LA SALLE
ENGINYERIA I ARQUITECTURA

BABEL 2018
Definició del llenguatge

Llenguatges de Programació

MARIA ANTÒNIA MOZOTA I COLOMA

TAULA DE CONTINGUTS

1 Definició del llenguatge Babel 2018

El llenguatge Babel 2018 és un llenguatge de programació imperatiu. No és sensitiu a minúscules i majúscules.

1.1 Aspectes lexicogràfics

a) **Identificadors.** No són sensitius a minúscules/majúscules. Comencen obligatòriament amb una lletra. Només poden contenir lletres, dígits i _ (*underscore*). Tenen un mida màxima de 32 caràcters.

b) **Constants lògiques**. Només pot ser `CERT` o `FALS` **.**

c) **Constants senceres.** Les constants senceres són sense signe.

d) **Constants cadenes**. Constants *strings* denotant un conjunt de caràcters entre cometes dobles (per exemple: "hola"). Són vàlids tots aquells caràcters imprimibles. Les constants cadenes només s'utilitzen per escriure. Es poden declarar constants cadenes.

e) **Operadors relacionals.** Als operadors relacionals són: == , > , < , >= , <= , != (diferent).

f) **Operadors lògiques.** AND, OR i NOT.

g) **Operadors aritmètics.** +, - , * , /

h) **Tipus predefinits**. Hi ha dos tipus simple predefinits: `SENCER` i `LOGIC`.

i) **Paraules claus.** Totes les paraules clau (per exemple: `llavors`, `funcio`, ...) són paraules reservades.

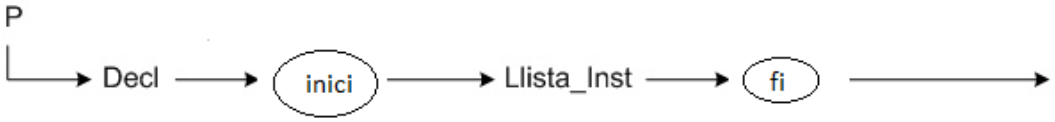
j) **Separadors**. Els espais (blancs) són significatius i actuen com separadors. A més a més d'aquests pot haver en qualsevol punt del programa d'entrada tabuladors i salts de línia.

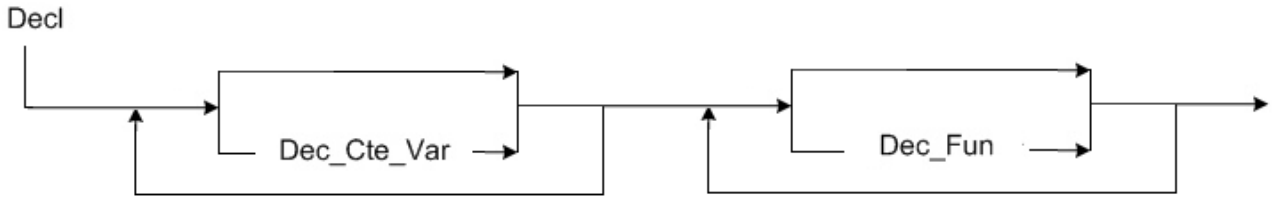
k) **Comentaris**. Els comentaris començant per **//** i acabant amb un salt de línia. Per tant, no poden ocupar més d'una línia. Els comentaris han de ser reconeguts i consumits. Els comentaris poden contenir qualsevol caràcter imprimible.

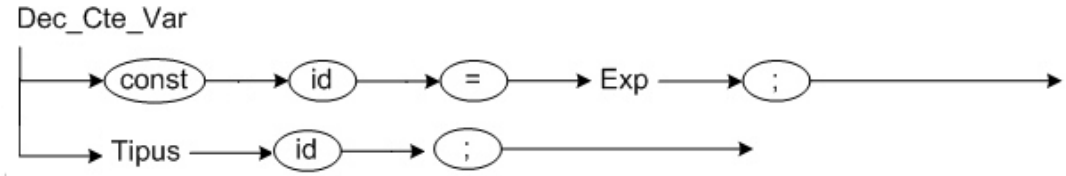
l) **Símbols especials**. Són símbols especials : **(**, **)**, **[**, **]**, **..** , **,** , **;** , **:** , etc.

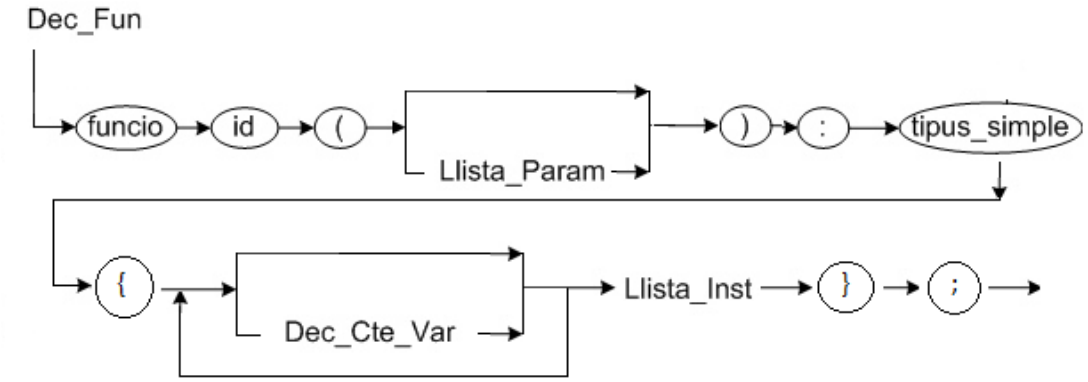
1.2 Sintaxi

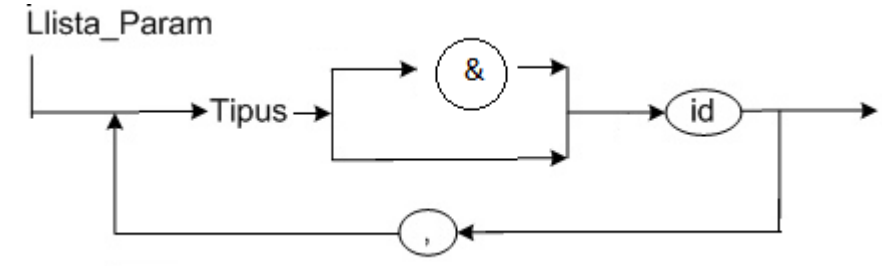
La sintaxi del llenguatge ve representada pels següents diagrames de *Conway.*



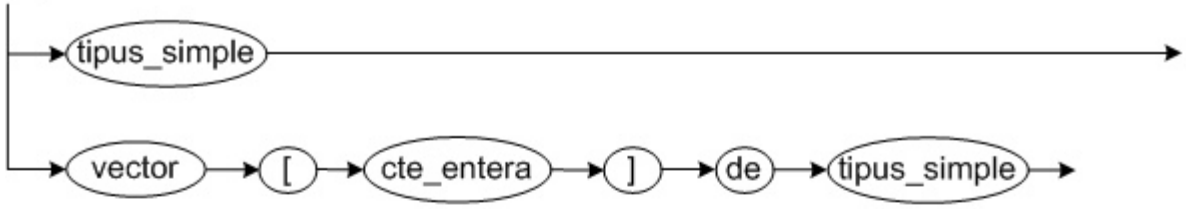




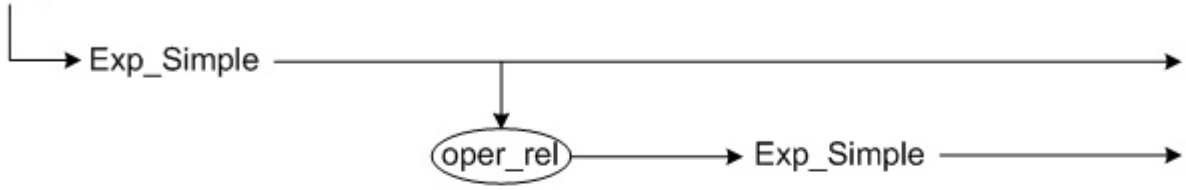




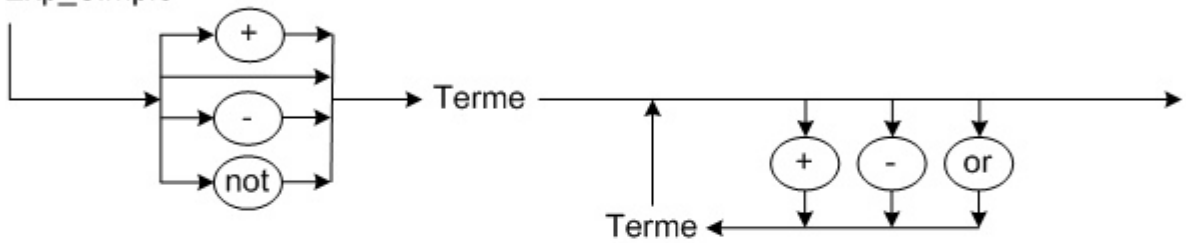
Tipus



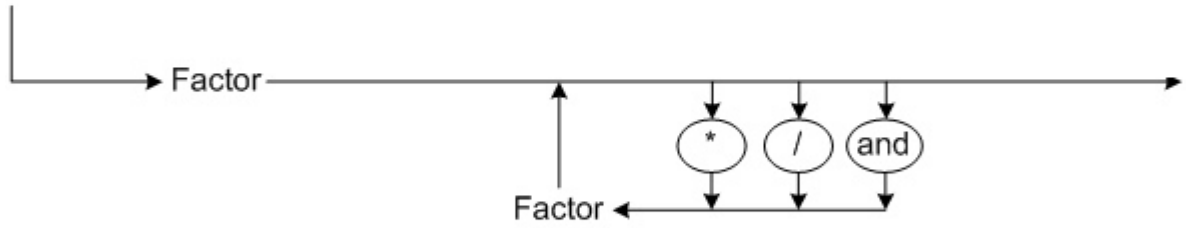
Exp



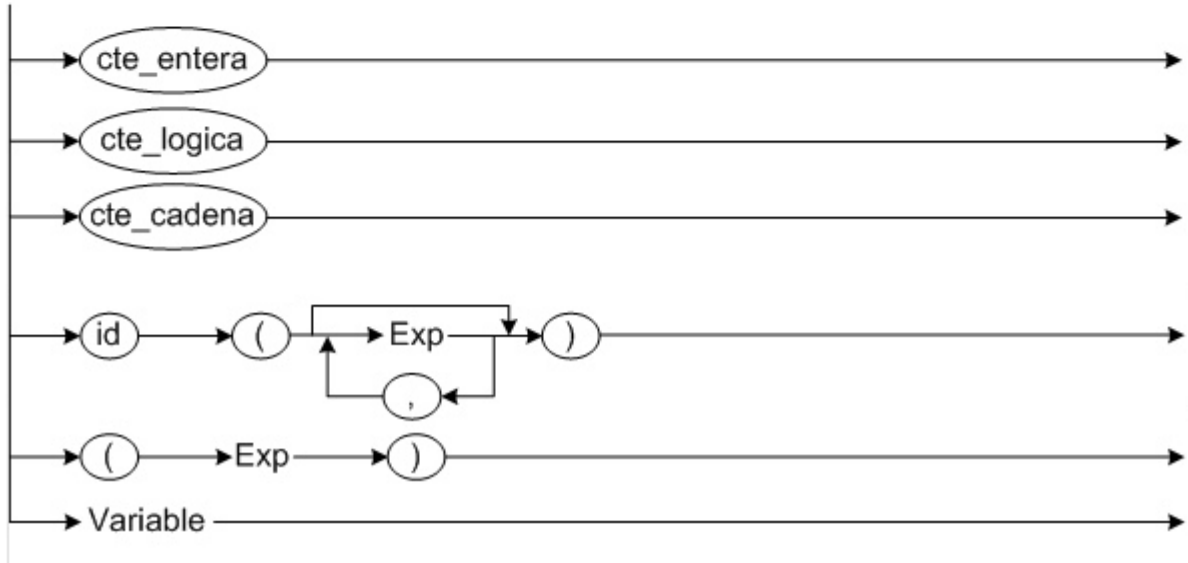
Exp_Simple



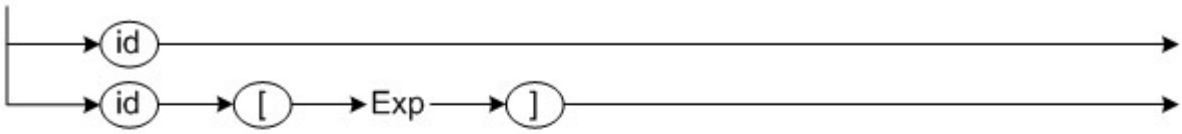
Terme



Factor

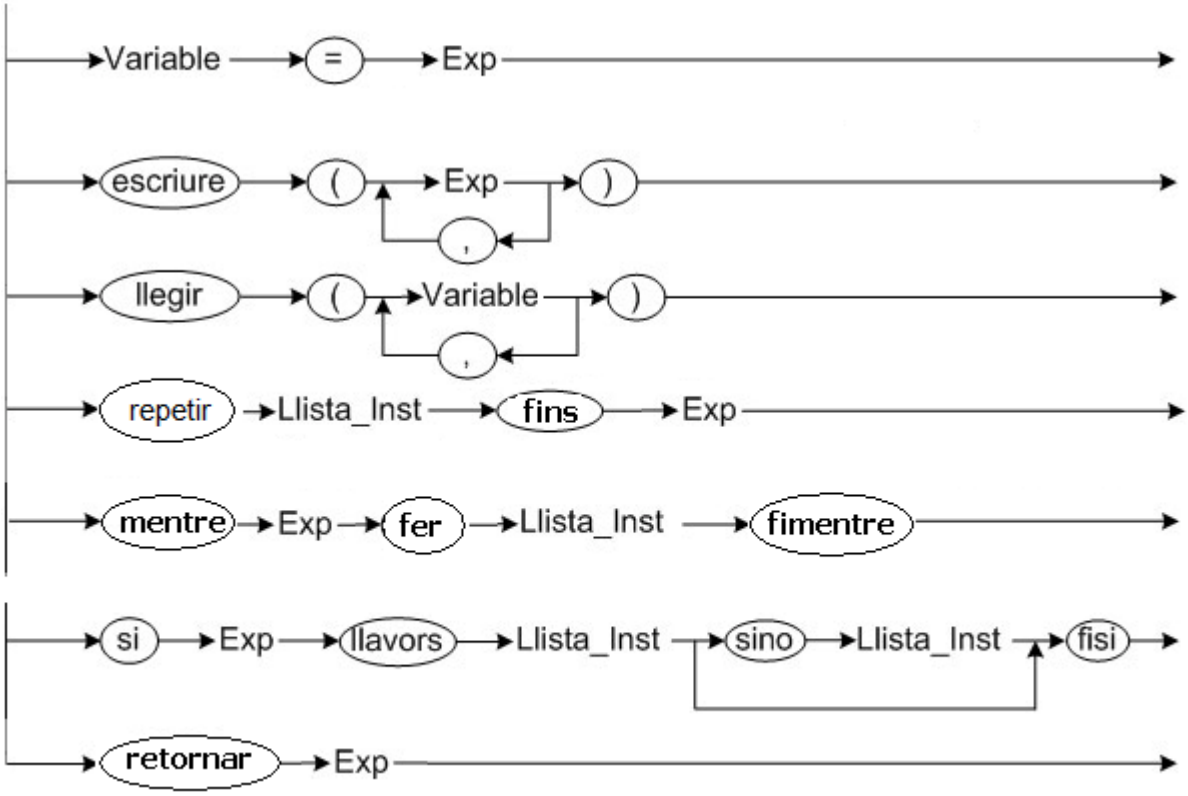


Variable



Llista_Inst

Inst



1.3 Semántica

La semàntica donada no és exhaustiva, si hi ha algun cas que no estigui especificat, el grup ha de proposar una.

- Declaració de constants:
  - Si la constant es declara a través d'algun identificador aquest ha d'existir com constant en el bloc actual o en el bloc principal.
  - L'expressió a la que s'inicialitza la constant ha de ser estàtica, és a dir, avaluada en temps de compilació. El tipus de la constant és el tipus de l'expressió.
  - Es poden declarar constants de tipus simple o de tipus cadena.

- Constants cadenes:
  - Es poden utilitzar com argument de l' instrucció **ESCRIURE**
  - Es poden declarar constants de tipus cadena

- Declaració de funcions:
  - El retorn del valor d'una funció es fa utilitzant l 'instrucció **RETORNAR**
  - Si es volem passar paràmetres per referència s'ha d'utilitzar **&.** Per defecte, el pas es per valor (còpia).
  - Cada funció defineix un nou context, és a dir, un nou abast.

- Declaració de constants, variables i funcions:
  - No pot existir un altre identificador amb el mateix nom en el mateix bloc.

- Ús d'identificadors :
  - Tot identificador de constant, variable i funció utilitzat ha d'estar prèviament declarat en el bloc on s'utilitza o en el bloc principal . S'ha de tenir en compte unes altres verificacions (segons el context i el tipus on aquest identificador es faci servir) que es tracten en l'apartat de instruccions *i* expressions.

- Tipus simples:
  - Sencer: és un tipus simple i ordenat. El seu rang ve determinat directament per la màquina per la que es compila.
  - Logic: és un tipus simple i ordenat (fals < cert).

- Tipus compostos :
  - Vectors estàtics d'1 dimensió. L'índex del vector va de 0 a N-1. on N és la constant sencera utilitzada a la declaració.

- Conversions i equivalències
  - Amb els tipus compostos es defineix una equivalència estructural.
  - No hi ha conversions implícites entre tipus simples.

- Expressions
  - Tota expressió té un tipus.
  - Tots els operands que intervenen en una expressió han de ser del mateix tipus.

  - Quan tenim una expressió que representa una crida a funció cal verificar:
    - que el nombre de paràmetres reals i formals és el mateix.

- que el tipus de cada paràmetre real és el mateix que el del formal
- no es poden passar per referència constants o expressions compostes de qualsevol tipus.

- En l'accés a una casella d'un vector verificar que el tipus de l'índex és de tipus sencer. Si l'índex és estàtic cal verificar en compilació que esta dins els límits. En execució, si l'índex esta fora dels límits, es produirà un error de execució.
- En expressions on apareguin operadors relacionals el tipus dels operands ha de ser el mateix, i només pot ser simple.
- La prioritat dels operadors esta definida en la sintaxi.
- La associativitat d'operadors d'igual prioritat és a esquerra.

- Instruccions

–<variable> = <expressio>

- S'avalua l'expressió i el seu valor se li assigna a la variable.
- La part esquerra i dreta de l'assignació ha de ser del mateix tipus. No es permet l'assignació de tipus compostos.
- En la part esquerra no pot haver un identificador de constant.

–llegir (<llista_variables>)

- Només es poden llegir variables de tipus simple.

–escriure (<llista_expressio>)

- Les expressions han de ser de tipus simple. Com cas especial es permet escriure constants cadenes.

–si <expressio> llavors <llistainst1> sino <llistainst2> fisi

- L'expressió ha de ser de tipus `lògic`.
- L'expressió ha d'avaluar-se en temps d'execució. Si és certa s'executaran les instruccions de `<llistainst1>` saltant-nos posteriorment les de la part del `sino`, en cas contrari s'executarà `<llistainst2>`.
- Aquesta instrucció no pot ser ambigua en cas de `si`'s aniuats.

–si <expressio> llavors <llistainst1> fisi

- L'expressió ha de ser de tipus `lògic`.
- L'expressió ha d'avaluar-se en temps d'execució. Si és certa s'executaran les instruccions de `<llistainst1>` en cas contrari salta a l' instrucció després del fisi
- Aquesta instrucció no pot ser ambigua en cas de `si`'s aniuats.

–repetir <llistainst> fins <expressio>

- s'executa la llista d'instruccions En arribar a fins s'avalua l'expressió. Si el valor es cert, l'execució del repetir s'acaba, si no, es torna a executar la llista d'instruccions i així successivament.
- l'expressió ha de ser de tipus lògic.

– mentre <expressio> fer <llistainst> fimentre

- s'avalua l'expressió. Si el valor es fals, l'execució del mentre s'acaba, si no es torna a executar la llista d'instruccions, es torna a avaluar l'expressió i així successivament.
- l'expressió ha de ser de tipus lògic.

- retornar <expressio>

- El tipus de la expressió ha de ser el que retorni la funció.