MAYO, 2022

UNICITY CHECK
CONTROL DE UNICIDAD

UNIVERSIDAD
Panamericana

Scope

- GLOBAL
  Valid variables throughout the program.

- LOCAL
  Valid variables in the body of procedures or functions.

- Restriction
  Local variables overlapping global ones.

<constantes>::= constantes <GpoConst>

<GpoConst>::= id:=<cte>

constantes PI:=3.1416;
MAX:=30;
PI:="hola";

→|PI|C|R|0|0|…

→|MAX|C|E|0|0|…

Verify that it hasn't been declared before.

Remember: the type of variable and the data type must be **declare**, as it is undefined it can be updated.

→|W|V|R|0|0|…
w,
m,
w[max][max],
h: real;

→|W|V|R|30|30|…
→|PI|C|R|0|0|…
→|H|V|R|0|0|…
→|MAX|C|E|0|0|…
→|X|V|E|0|0|…

Error in w's

variables x,pi : entero;
w,m,w [MAX][MAX],h : real ;

You can create a **list** and put the data type from beginning to end, but a **queue** that insert the types from the last.

```
funcion FacRec(n:entero):entero;
…
procedimiento selecciona{
…
        FacRec(n);
}
```

Guardar la línea donde se encuentra en la tabla de símbolos

Si se utiliza y no se declara, es error/warning.
Si se declara y no hay prototipo, no es error

→|FACREC|F|E|0|0|…|V|

funcion FacRec(n:entero):entero;
funcion FacRec():real; error
funcion FacRec(m:entero):real error
inicio

Especificar que hace falta la declaración

El número de parámetros no es unicidad
→|N|I|I|0|0|…| |
↳|F|E|0|0|FACREC| |…

<ProtFunc>::=funcion id([<Params>]):<Tipo>;[<ProtFunc>]

unicidad

<funcion>::=funcion id ([<Params>]) :<Tipo> [<Variables>]
inicio [<Block>] fin de funcion;

Clase y Tipo iguales, si es F se declara, si es V es error

Detendremos la compilación a los 4 errores

```
variables i,x1,…:alfabetico;
…
funcion Pato():entero
variables
        i,j:entero;
```

ProcAct
x
y
PATO

→|I|V|A|0|0| |…
  ↳ |L|E|0|0| PATO|…

variables i,x,…:entero;

…

funcion Pato(i:real):logico

funcion Pollo(i:entero):real

Variable entera ya está declarada

Nombres relacionados

Al descargar pila, crear lista de tipos.

```
<Params>::=<GpoPars>:<tipo>[; <Params>]
<GpoPars>::=Id[, <GpoPars>]
```

Apilando Ids.

```
String parms
parms.at(i);
funcion hola(x,w,z:entero;r,s:real):logico
```

Revisar tamaño de la lista vs. # parámetros

```
log=hola(3,x,5,w/3,8.5,6.4);
```

Lista, no va en Tabla de Símbolos

```
<lproc>::=Id([<Uparams>])
<lfunc>::=Id([<Uparams>])
```

Verificar la cantidad de parámetros

```
<Uparams>::=<ExpLog>[,<Uparams>]
```

Contabilizar parámetros

Control de flujo

```
<regresa>::=regresa[(<ExpLog>)]
```

Verificar que se está en función o error.
Puede existir en un procedimiento pero no regresará nada.

```
continua  ← Interumpe ciclos, error fuera de ciclo.
            Verificar en ciclos.
```

```
<funcion>::=funcion Id([Params]):<tipo>[<variables>]
inicio[<Block>] fin de funcion;
```

Verificar que haya existido al menos un regresa.
Puede ser un regresa inalcanzable.

Verificación de tipos

E:=E→ λ
R:=E→ λ
R:=R→ λ
A:=A→ λ
L:=L→ λ

E+E→ E
E-E→ E
E*E→ E
E/E→ R
E%E→ E
E^E→ R
-E→ E

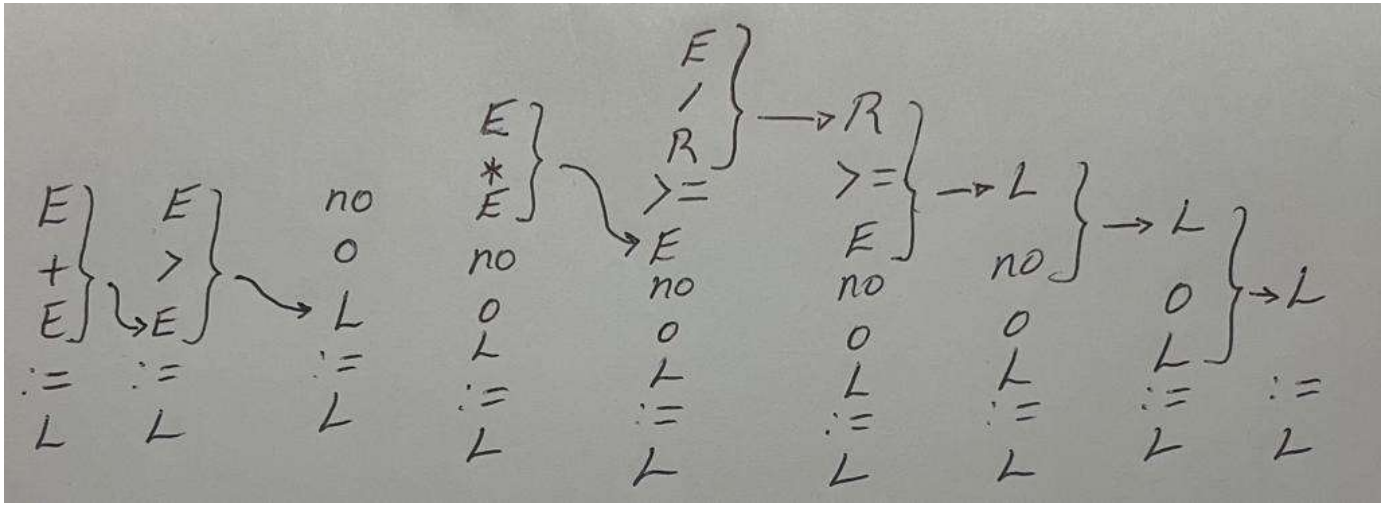
R+R→ R
R-R→ R
R+E→ R
R-E→ R
R * R→ R
R*E→ R
R/R→ R
R/E→ R
R^R→ R
R^E→ R
-R→ R

L o L→ L
L y L→ L
no L→ L

A+A→ A
A=A→ L
A<>A→ L

E OpRel E→ L
E OpRel R → L
R OpRel R→ L

```
LogN1:= x + 3 > 5 o no w*x >= m/2;
        E                 E      R
```



Observar al desapilar E>E.
Observar al desapilar E/R la operación correcta es R/E.
Observar al desapilar R>=E la operación correcta es E>=R.

LogN2:= x + 3 > 5 o no w*x;

R

~R?

no R → I

L o I → I

L:= I-λ

El chequeo de tipos en los estatutos se debe revisar al final de leer la <ExpLog> que el tipo sea L.

En el <Desde>

- En <Asigna>, E := E, sino error, se esperaba una asignación entera.
- En <Expr>, el tipo sobre la pila debe ser E.

En el <Cuando>

- Obtener y guardar el tipo del id.
- Verificar que el tipo del id sea igual al de la <Cte>

```
cuando el valor de i
inicio
        sea 1, 4.2, “hola”: m=i;
        sea falso, verdadero:m=i+8;
        otro: m=m;
Fin;
```

24. <regresa>::=regresa[(<Exprlog>)]

Tipo en la fila debe ser compatible con el tipo de función.

39.<Uparams>::=<Explog>[,<Uparams>]

Verificar con la lista de tipos en param de la función.

```
funcion xyz(x,z:real;y:entero):real
              R R       E
w=xyz(x,m+3,x/3)
         E
```

19. <asigna>::= Id[<Udim>]:=< Exprlog >

Verificar que no esté direccionado anteriormente.

23. <Udim>::=[<Expr>][<Udim>]

Verificar que el tipo sobre la pila sea E.

Verificar que id esté dimensionado contabilizar dimensiones.

m:= FacRec + Vector //Faltan parámetros , faltan dimensiones.
vector= FacRec; //Error
FacRec:=8; //Error
MAX :=105 //Error
LeeVec:=ImpVector; //Error
m:= vector(x); //Vector debe ser función

33.<termino>::=Id[|func|<Udim>]|(<Exprlog>)|CteEnt|CteReal|CteAlfa|verdadero|falso

35.<|func>::=Id ([<Uparams>])

Verificar que id sea función.

FacRec:=FacIter; //Error ID debe ser variable y faltan parámetros.
m:=FacRec[4]; //Error