Sintaxe Golang

BNF

Declaração de Função

FunctionDecl = "**func**" FunctionName Signature [FunctionBody]

FunctionName = identifier

FunctionBody = Block

Block

Block = "{" StatementList "}"

StatementList = { Statement ";" }

Declaração de Constantes

ConstDecl = "**const**" (ConstSpec | "(" { ConstSpec ";" } ")")

ConstSpec = IdentifierList [[Type] "=" ExpressionList]

IdentifierList = identifier { "," identifier }

ExpressionList = Expression { "," Expression }

Declaração de Tipos

TypeDecl = "**type**" (TypeSpec | "(" { TypeSpec ";" } ")")

TypeSpec = identifier Type

Declaração de Variáveis

VarDecl = "**var**" (VarSpec | "(" { VarSpec ";" } ")")

VarSpec = IdentifierList (Type ["=" ExpressionList] | "=" ExpressionList)

Operadores

Expression = UnaryExpr | Expression binary_op Expression

UnaryExpr = PrimaryExpr | unary_op UnaryExpr

binary_op = "||" | "&&"

rel_op = "==" | "!=" | "<" | "<=" | ">" | ">="

mul_op = "*" | "/" | "%"

unary_op = "+" | "-" | "!" | "^" | "*"

Laço de Repetição For

ForStmt = "**for**" [Condition | ForClause | RangeClause] Block

Condition = Expression

ForClause = [InitStmt] ";" [Condition] ";" [PostStmt]

InitStmt = SimpleStmt

PostStmt = SimpleStmt

Instruções

Statement = Declaration | SimpleStmt | ReturnStmt | BreakStmt | ContinueStmt | Block | IfStmt | SwitchStmt | ForStmt

SimpleStmt = EmptyStmt | ExpressionStmt | IncDecStmt | Assignment | ShortVarDecl

Declaration = ConstDecl | TypeDecl | VarDecl .

Switch

ExprSwitchStmt = "switch" [SimpleStmt ";"] [Expression] "{" { ExprCaseClause } "}"

ExprCaseClause = ExprSwitchCase ":" StatementList .

ExprSwitchCase = "case" ExpressionList | "default" .

Declaração do IF

IfStmt = "**if**" [SimpleStmt ";"] Expression Block ["else" (IfStmt | Block)]

Instrução de Retorno

ReturnStmt = "**return**" [ExpressionList]

EBNF

Production = production_name "=" [Expression] "."

Expression = Alternative { "|" Alternative }

Alternative = Term { Term }

Term = production_name | token ["…" token] | Group | Option | Repetition

Group = "(" Expression ")"

Option = "[" Expression "]"

Repetition = "{" Expression "}"