val):

cuadf.write("%d %d %d\n",s,dire,val)

def escribe\_ctes(dire, val):

cuadf.write("%d %d %d\n",s,dire,val)

def escribe\_cuad(cuadro, ope, oper1, oper2, res):

print("Imprimiendo:", repr(cuadro),repr(ope), repr(oper1), repr(oper2), repr(res))

cuadruplos[cuadro-1][0] = cuadro

cuadruplos[cuadro-1][1] = ope

cuadruplos[cuadro-1][2] = oper1

cuadruplos[cuadro-1][3] = oper2

cuadruplos[cuadro-1][4] = res

global cont

cont = cont + 1

#1

def pila\_id(oper):

PilaO.append(oper)

print( "Ya meti oper \n", repr(oper))

#2

def pila\_op(op):

POper.append(op)

print( "ya meti op \n", repr(op))

#3

def parentesisPush():

POper.append(9) #genera pared falsa en pila

def parentesisPop():

POper.pop #saca de la pila

#5 ARREGLAR

def termino():

global contemp

print( "Estoy en cuadruplo termino\n")

op = POper[-1]

POper.pop()

if op == 3 or op == 4: #operadores \* o /

oper2 = PilaO[-1]

PilaO.pop()

oper1 = PilaO[-1]

PilaO.pop()

if (check(op, oper1, oper2)): #checa si es valido

escribe\_cuad(cont, op, oper1, oper2, contemp+tipotemp()) #genera el cuadruplo

pila\_id(contemp+tipotemp()) #mete el resultado a la pila de operadores

contemp = contemp+1

else: #marca error si no es compatible

print( "\*\*\*ERROR DE TIPOS\*\*\*\* termino\n")

exit(1)

else:

fprintf( "No es termino pasar al siguiente \n")

pila\_op(op)

#6

def expresion():

global contemp

print( "Estoy en cuadruplo expresion osea suma y resta")

op = POper[-1]

POper.pop()

if op == 1 or op == 2: #operadores + o -

oper2 = PilaO[-1]

PilaO.pop()

oper1 = PilaO[-1]

PilaO.pop()

if check(op, oper1, oper2): #checa el tipo

escribe\_cuad(cont, op, oper1, oper2, contemp+tipotemp()) #genera el cuadruplo

pila\_id(contemp+tipotemp()) #mete el resultado a la pila de operadores

contemp = contemp+1

else: #marca error si no es compatible

print( "\*\*\*ERROR DE TIPOS\*\*\*\* expresion\n")

exit(1)

else:

print( "No es expresion pasar al siguiente \n")

pila\_op(op)

#7

def relacional():

print( "Estoy en cuadruplo relacional\n ");

global contemp

op = POper[-1]

POper.pop()

if op == 5 or op == 6 or op == 7 : # op == < || op == > || op = ==

oper2 = PilaO[-1]

PilaO.pop()

oper1 = PilaO[-1]

PilaO.pop()

if check(op, oper1, oper2): #checa el tipo

escribe\_cuad(cont, op, oper1, oper2, contemp+tipotemp())

pila\_id(contemp+tipotemp()); #mete el resultado a la pila de operadores

contemp = contemp + 1

else: #marca error si no es compatible

print( "\*\*\*ERROR DE TIPOS\*\*\*\* relacional\n");

exit(1)

else:

print( "No es relacional pasar al siguiente \n");

pila\_op(op)

#8

def assign():

global contemp

print("Estoy en quad assign\n")

op = POper[-1]

POper.pop()

if op == 8: #si el operador es =

oper2 = PilaO[-1]

PilaO.pop()

oper1 = PilaO[-1]

PilaO.pop()

if check(op, oper1, oper2): #checa el tipo

escribe\_cuad(cont, op, oper2, -1, oper1) #genera el cuadruplo

contemp = contemp + 1

else: #marca error si no es compatible

print( "Error de tipos en assign\n")

exit(1)

else:

print( "No es asignacion pasar al siguiente \n")

pila\_op(op)

#15

def logico():

global contemp

print( "Estoy en cuad logico\n");

op = POper[-1]

POper.pop()

if op == 8 or op == 9: # op == & || op == |

oper2 = PilaO[-1]

PilaO.pop()

oper1 = PilaO[-1]

PilaO.pop()

if check(op, oper1, oper2): #checa el tipo

escribe\_cuad(cont, op, oper1, oper2, (contemp+tipotemp()))

pila\_id(contemp+tipotemp()) # mete el resultado a la pila de operadores

contemp = contemp + 1

else: #marca error si no es compatible

print( "\*\*\*ERROR DE TIPOS\*\*\*\* logico\n")

exit(1)

else:

print( "No es logico, pasar al siguiente \n")

pila\_op(op)

#10

def if1():

global cont

print( "Estoy en cuadruplo if\n")

#if

#1.- genera gotoF y mete cont-1 a la pila de saltos

op = 20

oper1 = PilaO[-1]

if per1 < 10000 or oper1 >= 11000:

print ("\*\*\*ERROR DE TIPOS\*\*\*\* IF\n") #STANDAR ERROR DE PYTHON CHECAR

else:

POper.pop()

escribe\_cuad(cont, op, oper1, -1, -1)

PSaltos.append(cont-1)

#11

def else1():

global cont

#2genera goto (else), y saca falso de la pila y rellena el primer salto

op = 22 #goto

escribe\_cuad(cont, op, -1, -1, -1)

falso = PSaltos[-1]

cuadruplos[falso-1][4]=cont

PSaltos.append(cont-1)

#12

def if2():

fin = PSaltos[-1]

PSaltos.pop()

cuadruplos[fin-1][4]=cont

#13

def do1():

#do-while

#1.- mete cont a pila de saltos

PSaltos.append(cont)

#14

def do2():

#2.- genra gotoV e incrementa el cont

op = 21 #gotov

oper1 = Pila0[-1]

oper2 = -1

escribe\_cuad(cont, op, oper1, -1, contemp)

def print1():

op=33

POper.append(op)

def print2():

oper1=top(PilaO)

escribe\_cuad(cont,top(POper),oper1,-1,-1)

def cuadproc():

return cont

1. **Bibliografía**

Welcome to Python.org. (n.d.). Retrieved March 4, 2015, from https://www.python.org/

Tutorials Point Simply Easy Learning. (n.d.). Retrieved March 2, 2015, from <http://www.tutorialspoint.com/python/index.htm>

Viswanathan, B. (2014, January 18). Quora. Retrieved March 1, 2015, from <http://www.quora.com/Why-is-programming-and-coding-so-hard>

1. The way of the program. (n.d.). Retrieved March 2, 2015, from http://www.openbookproject.net/thinkcs/python/english2e/ch01.html