

Capítol 4 Anàlisi Semàntica

- 4.1 Volem implementar les instruccions `break` i `continue` pel llenguatge LS. Per això, tindrem que controlar si se estan utilitzant dintre de bucle o no amb les corresponents accions semàntiques.
- 4.2 Fer la gramàtica i l'anàlisi semàntica de declaracions de variables de tipus `int` en el llenguatge C. Acceptem els modificadors d'array i apuntador. Exemple: `int a,*b,c[5]**r[3][5],(*p)[3];`
- 4.3 Afegir a LS la instrucció `switch` amb l'estructura de C (amb `case valor: break` i `default:`). Fer la corresponent modificació gramatical i afegir les corresponents accions semàntiques per controlar que el tipus de la expressió selectora sigui el mateix que el dels valors dels casos i que els valors dels casos no estiguin repetits.
- 4.4 Controlar quan el resultat d'una expressió en C te adreça o no. Fer les corresponents comprovacions semàntiques. Considerà els operadors aritmètics i `&`, `*`, `->`, `..`, `[]`, `()`, `=`.

Capítol 4 Anàlisi Semàntica

4.5 Implementar constants en LS i permetre que es posin expressions amb resultat constant com mida d'arrays. Exemple:

Const a=10; var b:array [a+3] of integer;

4.6 Afegir a les expressions de LS la declaració local Let in. Utilitzeu com a base el resultat del problema 4.5. Exemple:
d=let dx=x1-x0,dy=y1-y0 in dx*dx+dy*dy;

Solució Problema 4.1

- Exemples de codi correcte

```
procedure p(x:integer)
```

```
begin
```

```
  while x>0 do
```

```
  begin
```

```
    x=x-1;
```

```
    if x<10 then break;
```

```
  end
```

```
  for i=1 to 100 do
```

```
  begin
```

```
    if i==50 then break;
```

```
    if i==40 then continue;
```

```
  end
```

```
end
```

Solució Problema 4.1

- Exemples de codi erroni en vermell

```
procedure p(x:integer)
begin
  while x>0 do
  begin
    x=x-1;
    if x<10 then break;
    while x>0 do
    begin
      continue;
    end
    break;
  end
  continue;
end
```

```
procedure p(x:integer)
begin
  while x>0 do
  begin
    procedure falla()
    begin
      break;
    end
    x=x-1;
    if x<10 then break;
  end
end
```

Solució Problema 4.1

- Codi a modificar en la gramàtica de LS
 - Afegir als terminals break i continue
 - Nova variable global del compilador `Var PilaDeBucles=[];`
 - Afegir a instrucció

```
Rule <instruccio> ::= ...  
| while <Expressio> do  
  @PilaDeBucles="while"::PilaDeBucles;  
  <instruccio>  
  @PilaDeBucles=PilaDeBucles.Tail;  
| for identificador = <Expressio> to <Expressio> do  
  @PilaDeBucles="for"::PilaDeBucles;  
  <instruccio>  
  @PilaDeBucles=PilaDeBucles.Tail;  
| ...  
| break  
  @if (PilaDeBucles==[]) throw Exception("Break fora de bucle"); ";"  
| continue  
  @if (PilaDeBucles==[]) throw Exception("Continue fora de bucle"); ";"
```

Solució Problema 4.1

- Codi a modificar en la gramàtica de LS

– Afegir a bloc

Rule <bloc>::=

@Var AntPilaDeBucles=PilaDeBucles;

@PilaDeBucles=[];

"BEGIN" {

<DecFun>|

<DecProc>|

<DecVar>|

<DecTipus>|

<instruccio>

} "END"

@PilaDeBucles=AntPilaDeBucles;

Problema 4.2 Solució

CODI

```
int a;  
int * *p;  
int *str[5];  
int (*str2)[6];  
int *(str3[7]);  
int mat[3][4];
```

TAULA DE SIMBOLS

VARIABLE:

Nom => a
Posició => Unbound
Tipus => Integer

VARIABLE:

Nom => p
Posició => Unbound
Tipus => Apuntador
TipusBase => Apuntador
TipusBase => Integer

VARIABLE:

Nom => str
Posició => Unbound
Tipus => Array
Mida => 5
TipusElements => Apuntador
TipusBase => Integer

VARIABLE:

Nom => str2
Posició => Unbound
Tipus => Array
Mida => 6
TipusElements => Apuntador
TipusBase => Integer

VARIABLE:

Nom => str3
Posició => Unbound
Tipus => Apuntador
TipusBase => Array
Mida => 7
TipusElements => Integer

VARIABLE:

Nom => mat
Posició => Unbound
Tipus => Array
Mida => 3
TipusElements => Array
Mida => 4
TipusElements => Integer

Problema 4.2 Solució

BNF_PARSER <Decs>

TERMINALS * identificador numero () Int "[" "]" , ;

BNF

Rule <Decs> ::= {<Dec>}

Rule <Dec> ::= int <DecSimple> {, <DecSimple> } ";"

Rule <DecSimple> ::=

<DecApuntador> {"[" numero "]"}

Rule <DecApuntador> ::=

"*" <DecApuntador>

| identificador

| "(" <DecSimple> ")"

END

Problema 4.2 Solució

BNF_PARSER <Decs>

TERMINALS * identificador numero () Int "[" "]" , ;

BNF

Rule <Decs>::=

@TS=TaulaSimbols();

{<Dec>}

@TS.Imprimir();

Rule <Dec>::= @Var nom,tipus;

int <DecSimple(nom,TInt,tipus)>

@TS.ComprovarDuplicat(nom);

@TS.Insertar(ETSVariable(nom,tipus,Unbound));

{, <DecSimple(nom,TInt,tipus)>

@TS.ComprovarDuplicat(nom);

@TS.Insertar(ETSVariable(nom,tipus,Unbound));

} ";"

Problema 4.2 Solució

```
Rule <DecSimple(&nom,TBase,&tipus)>::=  
  <DecApuntador(nom,TBase,tipus)>  
  @var Dimensions=[];  
  { @var sz;  
    "[" numero#(sz) "]"  
    @if (!TypeP(Int,sz) || sz<=0)  
      throw Exception("Mida d'array errònia");  
    @Dimensions=sz::Dimensions;  
  }  
  @for (sz<-Dimensions) tipus=TipusArray(sz,tipus);
```

```
Rule <DecApuntador(&nom,TBase,&tipus)>::=  
  "*" @tipus=TipusApuntador(Unbound);  
  <DecApuntador(nom,TBase,tipus.TipusBase)>  
  | identificador#(nom) @tipus=TBase;  
  | "(" <DecSimple(nom,TBase,tipus)> ")"
```

Problema 4.3 Solució

Rule <instruccio> ::=

if <Expressio> then <instruccio>

[else <instruccio>]

| while <Expressio> do <instruccio>

| for identificador = <Expressio> to <Expressio> do <instruccio>

| "BEGIN"

{ <DecFun> | <DecProc> | <DecVar> | <instruccio> }

"END" |

| break ";"

| continue ";"

| switch <expressio> do

{ case <cte> : { <instruccio> } } [default : { <instruccio> }]

"end"

Rule <cte> ::= numero | - numero | caracter

Problema 4.3 Solució

Rule <instruccio(EnBucle,EnSwitch)>::=

if <Expressio> then <instruccio(EnBucle,EnSwitch)>

[else <instruccio(EnBucle,EnSwitch)>]

| while <Expressio> do <instruccio(true, EnSwitch)>

| for identificador = <Expressio> to <Expressio> do

<instruccio(true, EnSwitch)>

| "BEGIN"

{ <DecFun> | <DecProc> | <DecVar> |

<instruccio(EnBucle,EnSwitch)> }

"END"

| break

@if (!EnBucle && !EnSwitch) throw Exception("Break fora ...");

";"

| continue

@if (!EnBucle) throw Exception("Continue fora de bucle");

";"

Problema 4.3 Solució

```
@var tsel,tcte;
switch <expressio(tsel,true)>
  @if (tsel!:=TInt && tsel!:=TReal && tsel!:=TChar)
    throw Exception("Error en el tipus del selector");
  do
    @var ctes=[],v;
    {
      case <cte(tcte,v)> : {<instruccio(EnBucle,true)> }
      @if (tcte!:=tsel) throw Exception("Error en el tipus de la constant del case");
      @if (-)(x<-ctes,x==v)) throw Exception("constant repetida en casos del switch");
      @ctes=v::ctes;
    } [default : {<instruccio(EnBucle,true)>} ]
  "end"
```

```
Rule <cte(&t,&v)> ::=
  numero#(v) @t=if (TypeP(int,v)) TInt else TReal;
  | - numero#(v) @t=if (TypeP(int,v)) TInt else TReal; @ v=-v;
  | caracter #(v) @t=TChar;
```

Problema 4.4 Solució

Rule <programa> ::= { <expressio> ";" }

Rule <Expressio> ::= <ExpArit> [= <Expressio>]

Rule <ExpArit> ::= <terme> { (+ | -) <terme> }

Rule <terme> ::= <factor> { (* | /) <factor> }

Rule <factor> ::= - <factor> | * <factor> | & <factor> | <Operand>

Rule <Operand> ::=

Numero

| Identificador ("(" [<Expressio> { , <Expressio> }] ")" | <Post>)

| "(" <Expressio> ")" <Post>

Rule <Post> ::=

{

"[" <Expressio> "]" | . Identificador | -> identificador

}

Problema 4.4 Solució

```
Rule <programa> ::= @var res;  
    { <expressio(res)> ";" @cout.Println(res); }
```

```
Rule <Expressio(&Resultat)> ::=  
    <ExpArit(Resultat)>  
    [  
        @if (Resultat!=LValue) throw Exception("error en = per no ser LValue");  
        = <Expressio(Resultat)>  
        @Resultat=LValue;  
    ]
```

```
Rule <ExpArit(&Resultat)> ::=  
    <terme(Resultat)>  
    { ( + | - ) <terme(Resultat)> @Resultat=RValue; }
```

```
Rule <terme(&Resultat)> ::=  
    <factor(Resultat)>  
    { ( * | / ) <factor(Resultat)> @Resultat=RValue; }
```

Problema 4.4 Solució

Rule <factor(&Resultat)>::=

- <factor(Resultat)> @Resultat=RValue;

| * <factor(Resultat)> @Resultat=LValue;

| & <factor(Resultat)>

 @if (Resultat!=LValue)

 throw Exception("error en & per no ser LValue");

 @Resultat=RValue;

| <Operand(Resultat)>

Problema 4.4 Solució

Rule <Operand(&Resultat)>::=

Numero @Resultat=RValue;

| Identificador

(

@var r;

"(" [<Expressio(r)> {, <Expressio(r)> }] ")"

@Resultat=RValue;

|

@Resultat=LValue;

<Post(Resultat)>

)

| "(" <Expressio(Resultat)> ")" <Post(Resultat)>

Problema 4.4 Solució

Rule <Post(&Resultat)>::=

```
{
    @if (Resultat!=LValue)
        throw Exception("error en [ ] per no ser LValue");
    "[" <Expressio(Resultat)> "]"
    @Resultat=LValue;
    | . identificador
        @if (Resultat!=LValue)
            throw Exception("error en . camp per no ser LValue");
        @Resultat=LValue;
    | -> identificador
        @Resultat=LValue;
}
```

Problema 4.5 Solució

- Possibles resultats de les expressions

`Const ResultatValor, ResultatConstant, ResultatRes;`

- Entrada de la taula de símbols per guardar constants

`Type [public] ETSConstant(`

`Tipus, // tipus de la constant`

`Valor // Valor de la constant`

`) from EntradaTaulaSimbols`

`Constructor [public] ETSConstant(nom:String,t,val)`

`from EntradaTaulaSimbols(nom) => This(t,val)`

- Declaració de constant

Rule `<DecConst> ::= @var t,nom,valor;`

Const `identificador#(nom) = <expressio(t,ResultatConstant,Valor)> ";"`

`@TS.ComprovarDuplicat(nom);`

`@TS.Insertar(ETSConstant(nom,t,valor));`

Problema 4.5 Solució

- Llocs on afegir la declaració de constants

Rule <programa>::=

@TS=TaulaSimbols();

{ <DecFun> | <DecProc> | <DecConst> | <DecVar> | <DecTipus> }

@TS.Imprimir();

Rule <bloc>::= "BEGIN"

{ <DecFun> | <DecProc> | <DecVar> | <DecConst> |
<DecTipus> | <instruccio> }

"END"

Rule <instruccio>::= ... |

"BEGIN" @TS.NouAmbit();

{ <DecFun> | <DecProc> | <DecConst> | <DecVar> | <instruccio> }

@TS.EliminarAmbit();

"END" | ...

Problema 4.5 Solució

- Modificació de la crida a expressió

```
Rule <instruccio> ::= ... |  
<Expressio(t, ResultatRes, v)> ";" |  
if <Expressio(t, ResultatValor, v)> @if (t!=TInt) ...
```

- Expressió amb el tractament de constant

```
Rule <Expressio(&t, Resultat, &V)> ::= @var t2, V2;  
<TerBool(t, Resultat, v)> {  
    @if (Resultat==ResultatRes) throw Exception("error ... ");  
    || <TerBool(t2, Resultat, V2)>  
    @if (t!:=TInt || t2!:=TInt) throw Exception(  
        "Error de tipus de dades en || ", t, t2);  
    @if (Resultat==ResultatConstant) V=if (V!=0 || v2!=0) 1 else 0;  
}
```

Problema 4.5 Solució

```
Rule <FacBool(&t,Resultat,&V)>::= @var t2,op,v2;
  <ExpArit(t,Resultat,v)> [
    @if (Resultat==ResultatRes) throw Exception("error... ");
    ( == @op=\==;
      | != @op=\!=;
      | ">" @op=\>;
      | "<" @op=\<;
      | >= @op=\>=;
      | <= @op=\<=; )
    <ExpArit(t2,Resultat,v2)>
    @switch ((t,t2)) {
      (TInt,TInt) | (TInt,TReal) | (TReal,TInt) | (TReal,TReal) => {}
      Others => throw Exception(
        "Error de tipus de dades en comparacio ",t,t2);
    };
    @t=TInt;
    @if (Resultat==ResultatConstant) V=if (op(V,v2)) 1 else 0;
  ]
```

Problema 4.5 Solució

```
Rule <ExpArit(&t,Resultat,&V)>::=
  @var t2,v2;
  <terme(t,Resultat,v)> {
    @if (Resultat==ResultatRes) throw Exception("error... ");
    (
      + <terme(t2,Resultat,v2)>
      @switch ((t,t2)) {
        (TInt,TInt) => t=TInt;
        (TInt,TReal) => t=TReal;
        (TReal,TInt) => t=TReal;
        (TReal,TReal) => t=TReal;
        Others => throw Exception(
          "Error de tipus de dades en suma ",t,t2);
      };
      @if (Resultat==ResultatConstant) V=V+v2;
      | ... )
    }
  }
```

Problema 4.5 Solució

```
Rule <factor(&t,Resultat,&V)>::= @var nom,ets; (  
  ! <factor(t,Resultat,v)>  
    @if (t!=TInt) throw Exception( "Error de tipus de dades a !",t);  
    @if (Resultat==ResultatConstant) V=if (V==0) 1 else 0;  
    @if (Resultat==ResultatRes) throw Exception("error... ");  
  | - <factor(t,Resultat,v)>  
    @if (Resultat==ResultatConstant) V=-V  
    @if (Resultat==ResultatRes) throw Exception("error... ");  
    @switch (t) {  
      TInt | TReal => {}  
      Others => throw Exception("Error de tipus ... ",t);  
    };  
  | "(" <Expressio(t,Resultat,v)> ")"
```

Problema 4.5 Solució

| **Numero#(v)**

```
@t=if (typeP(int,v)) TInt else TReal;
```

```
@if (Resultat!=ResultatValor && Resultat!=ResultatConstant)  
    throw Exception("el resultat no pot ser un valor ",Resultat);
```

| **Caracter#(v)**

```
@t=TChar;
```

```
@if (Resultat!=ResultatValor && Resultat!=ResultatConstant)  
    throw Exception("el resultat no pot ser un valor");
```

| **String#(v)**

```
@t=TipusArray(v.Length,TChar);
```

```
@if (Resultat!=ResultatValor && Resultat!=ResultatConstant)  
    throw Exception("el resultat no pot ser un valor");
```

Problema 4.5 Solució

| **Identificador#(nom)**

```
@ets=TS.Buscar(nom);
```

```
(
```

```
  @{
```

```
    if (!TypeP(ETSFuncio,ets)) throw Exception(nom," no es funció");
```

```
    if (ets.TipusRetorn:=TVoid && Resultat!=ResultatRes)
```

```
        throw Exception("crida a procediment dintre d'expressió");
```

```
    if (ets.TipusRetorn!:=TVoid && Resultat!=ResultatValor)
```

```
        throw Exception("el resultat no pot ser un valor");
```

```
  }
```

```
  @var ltparams=ets.Parametres;
```

```
  "(" ... ")"
```

| ...

Problema 4.5 Solució

```
| Identificador #(nom) ( ... |
  @if (!TypeP(ETSVariable,ets) && !TypeP(ETSConstant,ets))
    throw Exception(nom," no es variable ni constant");
  @if (TypeP(ETSConstant,ets)) v=ets.Valor
    else if (Resultat==ResultatConstant)
      throw Exception(nom," no es constant");
  @var texp;
  <Acces(ets.Tipus,t,Resultat,v)> (
    =
    @if (TypeP(ETSConstant,ets))
      throw Exception("Assignació a constant");
    <Expressio(texp,ResultatValor,v)>
    @if (t!:=texp) throw Exception("Error de tipus de dades en =");
    | $ @if (Resultat==ResultatRes)
      throw Exception("el resultat no pot ser un valor o constant");
    )
  ))
```

Problema 4.5 Solució

```
Rule <Acces(te,&tr,Resultat,&v)>::= @tr=te; {
    @var ti,vi;
    @if (!typeP(TipusArray,tr))
        throw Exception("Tipus erroni en array");
    "[" <Expressio(ti,
        if (Resultat==ResultatConstant) ResultatConstant
        else ResultatValor,
        vi)>
    @if (ti!=TInt) throw Exception("Tipus erroni en índex");
    "]"
    @if (Resultat==ResultatConstant) v=v[vi];
    @tr=tr.TipusElements;
    | ...
```

Problema 4.6 Solució

```
Rule <factor(&t,Resultat,&V)> ::= ... |  
  let @TS.NouAmbit();  
  <LetDec(Resultat)> {, <LetDec(Resultat)> } in  
  <Expressio(t,Resultat,v)>  
  @TS.EliminarAmbit();
```

```
Rule <LetDec(Resultat)> ::= @Var t,nom,v;  
  identificador#(nom) =  
  <Expressio(t,  
    if (Resultat==ResultatConstant) ResultatConstant  
    else ResultatValor,v)>  
  @{  
    TS.ComprovarDuplicat(nom);  
    if (Resultat==ResultatConstant) TS.Insertar(ETSConstant(nom,t,v));  
    else TS.Insertar(ETSVariable(nom,t,unbound));  
  }
```