ANALIZADOR SINTÁCTICO TINY(0)

Óscar Morujo Fernández

Sofía Capmany Fernández

G16

ESPECIFICACIÓN SINTÁCTICA

No terminales :

PROGRAMA, LDECS, DEC, NOMBRE_TIPO, LINST, INST, E0, E1, E2, E3, E4, E5, OP3NA, OP2AI, OP1AI.

Terminales :

sep_di	“&&”	**bne**	“<=”
semicolon	“;”	**beq**	“==”
id	identificador	**ble**	“<=”
r_real	“real”	**bge**	“>=”
r_int	“int”	**bgt**	“>”
r_bool	“bool”	**blt**	“<”
igual	“=”	**true**	“true”
mas	“+”	**false**	“false”
menos	“-”	**pcierre**	“)”
por	“*”	**pap**	“(“
div	“/”	**real**	número real
and	“and”	**ent**	número entero
or	“or”	**not**	“not”

Gramática:

PROGRAMA := LDECS **sep_di** LINST;
LDECS := LDECS **semicolon** DEC;
LDECS := DEC;
DEC := NOMBRE_TIPO **id**;
NOMBRE_TIPO := **r_real**;
NOMBRE_TIPO :=**r_int;**
NOMBRE_TIPO :=**r_bool;**
LINST := LINST **semicolon** INST;
LINST := INST;
INST := **id igual** E0;

E0 := E1 **mas** E0;
E0 := E1 **menos** E1;
E0 := E1;

E1 := E1 OP1AI E2;
E1 := E2;

E2 := E2 OP2AI E3;
E2 := E3;

E3 := E4 OP3NA E4;
E3 := E4;

E4 := **menos** E5;
E4 := **not** E4;
E4 := E5;

E5 := **ent**;
E5 := **real**;
E5 := **id**;
E5 := **true**;
E5 := **false**;
E5 := **pap** E0 **pcierre**;

OP3NA := **por**;
OP3NA := **div**;

OP2AI := **bne**;
OP2AI := **beq**;
OP2AI := **ble**;
OP2AI := **bge**;
OP2AI := **blt**;
OP2AI :=**bgt**;

OP1AI := **and**;
OP1AI := **or**;

ACONDICIONAMIENTO DE LA GRAMÁTICA (LL1)

Se añaden los siguientes **no terminales** al conjunto anterior : RLDECS, RLISNT, RES0, RES1, RES2 y RES3

Gramática LL1:

PROGRAMA := LDECS **sep_di** LINST;
LDECS := DEC RLDECS;
RLDECS := **semicolon** DEC RLDECS;
RLDECS := **ƛ**;
DEC := NOMBRE_TIPO **id**;
NOMBRE_TIPO := **r_real**;
NOMBRE_TIPO :=**r_int;**
NOMBRE_TIPO :=**r_bool;**
LINST := INST RLINST;
RLINST := **semicolon** INST RLINST;
RLINST := **ƛ**;
INST := **id igual** E0;

E0 := E1 RES0;
RES0 := **mas** E0;
RES0 := **menos** E1;
RES0 := **ƛ**;

E1 := E2 RES1;
RES1 := OP1AI E2 RES1;
RES1 := **ƛ**;

E2 := E3 RES2;
RES2 := OP2AI E3 RES2;
RES2 := **ƛ**;

E3 := E4 RES3;

RES3 := OP3NA E4;
RES3 := **ƛ**;

E4 := **menos** E5;
E4 := **not** E4;
E4 := E5;

E5 := **ent**;
E5 := **real**;
E5 := **id**;
E5 := **true**;
E5 := **false**;
E5 := **pap** E0 **pcierre**;

OP3NA := **por**;
OP3NA := **div**;

OP2AI := **bne**;
OP2AI := **beq**;
OP2AI := **ble**;
OP2AI := **bge**;
OP2AI := **blt**;
OP2AI :=**bgt**;

OP1AI := **and**;
OP1AI := **or**;

DIRECTORES DE LA GRAMÁTICA

A continuación se muestran los símbolos anulables y los primeros y los siguientes de cada no terminal

Anulables
RES2 RES3 RES0 RLDECS RES1 RLINST

PRIM(α)

No terminal	Iniciales
LINST	id
DEC	r_int r_real r_bool
INST	id
OP1AI	or and
PROGRAMA	r_int r_real r_bool
OP2AI	bge bne ble blt bgt beq
E0	not menos true false ent real id pap
E1	not menos true false ent real id pap
E2	not menos true false ent real id pap
E3	not menos true false ent real id pap
E4	not menos true false ent real id pap
NOMBRE_TIPO	r_int r_real r_bool
E5	true false ent real id pap
LDECS	r_int r_real r_bool
RES2	bge bne ble blt bgt beq
RES3	div por
RES0	menos mas
RLDECS	semicolon
RES1	or and
OP3NA	div por
RLINST	semicolon

SIG(α)

No terminal	Seguidores
LINST	$
DEC	sep_di semicolon
INST	$ semicolon
OP1AI	not menos true false ent real id pap
PROGRAMA	$
OP2AI	not menos true false ent real id pap
E0	$ pcierre semicolon
E1	$ menos pcierre mas semicolon
E2	or $ menos and pcierre mas semicolon
E3	bge or $ blt bgt beq menos and pcierre bne ble mas semicolon
E4	bge or $ blt bgt beq div por menos and pcierre bne ble mas semicolon
NOMBRE_TIPO	id
E5	bge or $ blt bgt beq div por menos and pcierre bne ble mas semicolon
LDECS	sep_di
RES2	or $ menos and pcierre mas semicolon
RES3	bge or $ blt bgt beq menos and pcierre bne ble mas semicolon
RES0	$ pcierre semicolon
RLDECS	sep_di
RES1	$ menos pcierre mas semicolon
OP3NA	not menos true false ent real id pap
RLINST	$

Una vez tenemos los primeros y siguientes de los no terminales, procedemos a ver cuales son los **directores** de cada regla en nuestra gramática:

REGLA	DIRECTORES
PROGRAMA := LDECs **sep_di** LINST	{r_bool,r_int,r_real}
LDECs := DEC RLDECs	{r_bool,r_int,r_real}
RLDECS := **semicolon** DEC RLDECS	{semicolon}
RLDECS := ƛ	{sep_di}

DEC := NOMBRE_TIPO **id**	{r_bool,r_int,r_real}
NOMBRE_TIPO := **r_real**	{r_real}
NOMBRE_TIPO := **r_int**	{r_int}
NOMBRE_TIPO := **r_bool**	{r_bool}
LINST := INST RLINST	{id}
RLINST := **semicolon** INST RLINST	{semicolon}
RLINST := **ƛ**	{eof}
INST := **id igual** E0	{id}
E0 := E1 RES0	{ent,false,id,menos,not,pap,real,true}
RES0 := **mas** E0	{mas}
RES0 := **menos** E1	{menos}
RES0 := **ƛ**	{pcierre,semicolon,eof}
E1 := E2 RES1	{ent,false,id,menos,not,pap,real,true}
RES1 := OP1AI E2 RES1	{and,or}
RES1 := **ƛ**	{mas,menos,pcierre,semicolon,eof}
E2 := E3 RES2	{ent,false,id,menos,not,pap,real,true}
RES2 := OP2AI E3 RES2	{beq,bne,blt,ble,bge,bgt}
RES2 := **ƛ**	{and,or,mas,menos,pcierre,semicolon,eof}
E3 := E4 RES3	{ent,false,id,menos,not,pap,real,true}
RES3 := OP3NA E4	{div,por}
RES3 := **ƛ**	{and,beq,bge,bgt,ble,bne,blt,mas,menos,or,pcierre,semicolon,eof}
E4 := **menos** E5	{menos}
E4 := **not** E4	{not}
E4 := E5	{ent,false,id,pap,real,true}
E5 := **ent** | **real** | **id** | **false** | **true** |**pap** E0 **pcierre**	{ent,real,id,pap,false,true}
OP3NA := **por** | **div**	{por,div}

OP2AI := **bne** | **beq** | **ble** | **bge** | **blt** | **bgt**	{beq,bne,blt,ble,bge,bgt}
OP1AI := **and** | **or**	{and,or}

SÍMBOLOS PARA EL DIAGNÓSTICO DE ERRORES

Los directores primeros se incluyen en la información de diagnóstico ya que podemos asegurar que son símbolos esperados.Sin embargo habrá que estudiar los que son siguientes para ver qué símbolos se deben de incluir en la información para no confundir al usuario con símbolos que no corresponden.

REGLA	DIRECTORES
PROGRAMA: todos los directores son símbolos primeros.	{r_bool,r_int,r_real}
LDECS:todos los directores son símbolos primeros.	{r_bool,r_int,r_real}
RLDECS: siempre que tenemos un resto de lista de declaraciones podemos esperar un “;” que es un símbolo primero de RLDECS, y también “&&” que es un siguiente.	{semicolon,sep_di}
DEC : todos los directores son símbolos primeros.	{r_bool,r_int,r_real}
NOMBRE_TIPO : todos los directores son símbolos primeros.	{r_bool,r_int,r_real}
LINST : todos los directores son símbolos primeros.	{id}
RLINST :el semicolon es un director primero que siempre podemos esperar, pero también podemos esperar un siguiente que es el eof, ya que el código puede terminar con la última instrucción.	{semicolon,eof}
INST :todos los directores son símbolos primeros.	{id}
E0 : los directores primeros siempre se	{ent,false,id,menos,not,pap,real,true}

pueden esperar, pero no podremos esperar “(“ ni “;” que son directores siguientes.	
RES0 : “+” y “-” siempre los podemos esperar, pero de los directores que son símbolos siguientes no podremos esperar “)“ , “;” ni “eof”.	{mas,menos}
E1: todos los directores son símbolos primeros.	{ent,false,id,menos,not,pap,real,true}
RES1 : podremos esperar los directores primeros “and” y “or”, además de los directores siguientes “+” y “-”. No podremos esperar ni “)” ni “;” ni “eof”.	{and,or,mas,menos}
E2 : todos los directores son símbolos primeros.	{ent,false,id,menos,not,pap,real,true}
RES2 : podremos esperar los operadores relacionales que son directores primeros, además de “and”, “or”, “+” y “-” que son directores siguientes. No podremos esperar ni “)” ni “;” ni “eof”.	{beq,bne,blt,ble,bge,bgt,and,or,mas,menos}
E3: todos los directores son símbolos primeros.	{ent,false,id,menos,not,pap,real,true}
RES3 : podremos esperar los primeros “/” y “*”, y los siguientes “and”,”or””,”+”,”-” y operadores relacionales. No podremos esperar ni “)” ni “;” ni “eof”.	{div,por,and,beq,bge,bgt,ble,bne,blt,mas,menos,or}
E4: todos los directores son símbolos primeros.	{ent,false,id,menos,not,pap,real,true}
E5: todos los directores son símbolos primeros.	{ent,real,id,pap,false,true}
OP3NA : todos los directores son símbolos primeros.	{por,div}
OP2AI : todos los directores son símbolos primeros.	{beq,bne,blt,ble,bge,bgt}
OP1AI : todos los directores son símbolos primeros.	{and,or}