Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Compiladores e Intérpretes

Profesor Kirstein Gätjens Soto

Etapa 1 del Compilador

Estudiantes

Adrián Garnier Artiñano

Steven Moya Quiñones - 2017158678

Fecha de entrega

31 de marzo de 2019

I Semestre, 2019

Tabla de contenidos

Documentación del lenguaje

Tipos simples

Tipo	**Descripción**	**Ejemplos literal**
numerus	Número entero con signo de 16 bits.	123 -978 0x3ef 0rXXXIV
fractio	Número racional de la forma $\frac{a}{b}$. a y $b$son dos números enteros con signo de 16 bits cada uno. Siempre se guarda de forma simplificada	1|2 -4|5 1|100
catena	Representación de una secuencia de caracteres. Posee un máximo de 255 caracteres y mide 32 bytes.	“Hola mundo”
imago	Carácter ASCII de 7 bits. Mide 1 byte.	‘A’ ‘Z’ ‘1’
gregorius	Fecha del calendario gregoriano. Mide 3 bytes.	2019$5$23 1988$5$1
liber	Archivo. Mide 128 bytes.	*No posee literal.*
dualis	Representa el concepto de verdadero y falso. Mide 1 byte.	veridicus falsidicus

Caracteres de escape

`\n`	Salto de línea
`\t`	Tabulador
`\'`	Comilla simple
`\"`	Comilla doble
`\r`	Retorno del carro

Tipos compuesto

Tipo	Descripción	Ejemplos literal
Arreglos	Colección de datos. Se declara de la forma: ordo [size][size]...[size] autem tipo. Ejemplo: ordo [2][3] autem numerus ordo [5][1][3] autem imago.	{1,2,3,4,5} {{1,2},{3,4},{5,6}} {“A1”,”A2”}
Registros	Colección de valores de distintos tipo agrupados. Se declara de la forma coniugo *tipo* nombre1 . *tipo* nombre2. ... *tipo* nombre_n . dixi Ejemplo: coniugo numerus X . numerus Y . dixi	<-1,”A”,falsificus-> <-23,2019$1$1,{1,2}->

Secciones

El lenguaje posee un total de seis secciones. Estas secciones contienen las distintas partes que conforman un programa. Las secciones son:

nomen

Esta sección es obligatoria. Aquí se coloca el nombre del programa. Ejemeplo:

```
nomen example
```

perpetus

El lenguaje hace distinción de constantes y variables. Las primeras se declaran en esta sección.

Sintaxis:

```
perpetus
      tipo var1 dito valor .
      tipo var2 dito valor .
      ...
      tipo var_n dito valor .
```

Ejemplo:

```
perpetus
```

```
numerus A sum dito 5.
fractio F sum dito 3|7.
```

furibundus

En esta sección declara tipos personalizados por el usuario.

Sintaxis:

```
furibundus
      Nombre1 sum tipo .
      Nombre2 sum tipo .
      ...
      Nombre_n sum tipo .
```

Ejemplo:

```
furibundus
      Edad sum numerus .
      Mensaje sum catena .
```

commutabilis

En esta sección se declara variables. Pueden separarse por comas.

Sintaxis:

```
commutabilis
      tipo var1, var2, ..., var_n.
      tipo varA est valor .
      ...
      tipo var_X est valor .
```

Ejemplo:

```
commutabilis
      numerus A est 5, B, C est 89 .
      imago Z .
```

exemplar

En esta sección se declaran los encabezados o prototipos de las rutinas.

Sintaxis:

```
exemplar
      efficio nombre (parámetros_formales): tipo_retorno .
      directus nombre (parámetros_formales) .
```

Ejemplos:

```
exemplar
      efficio fibonacci (numerus N) : numerus .
      directus imprimirMensaje (catena C) .
```

Parámetros formales

Los parámetros formales se separan con punto, se coloca primero el tipo y luego el nombre del parámetro. También existe la posibilidad de que no haya parámetros .
Ejemplo:

```
directus ejemplo (numerus N . catena C . imago I) .
```

Se podrá colocar más de un nombre por tipo. Ejemplo:

```
directus ejemplo (numerus N1, N2 . catena C1, C2 . imago I) .
```

Existe los parámetros por referencia. Se coloca la partícula *ign* antes del tipo, *ign* afecta a todos los parámetros separados por coma. Ejemplo:

```
directus ejemplo (catena C1. ign fractio F1, F2, F3) .
##F1, F2 y F3 son parámetros por referencia del tipo fractio.
```

corpus

En esta parte se declara el comportamiento de las rutinas.

Sintaxis:

```
corpus
      efficio nombre (parámetros_formales): tipo_retorno .
      declaración_variables
      bloque_código

      directus nombre (parámetros_formales) .
      declaración_variables
      bloque_código
```

Ejemplos:

```
exemplar
      efficio sumar (numerus A, B) : numerus .
      commutabilis
            numerus C .
      initum
            C := A + B .
            reditus C .
      finis

      directus imprimirMensaje (catena C) .
            scriboCatena (C) .
```

firmamentum

Es la sección principal del programa. Lo primero que se ejecuta del código.

Sintaxis:

```
firmamentum
      bloque_código
```

Ejemplo:

```
firmamentum
      A := B + C .
      B := (A * A) + 2 .
      in A > B certus
            scribonumerus(A) .
      mentiri
            scribonumerus(C) . .
```

Operadores distintos

Nombre	Sintaxis	Descripción	Ejemplo

			Código	Retorno
()	(*exp*)	Altera la precedencia	(3+5)	8
[]	arreglo[*exp*]	Acceso a arreglos	Si a = {1,4} a[0]	1
@	registro@ID	Accede al campo del registro	punto@X	
@@	catena@num	Acceso strings	"ABC"@2	'B'
incrementum	incrementum numerus	Incrementa	incrementus 5	6
decrementum	decrementum numerus	Decrementa	decrementum 5	4
non	non dualis	Negación lógica	non veridicus	falsificus
quantus	quantus *exp*	Tamaño en bytes según tipo.	quantus 34	2
[>>]	[>>] imago	Cambia a mayúscula	[>>]'a'	'A'
[<<]	[<<] imago	Cambia a minúscula	[<<]'A'	'a'
[&?]	[&?] imago	Pregunta si no es número	[&?] 'a'	veridicus
[&#]	[&#] imago	Pregunta si es número	[&#] 'a'	falsificus
&#	&# catena	Largo de string	&# "hola"	4
||	|| fractio	Parte entera	|| 3|2	1
|^	|^ fractio	Redondeo	|^ 19|10	2
lectio	lectio liber	Lee un archivo	lectio file	catena
prope	prope liber	Cierra un archivo	prope file	-
&+	catena &+ catena	Concatena dos strings	"ho" &+ "la"	"hola"
&?	catena &? imago	Retorna el índice donde esté la primera coincidencia del carácter.	"hola" &? 'h'	1
et xaut aut	dualis et dualis	Y lógico xor lógico aut lógico	veridicus et veridicus	veridicus
* / - +	n operador m	Aritméticos convencionales		

|* |/ |- |+ %				

Manejo de archivos

asignación de un archivo

```
commutabilis
      liber file .
firmamentum
      file ligo “C:\path\to\file.txt” .
```

Apertura archivo

```
commutabilis
      liber file .
firmamentum
      file ligo “C:\path\to\file.txt” .
      file patentibus modo.
##modos. Lego -> lectura, scribo -> escruitura, rescribo -> reescritua
```

Escritura de un archivo

```
commutabilis
      liber file .
firmamentum
      file ligo “C:\path\to\file.txt” .
      file patentibus rescribo .
      file scripturam “Hola mundo” .
```

Cerrar un archivo

```
commutabilis
      liber file .
firmamentum
      file ligo “C:\path\to\file.txt” .
      file patentibus rescribo .
      file scripturam “Hola mundo” .
      prope file .
```

Lectura de un archivo

```
commutabilis
      liber file .
      catena line .
firmamentum
      file ligo “C:\path\to\file.txt” .
      file patentibus lego.
      line := lectio file .
      prope file .
```

Sentencias

If

Es un If que puede tener las bifurcaciones en cualquier orden. Es decir, el then y el else van en cualquier orden y podría faltar alguno.

Sintaxis	*Ejemplo*

in *condición* certus
  bloque_código .
	in A = B certus
	  scribocatena(A igual B) . .

in *condición* mentiri
  bloque_código .
	in A = B mentiri
	  scribocatena(A distinto a B) . .

in *condición* certus
  bloque_código
mentiri
  bloque_código .
	in A = B certus
	  scribocatena(A igual B) .
	mentiri
	  scriboCatena(A distinto a B) . .

in *condición* mentiri
  bloque_código
certus
  bloque_código .
	in A = B mentiri
	  scribocatena(A distinto a B) .
	certus
	  scriboCatena(A igual a B) . .

Switch

Sintaxis	*Ejemplo*

aeger *exp*
initum
  casus 1 opus *codigo*
  …
  casus n opus *codigo*
  detrimentum opus *codigo*
finis
	aeger A
	initum
	  casus 1 opus scribocatena(“A”). neco.
	  casus 2 opus scribocatena(“B”). neco.
	  casus 3 opus scribocatena(“C”). neco.
	  detrimentum opus scribocatena(“Z”).
	finis

For

Sintaxis	*Ejemplo*

panis ID:=*exp* auctum *exp* gradus *exp* opus
bloque_codigo.
	panis I:=0 auctum 100 gradus 1 opus
	initum
	  N := I * I .
	  scribonumerus(N) .
	finis

While

Sintaxis	*Ejemplo*

tempus *exp* opus
bloque_codigo
	tempus A>0 opus
	initum
	  N +:= A + 5 .
	  decrementum A .
	finis

Do-loop until

Sintaxis	*Ejemplo*

itero
bloque_codigo
usque *exp*
	itero
	initum
	  A *:= (N+3) .
	  incrementum N .
	finis
	usque A > 34000

With

Permite acceder al contenido de los registro sin necesidad de usar el accesor.

Sintaxis	*Ejemplo*

sigla *registro* opus
  sentencia 1 .
  sentencia 2 .
  ....
  sentencia n .
dixi
	sigla Punto opus
	  X := 3 .
	  Y := 7 .
	dixi

Bloques de código

Un bloque de código puede tener de una a varias sentencias. Si solo hay una sentencia, sólo se coloca la sentencia por sí sola. Si la cantidad de sentencia es mayor o igual que dos, las sentencias se encierran con initum y finis.

Ejemplo:

```
initum
      A := 4 + 5 .
      B := A +8 .
      C := 100 %  (A + B) .
finis
```

Sentencias de flujo

Algunas sentencias permiten cambiar el flujo del código.

claudeo

Detiene la ejecución del programa.

pergo

Se salta una iteración dentro de un ciclo.

neco

Se sale de la instrucción. En caso de for, rompe el ciclo. En un switch pas ala isntrucción que le sigue al switch.

reditus

Retorna dentro de una función.

Tipo creativo

Para tipo atómico creativo se escoge tipo **fecha.** Este tipo medirá 3 bytes.

14 bits para el año	0-9999
4 bits para el mes	0-12
5 bits para el día	0-31
Total de 23 bits	

El nombre en latín corresponde a **gregorius** en honor a Gregorio XIII por establecer el calendario que lleva su nombre.
La literal de gregorius es yyyyMMdd. Ejemplo: gregorius date est 2019$8$31 .

Operaciones

Operadores unarios (nivel 1)

annus	Obtiene el año. Retorna numerus
mensis	Obtiene el mes. Retorna numerus.
dies	Obtiene el día. Retorna numerus.
nunc	Retorna la fecha actual.
$d	Retorna el número de día de la semana de una fecha
$n	Retorna el enésimo día del año de una fecha
$?	Retorna veridicus si el año de la fecha es bisiesto

Ejemplos:

```
gregorius date .
numerus year, month, day .

date := nunc date .
year := annus date .
month := mensis date .
day := dies date.
```

Operadores binarios (nivel entre 11 y 12)

$annus	Cambia el año.
$mensis	Cambia el mes.
$dies	Cambias el día.
$+	Suma dos fechas
$-	Resta dos fechas

Ejemplo:

```
gregorius date, date2 .

date := nunc date .
date2 := date $annus 1986 .
date2 := date $mensis 3 .
date2 := date $dies 28 .
date2 := date $annus 1986 $mensis 4 $dies 28 .
```

Algoritmos de conversión

gregorius a numerus

La suma del año, mes y día.

gregorius a dualis

Será true si el año es bisiesto, false si no lo es.

gregorius a catena

Se obtiene un representación en catena. Por ejemplo: 2019$3$17 es “2019/3/17”

gregorius a imago

(La suma del año, mes, día) % 256

gregorius a fractio

La conversión representa el enésimo día del año. Por ejemplo, el 3 de enero del 2019 es el tercer día del año, por tanto la conversión a fractio es 1|3.

numerus a gregorius

Sea *n* un numerus. La fecha será:
año = (n/12)+1583
mes = (n % 12) + 1

día = (n % 28) + 1

dualis a gregorius

veridicus = 1582$10$15
falsidicus = 1584$10$15

catena a gregorius

Sea *c* una catena. La fecha será:
año = 12*__largo__(*c*) + 1583
mes = (primer byte de *c* % 12) + 1
día = (segundo byte de *c* % 28) + 1

imago a gregorius

Sea *i* el número de caracter ASCII de imago. La fecha será:
año = 12 * *i* + 1583
mes = (*i* % 12) + 1
día = (*i* % 28) +1

fractio a gregorius

Sea *f* una fractio con *f* = *n*|*d*. La fecha será:
año = floor(*f*) + 1583
mes = (abs(*n*)%12)+1
día = (abs(*d*)%28)+1

Instrucciones creativas

Try-catch

No es en realidad un try-catch convencional como podría imaginarse en un lenguaje orientado a objetos. Es una instrucción que permite bifurcar el flujo de las instrucciones en un momento dado. Se le nombra como try-catch porque podría emplearse en simular el manejo de excepciones.

Sintaxis:

```
probare
      bloque_instrucciones.
      revello.
captare
      bloque_instrucciones.
```

Es decir, la instrucción tendrá dos bloques de instrucciones. En el momento en que se llame a la sentencia *revello* se ejecutará el segundo bloque de código. Si nunca se usa *revello* solo se ejecuta el primer bloque de código.

Ejemplo:

```
directus example (numerus A, B, C)
initum
      probare
      initum
            a := b - 3 .
            in a = 0 certus
                  revello . .
            c := a + (b - (c / a)) .
            in c = 0 certus
                  revello . .
            b := a / c.
            example(a, b, c).
      finis
      captare
            scribocatena ("Division by zero") .
finis
```

Intersección repetitiva

Esta instrucción utiliza dos arreglos para funcionar. Para cada elemento resultado de la intersección de los arreglos, se hace una iteración.

Sinatxis:

```
decusis expresión cum expresión per variable opus
      bloque_instrucciones
```

Ejemplo:

```
commutabilis
      numerus I, J.
      ordo[6] autem numerus A est {1,2,3,5,10,17} .
      ordo[3] autem numerus B est {1,5,10} .
      ordo[9] autem C .
firmamentum
decusis A cum B per I opus
      scribonumerus(I) .

##Ejemplo en combinación con el try catch. Unión de arreglos si y
##solo si la intersección es vacía
probare
initum
      decusis A cum B per I opus
            revello .
      panis I:=0 auctum 9 gradus 1 opus
      initum
            in I < 6 certus
                  C[I]:=A[I] .
            mentiri
                  C[I]:=B[I%3] . .
      finis
finis
captare
initum
      scribocatena("Hay intersección. No se puede hacer unión") .
      scribocatena("Se genera nuevo arreglo") .
      panis I:=0 auctum 9 gradus 1 opus
            C[I]:=I .
finis
```

Operaciones tipo básicos

Numerus

incrementum y *

decrementum es un operador binario prefijo que disminuye en 1 el valor de la variable. * representa la multiplicación usual.

Ejemplo:

```
commutabilis
      numerus A est 0, B est 1.
firmamentum
      tempus decrementum A > 0 opus
      initum
            scribonumerus (A) .
            B := B * A .
            scribonumerus (B) .
      finis
```

Gramática

1. <S>::=nomen NombrePrograma <Perpetus> <Furibundus> <Commutabilis> <Exemplar> <Corpus> firmamentum <ListaSentencia>
2. <Perpetus>::=perpetus <Constantes>
3. <Perpetus>::=
4. <Constantes>::=<Tipo> IDENTIFICADOR dito <Literal> <SigCons> . <Constantes>
5. <Constantes>::=
6. <SigCons>::=, IDENTIFICADOR dito <Literal>
7. <SigCons>::=
8. <Furibundus>::=furibundus <Tipos>
9. <Furibundus>::=
10. <Tipos>::=IDENTIFICADOR sum <TiposAux> . <Tipos>
11. <Tipos>::=
12. <TiposAux> ::= <Tipo>
13. <TipoAux> ::= coniugo <Tipo> IDENTIFICADOR . <Attrs> dixi
14. <Commutabilis>::=commutabilis <Variables>
15. <Commutabilis>::=
16. <Variables>::=<Tipo> IDENTIFICADOR <Var_asignacion> <SigVar> . <Variables>
17. <Variables>::=coniugo <Tipo> IDENTIFICADOR . <Attrs> dixi IDENTIFICADOR <Var_asignacion> <SigVar> . <Variables>
18. <Variables>::=ordo [numerus_lit] <SigTam> autem <Tipo> IDENTIFICADOR <Var_asignacion> . <Variables>
19. <Variables>::=
20. <SigVar>::=, IDENTIFICADOR <Var_asignacion> <SigVar>
21. <SigVar>::=
22. <Var_asignacion>::=est <Literal>
23. <Var_asignacion>::=
24. <Attrs>::=<Tipo> IDENTIFICADOR . <Attrs>
25. <Attrs>::=
26. <SigTam>::=[numerus_lit] <SigTam>
27. <SigTam>::=
28. <Exemplar>::=exemplar <Prototipo>
29. <Exemplar>::=
30. <Prototipo>::=efficio IDENTIFICADOR (<Param>) : <Tipo> .
31. <Prototipo>::=directus IDENTIFICADOR (<Param>) . <Prototipo>
32. <Prototipo>::=
33. <Param>::=<Modificador> <Tipo> IDENTIFICADOR <SiguienteID> <SigParam>
34. <Param>::=
35. <SigParam>::=. <Modificador> <Tipo> IDENTIFICADOR <SiguienteID> <SigParam>
36. <SigParam>::=
37. <SiguienteID>::=, IDENTIFICADOR <SiguienteID>
38. <SiguienteID>::=
39. <Modificador>::=ign
40. <Modificador>::=
41. <Corpus>::=corpus <Rutinas>
42. <Corpus>::=
43. <Rutinas>::=efficio IDENTIFICADOR (<Param>) : <Tipo> . <Commutabilis> initum <ListaSentencia> finis <Rutinas>
44. <Rutinas>::=directus IDENTIFICADOR (<Param>) . <Commutabilis> initum <ListaSentencia> finis <Rutinas>
45. <Rutinas>::=
46. <Tipo>::=numerus
47. <Tipo>::=imago
48. <Tipo>::=catena
49. <Tipo>::=dualis

```
50.  <Tipo>::=liber
51.  <Tipo>::=fractio
52.  <Tipo>::=gregorius
53.  <Tipo>::=IDENTIFICADOR
54.  <Literal>::=numerus_lit
55.  <Literal>::=imago_lit
56.  <Literal>::=catena_lit
57.  <Literal>::=dualis_lit
58.  <Literal>::=fractio_lit
59.  <Literal>::=<Literal_reg>
60.  <Literal>::=<Literal_arr>
61.  <Literal>::=gregorius_lit
62.  <Literal_reg>::=<- <Literal> <LiteralesComa> ->
63.  <Literal_arr>::={ <Literal> <LiteralesComa> }
64.  <LiteralesComa>::=, <Literal> <LiteralesComa>
65.  <LiteralesComa>::=
66.  <ParteAsignacion>::=IDENTIFICADOR <ParteAsignacionAux>
67.  <ParteAsignacionAux>::=
68.  <ParteAsignacionAux>::=@ IDENTIFICADOR <ParteAsignacionAux>
69.  <ParteAsignacionAux>::=[ <ExpresionTernaria> ] <ParteAsignacionAux>
70.  <ExpresionPostfija>::=IDENTIFICADOR <Invocacion>
71.  <ExpresionPostfija>::=( <ExpresionImpono> )
72.  <ExpresionPostfija>::=<Literal>
73.  <Invocacion>::=<ParteAsignacionAux>
74.  <Invocacion>::=( <ListaArgumentos> ) <ParteAsignacionAux>
75.  <Invocacion>::=@@ <ExpresionPostfija>
76.  <ListaArgumentos>::=<ExpresionImpono> <ListaArgumentosAux>
77.  <ListaArgumentos>::=
78.  <ListaArgumentosAux>::=, <ExpresionImpono> <ListaArgumentosAux>
79.  <ListaArgumentosAux>::=
80.  <ExpresionUnaria>::=<ExpresionPostfija>
81.  <ExpresionUnaria>::=<OperadorUnario> <ExpresionUnaria>
82.  <OperadorUnario>::=incrementum
83.  <OperadorUnario>::=decrementum
84.  <OperadorUnario>::=non
85.  <OperadorUnario>::=quantus
86.  <OperadorUnario>::=[>>]
87.  <OperadorUnario>::=[<<]
88.  <OperadorUnario>::=[&?]
89.  <OperadorUnario>::=[#?]
90.  <OperadorUnario>::=&#
91.  <OperadorUnario>::=||
92.  <OperadorUnario>::=|^
93.  <OperadorUnario>::=annus
94.  <OperadorUnario>::=mensis
95.  <OperadorUnario>::=dies
96.  <OperadorUnario>::=nunc
97.  <ExpresionLectura>::=<ExpresionUnaria>
98.  <ExpresionLectura>::=lectio <ExpresionLectura>
99.  <ExpresionLectura>::=prope <ExpresionLectura>
100. <ExpresionString>::=<ExpresionLectura> <ExpresionStringAux>
101. <ExpresionStringAux>::=<OperadorString> <ExpresionString>
102. <ExpresionStringAux>::=
103. <OperadorString>::=&+
104. <OperadorString>::=&?
105. <ExpresionMultiplicativa>::=<ExpresionString> <ExpresionMultiplicativaAux>
106. <ExpresionMultiplicativaAux>::=<OperadorMultiplicativo> <ExpresionMultiplicativa>
```

```
<ExpresionMultiplicativaAux>::=
<OperadorMultiplicativo>::=*
<OperadorMultiplicativo>::=/
<OperadorMultiplicativo>::=%
<OperadorMultiplicativo>::=|*
<OperadorMultiplicativo>::=|/
<ExpresionAditiva>::=<ExpresionMultiplicativa> <ExpresionAditivaAux>
<ExpresionAditivaAux>::=<OperadorAditivo> <ExpresionAditiva>
<ExpresionAditivaAux>::=
<OperadorAditivo>::=+
<OperadorAditivo>::=-
<OperadorAditivo>::=|+
<OperadorAditivo>::=|-
<ExpresionRelacional>::=<ExpresionAditiva> <ExpresionRelacionalAux>
<ExpresionRelacionalAux>::=<OperadorRelacional> <ExpresionRelacional>
<ExpresionRelacionalAux>::=
<OperadorRelacional>::=<
<OperadorRelacional>::=>
<OperadorRelacional>::=>=
<OperadorRelacional>::=<=
<ExpresionIgualdad>::=<ExpresionRelacional> <ExpresionIgualdadAux>
<ExpresionIgualdadAux>::=<OperadorIgualdad> <ExpresionIgualdad>
<ExpresionIgualdadAux>::=
<OperadorIgualdad>::==
<OperadorIgualdad>::=><
<ExpresionAndLogica>::=<ExpresionIgualdad> <ExpresionAndLogicaAux>
<ExpresionAndLogicaAux>::=et <ExpresionAndLogica>
<ExpresionAndLogicaAux>::=
<ExpresionXorLogica>::=<ExpresionAndLogica> <ExpresionXorLogicaAux>
<ExpresionXorLogicaAux>::=xaut <ExpresionXorLogica>
<ExpresionXorLogicaAux>::=
<ExpresionOLogica>::=<ExpresionXorLogica> <ExpresionOLogicaAux>
<ExpresionOLogicaAux>::=aut <ExpresionOLogica>
<ExpresionOLogicaAux>::=
<ExpresionArchivo>::=<ExpresionOLogica> <ExpresionArchivoAux>
<ExpresionArchivoAux>::=patentibus <ModoArchivo> <OperadorArchivo> <ExpresionArchivo>
<ExpresionArchivoAux>::=<OperadorArchivo> <ExpresionArchivo>
<ExpresionArchivoAux>::=
<OperadorArchivo>::=ligo
<OperadorArchivo>::=scripturam
<ExpresionFecha>::=<ExpresionArchivo> <ExpresionFechaAux>
<ExpresionFechaAux>::=<OperadorFecha> <ExpresionFecha>
<ExpresionFechaAux>::=
<OperadorFecha>::=$annus
<OperadorFecha>::=$mensis
<OperadorFecha>::=$dies
<ModoArchivo>::=rescribo
<ModoArchivo>::=lego
<ModoArchivo>::=scribo
<ExpresionTernaria>::=<ExpresionFecha> <ExpresionTernariaAux>
<ExpresionTernariaAux>::=& <Expresion> <OperadorTernario> <ExpresionTernaria>
<ExpresionTernariaAux>::=
<OperadorTernario>::=|>
<OperadorTernario>::=|<
<ExpresionImpono>::=<ExpresionTernaria> <ExpresionImponoAux>
<ExpresionImponoAux>::=impono <Tipo>
<ExpresionImponoAux>::=
```

164. <ExpresionAsignacion>::=<ParteAsignacion> <OperadorAsignacion> <ExpresionAsignacion>
165. <ExpresionAsignacion>::=<ExpresionImpono>
166. <OperadorAsignacion>::=+=
167. <OperadorAsignacion>::=*=
168. <OperadorAsignacion>::=-=
169. <OperadorAsignacion>::=/=
170. <OperadorAsignacion>::=%=
171. <OperadorAsignacion>::=|+=
172. <OperadorAsignacion>::=|-=
173. <OperadorAsignacion>::=||=
174. <OperadorAsignacion>::=|^=
175. <OperadorAsignacion>::=|/=
176. <OperadorAsignacion>::=|*=
177. <OperadorAsignacion>::=et=
178. <OperadorAsignacion>::=xaut=
179. <OperadorAsignacion>::=aut=
180. <OperadorAsignacion>::=:=
181. <OperadorAsignacion>::=$annus=
182. <OperadorAsignacion>::=$mensis=
183. <OperadorAsignacion>::=$dies=
184. <Expresion>::=<ExpresionAsignacion> .
185. <Sentencia>::=<Expresion>
186. <Sentencia>::=<SentenciaSeleccion>
187. <Sentencia>::=<SentenciaIteracion>
188. <Sentencia>::=<SentenciaFlujo>
189. <Sentencia>::=<SentenciaRegistro>
190. <Sentencia>::=<SentenciaTry>
191. <ListaSentencia>::=<Sentencia> <ListaSentencia>
192. <ListaSentencia>::=
193. <BloqueSentencia>::=initum <ListaSentencia> finis
194. <BloqueSentencia>::=<Sentencia>
195. <SentenciaIteracion>::=panis IDENTIFICADOR := <ExpresionTernaria> auctum <ExpresionTernaria> gradus <ExpresionTernaria> opus <BloqueSentencia>
196. <SentenciaIteracion>::=tempus <ExpresionTernaria> opus <BloqueSentencia>
197. <SentenciaIteracion>::=itero <BloqueSentencia> usque <ExpresionTernaria>
198. <SentenciaIteracion>::=decusis <ExpresionTernaria> cum <ExpresionTernaria> per IDENTIFICADOR opus <BloqueSentencia>
199. <SentenciaFlujo>::=neco
200. <SentenciaFlujo>::=pergo
201. <SentenciaFlujo>::=claudeo
202. <SentenciaFlujo>::=reditus <ExpresionTernaria>
203. <SentenciaFlujo>::=revello
204. <SentenciaSeleccion>::=in <ExpresionTernaria> <Then> .
205. <SentenciaSeleccion>::=aeger <ExpresionTernaria> initum casus opus <Literal> <ListaSentencia> <Casos> <Default> finis
206. <Then>::=<Verdadero>
207. <Then>::=<Falso>
208. <Verdadero>::=certus <BloqueSentencia> <ElseVerdadero>
209. <ElseVerdadero>::=mentiri <BloqueSentencia>
210. <ElseVerdadero>::=
211. <Falso>::=mentiri <BloqueSentencia> <ElseFalso>
212. <ElseFalso>::=certus <BloqueSentencia>
213. <ElseFalso>::=
214. <Casos>::=casus opus <Literal> <ListaSentencia> <Casos>
215. <Casos>::=
216. <Default>::=detrimentum <ListaSentencia>
217. <SentenciaRegistro>::=sigla IDENTIFICADOR opus <ListaSentencia> dixi
218. <SentenciaTry>::=probare <BloqueSentencia> captare <BloqueSentencia>

Lista de códigos para las familias y sus respectivos colores en el pretty printer

Terminal	Cod	Color
-	0	#8d34ca
$-	1	#00b050
$?	2	#00b050
$+	3	#00b050
$annus	4	#00b050
$annus=	5	#00b050
$d	6	#00b050
$dies	7	#00b050
$dies=	8	#00b050
$mensis	9	#00b050
$mensis=	10	#00b050
$n	11	#00b050
%	12	#8d34ca
%=	13	#8d34ca
&	14	#ffa021
&#	15	#ffa021
&?	16	#ffa021
&+	17	#ffa021
(	18	#bc70b1
)	19	#bc70b1
*	20	#8d34ca
*=	21	#8d34ca
,	22	#c45911
.	23	#bc70b1
/	24	#8d34ca
/=	25	#8d34ca
:	26	#bc70b1
:=	27	#bc70b1
@	28	#bc70b1
@@	29	#bc70b1
[	30	#bc70b1
[#?]	31	#006c00
[&?]	32	#006c00
[<<]	33	#006c00

[>>]	34	#006c00
]	35	#bc70b1
{	36	#bc70b1
|-	37	#a84679
|*	38	#a84679
|*=	39	#a84679
|/	40	#a84679
|/=	41	#a84679
|^	42	#a84679
|^=	43	#a84679
||	44	#a84679
||=	45	#a84679
|+	46	#a84679
|+=	47	#a84679
|<	48	#ffa021
|-=	49	#a84679
|>	50	#ffa021
}	51	#bc70b1
+	52	#8d34ca
+=	53	#8d34ca
<	54	#354259
<-	55	#bc70b1
<=	56	#354259
=	57	#354259
-=	58	#8d34ca
>	59	#354259
->	60	#bc70b1
><	61	#354259
>=	62	#354259
aeger	63	#3329f3
annus	64	#00b050
auctum	65	#3329f3
aut	66	#354259
aut=	67	#354259
autem	68	#3329f3
captare	69	#3329f3
casus	70	#3329f3
catena	71	#a80000

catena_lit	72	#dd4f8f
certus	73	#3329f3
claudeo	74	#3329f3
commutabilis	75	#3b3838
coniugo	76	#a80000
corpus	77	#3b3838
cum	78	#3329f3
decrementum	79	#8d34ca
decusis	80	#3329f3
detrimentum	81	#3329f3
dies	82	#00b050
directus	83	#3329f3
dito	84	#958445
dixi	85	#c75151
dualis	86	#a80000
dualis_lit	87	#6a283c
efficio	88	#3329f3
est	89	#958445
et	90	#354259
et=	91	#354259
exemplar	92	#3b3838
finis	93	#c75151
firmamentum	94	#3b3838
fractio	95	#a80000
fractio_lit	96	#0063da
furibundus	97	#3b3838
gradus	98	#3329f3
gregorius	99	#a80000
gregorius_lit	100	#39ffac
IDENTIFICADOR	101	#000000
ign	102	#3329f3
imago	103	#a80000
imago_lit	104	#70ad47
impono	105	#3329f3
in	106	#3329f3
incrementum	107	#8d34ca
initum	108	#c75151
itero	109	#3329f3

lectio	110	#eac272
lego	111	#b2d000
liber	112	#a80000
ligo	113	#eac272
mensis	114	#00b050
mentiri	115	#00b050
neco	116	#3329f3
nomen	117	#3b3838
non	118	#354259
numerus	119	#a80000
numerus_lit	120	#00ff00
nunc	121	#00b050
opus	122	#c75151
ordo	123	#a80000
panis	124	#3329f3
patentibus	125	#eac272
per	126	#3329f3
pergo	127	#3329f3
perpetus	128	#3b3838
probare	129	#3329f3
prope	130	#eac272
quantus	131	#3329f3
reditus	132	#3329f3
rescribo	133	#b2d000
revello	134	#3329f3
scribo	135	#b2d000
scripturam	136	#eac272
sigla	137	#3329f3
sum	138	#958445
tempus	139	#3329f3
usque	140	#3329f3
xaut	141	#354259
xaut=	142	#354259
EOF	143	

Estándar para errores

El error tiene el sguiente formato: **SIGLA-##:** *mensaje de error.* Donde SIGLA identifica el error perteneciente a la etapa del compilador.

Generales

La sigla de este tipo de error es GN. Estos errores se relacionan con el procesamiento del archivo fuente del programa.

GN-0	El archivo no existe
GN-1	Se esperaba la extensión str
GN-2	El archivo no se pudo abrir
GN-3	Ha ocurrido un error al leer la siguiente porción del archivo
GN-4	Ha ocurrido un error al cerrar el buffer
GN-5	El buffer ya estaba cerrado
GN-6	Debe cerrar el escáner para volver a inicializarlo
GN-7	Debe iniciar el escáner antes de usarlo

Léxicos

La sigla de este tipo de error es LX. Estos errores se la relacionan con el procesamiento de

LX-1	Caracter inesperado
LX-2	Literal de numerus mal formada en la linea x columna y
LX-3	Caracter de escape inválido en la linea x columna y
LX-4	Literal de catena mal formada en la linea x columna y
LX-5	Literal de imago mal formada en la linea x columna y
LX-6	Literal de fractio mal formada en la linea x columna y
LX-7	Literal de gregorius mal formada en la linea x columna y
LX-8	Identificador mal formado

Sintácticos

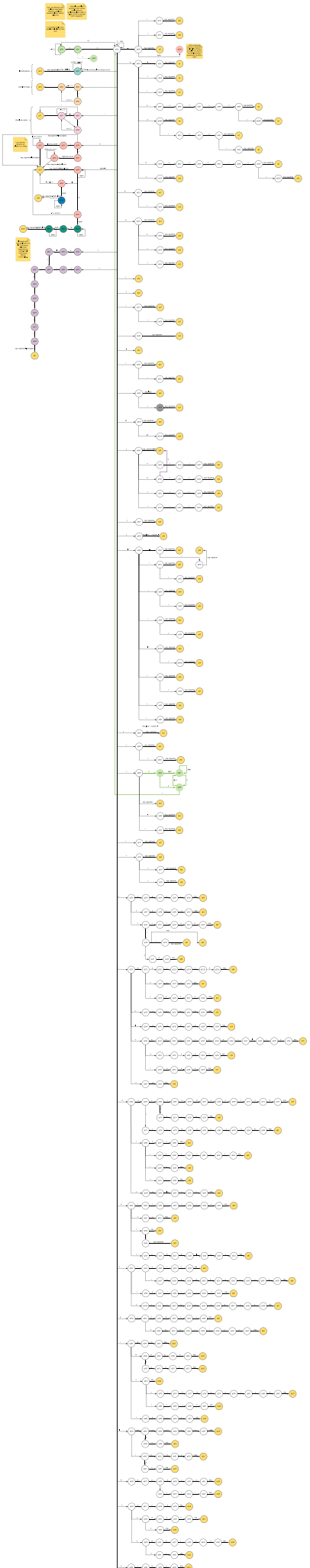
La sigla es ST.

Semánticos

La sigla es SM

Automata

(Se adjunta como SVG)



Estándar para el nombre de las pruebas

Las pruebas se realizarán mediante un número consecutivo. El nombre se mantendrá a siete un caracteres como *test###*. Se conserva tres caracteres para el consecutivo.

Índice de pruebas

Nombre	Descripción
test000	Prueba para probar la generación de expresiones aritméticas
test001	Prueba de fibonacci
test002	Prueba de un arreglo de tipo registro
test003	Prueba de un los lados válidos de un triángulo. Usa la instrucción creativa probare.
test004	Prueba del cálculo del área de un círculo
test005	Prueba sobre el cálculo de la edad según fecha de nacimiento. Uso de tipo creativo gregorius
test006	Prueba sobre el uso de la instrucción creativa intersección repetitiva. La suma de los cuadrados.
test007	Prueba sobre las conversiones de tipos entre imago, catena, fractio, dualis y numerus.
test008	Prueba sobre el uso de impono para saltar de un tipo a otro y asignarlo a un tercero.
test009	Prueba de uso de parámetros por referencia. Se programa un generador de números pseudo aleatorios usando el algoritmo de *Linear congruential method.*
test010	Prueba de todos los terminales del lenguaje. Pruebas de identación
test011	Prueba de literales
test012	Pruebas de comentarios
test013	Comentarios sin cerrar
test014	Caracter monstruo