

Gramática en notación EBNF modificada para ser LL(1)

1. `<Program> ::= program id ; <BlockBody> .`
2. `<BlockBody> ::= [<ConstantDefinitionPart>] [<TypeDefinitionPart>] [<VariableDefinitionPart>] {<ProcedureDefinition>} <CompoundStatement>`
3. `<ConstantDefinitionPart> ::= const <ConstantDefinition> {<ConstantDefinition>}`
4. `<ConstantDefinition> ::= id = <Constant> ;`
5. `<TypeDefinitionPart> ::= type <TypeDefinition> {<TypeDefinition>}`
6. `<TypeDefinition> ::= id = <NewType> ;`
7. `<NewType> ::= <NewArrayType> | <NewRecordType>`
8. `<NewArrayType> ::= array [<IndexRange>] of id`
9. `<IndexRange> ::= <Constant> .. <Constant>`
10. `<NewRecordType> ::= record <FieldList> end`
11. `<FieldList> ::= <RecordSection> { ; <RecordSection> }`
12. `<RecordSection> ::= id { , id } : id`
13. `<VariableDefinitionPart> ::= var <VariableDefinition> {<VariableDefinition>}`
14. `<VariableDefinition> ::= <VariableGroup> ;`
15. `<VariableGroup> ::= id { , id } : id`
16. `<ProcedureDefinition> ::= procedure id <ProcedureBlock> ;`
17. `<ProcedureBlock> ::= [(<FormalParameterList>)] ; <BlockBody>`
18. `<FormalParameterList> ::= <ParameterDefinition> { ; <ParameterDefinition> }`
19. `<ParameterDefinition> ::= [var] <VariableGroup>`
20. `<Statement> ::= id <StatementGroup> | <IfStatement> | <WhileStatement> | <CompoundStatement> | ϵ`
21. `<StatementGroup> ::= {<Selector>} := <Expression> | <ProcedureStatement>`

Se elimina la Regla 21
22. `<ProcedureStatement> ::= (<ActualParameterList>)`
23. `<ActualParameterList> ::= <Expression> { , <Expression> }`

Se elimina la Regla 24
24. `<IfStatement> ::= if <Expression> then <Statement> [else <Statement>]`
25. `<WhileStatement> ::= while <Expression> do <Statement>`
26. `<CompoundStatement> ::= begin <Statement> { ; <Statement> } end`
27. `<Expression> ::= <SimpleExpression> [<RelationalOperator> <SimpleExpression>]`
28. `<RelationalOperator> ::= < | = | > | <= | >=`
29. `<SimpleExpression> ::= [<SignOperator>] <Term> {<AdditiveOperator> <Term>}`
30. `<SignOperator> ::= + | -`

- 31. $\langle \text{AdditiveOperator} \rangle ::= + \mid - \mid \text{or}$
- 32. $\langle \text{Term} \rangle ::= \langle \text{Factor} \rangle \{ \langle \text{MultiplyingOperator} \rangle \langle \text{Factor} \rangle \}$
- 33. $\langle \text{MultiplyingOperator} \rangle ::= * \mid \text{div} \mid \text{mod} \mid \text{and}$
- 34. $\langle \text{Factor} \rangle ::= \text{numeral} \mid \text{id} \{ \langle \text{Selector} \rangle \} \mid (\langle \text{Expression} \rangle) \mid \text{not } \langle \text{Factor} \rangle$
- 35. $\langle \text{Selector} \rangle ::= \langle \text{IndexSelector} \rangle \mid \langle \text{FieldSelector} \rangle$
- 36. $\langle \text{IndexSelector} \rangle ::= [\langle \text{Expression} \rangle]$
- 37. $\langle \text{FieldSelector} \rangle ::= . \text{id}$
- 38. $\langle \text{Constant} \rangle ::= \text{numeral} \mid \text{id}$

Se elimina la Regla 36

Gramática en notación BNF

1. $\langle \text{Program} \rangle ::= \text{program id} ; \langle \text{BlockBody} \rangle .$
2. $\langle \text{BlockBody} \rangle ::= \langle \text{ConstantDefinitionPart} \rangle \langle \text{TypeDefinitionPart} \rangle \langle \text{VariableDefinitionPart} \rangle \langle \text{ProcedureDefinition} \rangle \langle \text{CompoundStatement} \rangle$
3. $\langle \text{ConstantDefinitionPart} \rangle ::= \text{const} \langle \text{ConstantDefinition} \rangle \langle \text{ConstantDefinition2} \rangle \mid \epsilon$
4. $\langle \text{ConstantDefinition} \rangle ::= \text{id} = \langle \text{Constant} \rangle ;$
5. $\langle \text{ConstantDefinition2} \rangle ::= \langle \text{ConstantDefinition} \rangle \langle \text{ConstantDefinition2} \rangle \mid \epsilon$
6. $\langle \text{TypeDefinitionPart} \rangle ::= \text{type} \langle \text{TypeDefinition} \rangle \langle \text{TypeDefinition2} \rangle \mid \epsilon$
7. $\langle \text{TypeDefinition} \rangle ::= \text{id} = \langle \text{NewType} \rangle ;$
8. $\langle \text{TypeDefinition2} \rangle ::= \langle \text{TypeDefinition} \rangle \langle \text{TypeDefinition2} \rangle \mid \epsilon$
9. $\langle \text{NewType} \rangle ::= \langle \text{NewArrayType} \rangle \mid \langle \text{NewRecordType} \rangle$
10. $\langle \text{NewArrayType} \rangle ::= \text{array} [\langle \text{IndexRange} \rangle] \text{ of id}$
11. $\langle \text{IndexRange} \rangle ::= \langle \text{Constant} \rangle .. \langle \text{Constant} \rangle$
12. $\langle \text{NewRecordType} \rangle ::= \text{record} \langle \text{FieldList} \rangle \text{ end}$
13. $\langle \text{FieldList} \rangle ::= \langle \text{RecordSection} \rangle \langle \text{FieldList2} \rangle \mid \epsilon$
14. $\langle \text{FieldList2} \rangle ::= ; \langle \text{RecordSection} \rangle \langle \text{FieldList2} \rangle \mid \epsilon$
15. $\langle \text{RecordSection} \rangle ::= \text{id} \langle \text{RecordSection2} \rangle : \text{id}$
16. $\langle \text{RecordSection2} \rangle ::= , \text{id} \langle \text{RecordSection2} \rangle \mid \epsilon$
17. $\langle \text{VariableDefinitionPart} \rangle ::= \text{var} \langle \text{VariableDefinition} \rangle \langle \text{VariableDefinition2} \rangle \mid \epsilon$
18. $\langle \text{VariableDefinition} \rangle ::= \langle \text{VariableGroup} \rangle ;$
19. $\langle \text{VariableDefinition2} \rangle ::= \langle \text{VariableDefinition} \rangle \langle \text{VariableDefinition2} \rangle \mid \epsilon$
20. $\langle \text{VariableGroup} \rangle ::= \text{id} \langle \text{VariableGroup2} \rangle : \text{id}$
21. $\langle \text{VariableGroup2} \rangle ::= , \text{id} \langle \text{VariableGroup2} \rangle \mid \epsilon$
22. $\langle \text{ProcedureDefinition} \rangle ::= \text{procedure id} \langle \text{ProcedureBlock} \rangle ; \langle \text{ProcedureDefinition} \rangle \mid \epsilon$
23. $\langle \text{ProcedureBlock} \rangle ::= \langle \text{ProcedureBlock2} \rangle ; \langle \text{BlockBody} \rangle$
24. $\langle \text{ProcedureBlock2} \rangle ::= (\langle \text{FormalParameterList} \rangle) \mid \epsilon$
25. $\langle \text{FormalParameterList} \rangle ::= \langle \text{ParameterDefinition} \rangle \langle \text{ParameterDefinition2} \rangle \mid \epsilon$
26. $\langle \text{ParameterDefinition} \rangle ::= \text{var} \langle \text{VariableGroup} \rangle \mid \langle \text{VariableGroup} \rangle$
27. $\langle \text{ParameterDefinition2} \rangle ::= ; \langle \text{ParameterDefinition} \rangle \langle \text{ParameterDefinition2} \rangle \mid \epsilon$
28. $\langle \text{Statement} \rangle ::= \text{id} \langle \text{StatementGroup} \rangle \mid \langle \text{IfStatement} \rangle \mid \langle \text{WhileStatement} \rangle \mid \langle \text{CompoundStatement} \rangle \mid \epsilon$
29. $\langle \text{StatementGroup} \rangle ::= \langle \text{Factor2} \rangle ::= \langle \text{Expression} \rangle \mid \langle \text{ProcedureStatement} \rangle$
30. $\langle \text{ProcedureStatement} \rangle ::= (\langle \text{ActualParameterList} \rangle) \mid \epsilon$
31. $\langle \text{ActualParameterList} \rangle ::= \langle \text{Expression} \rangle \langle \text{Expression2} \rangle$
32. $\langle \text{Expression2} \rangle ::= , \langle \text{Expression} \rangle \langle \text{Expression2} \rangle \mid \epsilon$

33. <IfStatement> ::= **if** <Expression> **then** <Statement> <IfStatement2>

34. <IfStatement2> ::= **else** <Statement> | ϵ

35. <WhileStatement> ::= **while** <Expression> **do** <Statement>

36. <CompoundStatement> ::= **begin** <Statement> <Statement2> **end**

37. <Statement2> ::= ; <Statement> <Statement2> | ϵ

38. <Expression> ::= <SimpleExpression> <ExpressionGroup>

39. <ExpressionGroup> ::= <RelationalOperator> <SimpleExpression> | ϵ

40. <RelationalOperator> ::= < | = | > | <= | <> | >=

41. <SimpleExpression> ::= <Sign> <Term> <SimpleExpressionGroup>

42. <Sign> ::= <SignOperator> | ϵ

43. <SimpleExpressionGroup> ::= <AdditiveOperator> <Term> <SimpleExpressionGroup> | ϵ

44. <SignOperator> ::= + | -

45. <AdditiveOperator> ::= + | - | **or**

46. <Term> ::= <Factor> <Multiplying>

47. <Multiplying> ::= <MultiplyingOperator> <Factor> <Multiplying> | ϵ

48. <MultiplyingOperator> ::= * | **div** | **mod** | **and**

49. <Factor> ::= **numeral** | **id** <Factor2> | (<Expression>) | **not** <Factor>

50. <Factor2> ::= <Selector> <Factor2> | ϵ

51. <Selector> ::= <IndexSelector> | <FieldSelector>

52. <IndexSelector> ::= [<Expression>]

53. <FieldSelector> ::= . **id**

54. <Constant> ::= **numeral** | **id**

NO TERMINAL	FIRST	FOLLOW
Program	program	\$
BlockBody	const type var procedure begin	. ;
ConstantDefinitionPart	const ϵ	type var procedure begin
ConstantDefinition	id	id type var procedure begin
ConstantDefinition2	id ϵ	type var procedure begin
TypeDefinitionPart	type ϵ	var procedure begin
TypeDefinition	id	id var procedure begin
TypeDefinition2	id ϵ	var procedure begin
NewType	array record	;
NewArrayType	array	;
IndexRange	numeral id]
NewRecordType	record	;
FieldList	id	end
FieldList2	; ϵ	end
RecordSection	id	; end
RecordSection2	, ϵ	:
VariableDefinitionPart	var ϵ	procedure begin
VariableDefinition	id	id procedure begin
VariableDefinition2	id ϵ	procedure begin
VariableGroup	id	;)
VariableGroup2	, ϵ	:
ProcedureDefinition	procedure ϵ	begin
ProcedureBlock	(;	;
ProcedureBlock2	(ϵ	;
FormalParameterList	var id ϵ)
ParameterDefinition	var id	;)
ParameterDefinition2	; ϵ)
Statement	id if while begin ϵ	else end ;
StatementGroup	[. := (ϵ	else end ;
ProcedureStatement	(else end ;
ActualParameterList	+ - numeral id not ()
Expression2	, ϵ)
IfStatement	if	else end ;

IfStatement2	else ϵ	else end ;
WhileStatement	while	else end ;
CompoundStatement	begin	else end ; .
Statement2	; ϵ	end
Expression	+ - numeral id not (else end ; ,) do] then
ExpressionGroup	< = > <= <> >= ϵ	else end ; ,) do] then
RelationalOperator	< = > <= <> >=	+ - numeral id not (
SimpleExpression	+ - numeral id not (else end ; ,) do] then < = > <= <> >=
Sign	+ - ϵ	numeral id not (
SimpleExpressionGroup	+ - or ϵ	else end ; ,) do] then < = > <= <> >=
SignOperator	+ -	numeral id not (
AdditiveOperator	+ - or	numeral id not (
Term	numeral id not (+ - or else end ; ,) do] then < = > <= <> >=
Multiplying	* div mod and ϵ	+ - or else end ; ,) do] then < = > <= <> >=
MultiplyingOperator	* div mod and	numeral id not (
Factor	numeral id not (* div mod and + - or else end ; ,) do] then < = > <= <> >=
Factor2	[. ϵ	:= * div mod and + - or else end ; ,) do] then < = > <= <> >=
Selector	[.	[. := * div mod and + - or else end ; ,) do] then < = > <= <> >=
IndexSelector	[[. := * div mod and + - or else end ; ,) do] then < = > <= <> >=
FieldSelector	.	[. := * div mod and + - or else end ; ,) do] then < = > <= <> >=
Constant	numeral id	; ..]

PRODUCCIÓN	PREDICTION
Program -> program id ; BlockBody .	program
BlockBody -> ConstantDefinitionPart TypeDefinitionPart VariableDefinitionPart ProcedureDefinition CompoundStatement	const type var procedure begin
ConstantDefinitionPart -> const ConstantDefinition ConstantDefinition2	const
ConstantDefinitionPart -> ϵ	type var procedure begin
ConstantDefinition -> id = Constant ;	id
ConstantDefinition2 -> ConstantDefinition ConstantDefinition2	id
ConstantDefinition2 -> ϵ	type var procedure begin
TypeDefinitionPart -> type TypeDefinition TypeDefinition 2	type
TypeDefinitionPart -> ϵ	var procedure begin
TypeDefinition -> id = NewType ;	id
TypeDefinition2 -> TypeDefinition TypeDefinition2	id
TypeDefinition2 -> ϵ	var procedure begin
NewType -> NewArrayType	array
NewType -> NewRecordType	record
NewArrayType -> array [IndexRange] of id	array
IndexRange -> Constant .. Constant	numeral id
NewRecordType -> record RecordSection FieldList end	record
FieldList -> ; RecordSection FieldList	id
FieldList2 -> ; RecordSection FieldList2	;
FieldList2 -> ϵ	end
RecordSection -> id RecordSection2 : id	id
RecordSection2 -> , RecordSection2	,
RecordSection2 -> ϵ	:
VariableDefinitionPart -> var VariableDefinition VariableDefinition2	var
VariableDefinitionPart -> ϵ	procedure begin
VariableDefinition -> RecordSection ;	id
VariableDefinition2 -> VariableDefinition VariableDefinition2	id
VariableDefinition2 -> ϵ	procedure begin
VariableGroup -> id VariableGroup 2 : id	id
VariableGroup 2 -> , VariableGroup 2	,
VariableGroup 2 -> ϵ	:
ProcedureDefinition -> procedure id ProcedureBlock ; ProcedureDefinition	procedure
ProcedureDefinition -> ϵ	begin

ProcedureBlock -> ProcedureBlock2 ; BlockBody	(;
ProcedureBlock2 -> (FormalParameterList)	(
ProcedureBlock2 -> ϵ	;
FormalParameterList -> ParameterDefinition ParameterDefinition2	var id
FormalParameterList -> ϵ)
ParameterDefinition -> var RecordSection	var
ParameterDefinition -> RecordSection	id
ParameterDefinition2 -> ; ParameterDefinition ParameterDefinition2	;
ParameterDefinition2 -> ϵ)
Statement -> id StatementGroup	id
Statement -> IfStatement	if
Statement -> WhileStatement	while
Statement -> CompoundStatement	begin
Statement -> ϵ	else end ;
StatementGroup -> Factor2 := Expression	[. :=
StatementGroup -> ProcedureStatement	(else end ;
ProcedureStatement -> (ActualParameterList)	(
ProcedureStatement -> ϵ	else end ;
ActualParameterList -> Expression Expression2	+ - numeral id not (
Expression2 -> , Expression Expression2	,
Expression2 -> ϵ)
IfStatement -> if Expression then Statement IfStatement2	if
IfStatement2 -> else Statement	else
IfStatement2 -> ϵ	else end ;
WhileStatement -> while Expression do Statement	while
CompoundStatement -> begin Statement Statement2 end	begin
Statement2 -> ; Statement Statement2	;
Statement2 -> ϵ	end
Expression -> SimpleExpression ExpressionGroup	+ - numeral id not (
ExpressionGroup -> RelationalOperator SimpleExpression	< = > <= <> >=
ExpressionGroup -> ϵ	else end ; ,) do] then
RelationalOperator -> < = > <= <> >=	< = > <= <> >=
SimpleExpression -> Sign Term SimpleExpressionGroup	+ - numeral id not (

Sign -> SignOperator	+ -
Sign -> ϵ	numeral id not (
SimpleExpressionGroup -> AdditiveOperator Term SimpleExpressionGroup	+ - or
SimpleExpressionGroup -> ϵ	else end ; ,) do] then < = > <= <> >=
SignOperator -> + -	+ -
AdditiveOperator -> + - or	+ - or
Term -> Factor Multiplying	numeral id not (
Multiplying -> MultiplyingOperator Factor Multiplying	* div mod and
Multiplying -> ϵ	+ - or else end ; ,) do] then < = > <= <> >=
MultiplyingOperator -> * div mod and	* div mod and
Factor -> numeral	numeral
Factor -> id Factor2	id
Factor -> (Expression)	(
Factor -> not Factor	not
Factor2 -> Selector Factor2	[.
Factor2 -> ϵ	:= * div mod and + - or else end ; ,) do] then < = > <= <> >=
Selector -> IndexSelector	[
Selector -> FieldSelector	.
IndexSelector -> [Expression]	[
FieldSelector -> . id	.
Constant -> numeral	numeral
Constant -> id	id