1 Group Details

Prithviraj Chavan (14196) (prithvi@iitk.ac.in)
Pooja Yadav (14469) (poojay@iitk.ac.in)
Divyat Mahajan (14227) (divyatm@iitk.ac.in)

2 T-Diagram of Compiler

Source Language: D
Implementation Language: Python
Target Language: x86

3 Syntax for D Language in BNF

- addExpression: mulExpression | addExpression ('+' | '-' | '∼') mulExpression;
- aliasDeclaration: 'alias' aliasInitializer (',' aliasInitializer)* ';' | 'alias' storageClass* type identifierList ';' ;
- aliasInitializer: Identifier templateParameters? '=' storageClass* type | Identifier templateParameters? '=' functionLiteralExpression ;
- aliasThisDeclaration: 'alias' Identifier 'this' ';' ;
- andAndExpression: orExpression | andAndExpression '&&' orExpression;
- andExpression: cmpExpression | andExpression '&' cmpExpression ;
- argumentList: assignExpression (',' assignExpression?)* ;
- arguments: '(' argumentList? ')' ;
- arrayInitializer: '[' ']' | '[' arrayMemberInitialization (',' arrayMemberInitialization?)* ']' ;
- arrayLiteral: '[' argumentList? ']' ;
- arrayMemberInitialization: (assignExpression ':')? nonVoidInitializer ;
- assignExpression: ternaryExpression (assignOperator expression)? ;
- assignOperator: '=' | '»=' | '«=' | '+=' | '-=' | '*=' | '%=' | '&=' | '/=' | '|=' | '≙' | '≚' ;
- attribute: | 'private' | 'protected' | 'public' | 'static' | 'extern' | 'final' | 'auto' | 'const' | 'immutable' ;
- attributeDeclaration: attribute ':' ;
- autoDeclaration: storageClass+ Identifier '=' initializer (',' Identifier '=' initializer)* ';' ;
- blockStatement: '' declarationsAndStatements? '' ;

- bodyStatement: 'body' blockStatement ;
- breakStatement: 'break' Identifier? ';' ;
- baseClass: type2 ;
- baseClassList: baseClass (',' baseClass)* ;
- builtinType: 'bool' | 'short' | 'ushort' | 'int' | 'uint' | 'long' | 'ulong' | 'char' | 'float' | 'void' ;
- caseRangeStatement: 'case' assignExpression ':' '...' 'case' assignExpression ':' declarationsAndStatements ;
- caseStatement: 'case' argumentList ':' declarationsAndStatements ;
- castExpression: 'cast' '(' (type | castQualifier)? ')' unaryExpression ;
- castQualifier: 'const' | 'immutable' ;
- classDeclaration: 'class' Identifier ';' | 'class' Identifier (':' baseClassList)? structBody ;
- cmpExpression: shiftExpression | equalExpression | identityExpression | relExpression ;
- constraint: 'if' '(' expression ')' ;
- constructor: 'this' templateParameters? parameters memberFunctionAttribute* constraint? (functionBody | ';') ;
- continueStatement: 'continue' Identifier? ';' ;
- declaration: attribute* declaration2 | attribute+ '' declaration* '' ;
- declaration2: aliasDeclaration | aliasThisDeclaration | anonymousEnumDeclaration | attributeDeclaration | classDeclaration | conditionalDeclaration | constructor | destructor | enumDeclaration | functionDeclaration | importDeclaration | mixinDeclaration | unionDeclaration | variableDeclaration ;
- declarationsAndStatements: declarationOrStatement+ ;
- declarationOrStatement: declaration | statement ;
- declarator: Identifier | Identifier '=' initializer | Identifier templateParameters '=' initializer ;
- defaultStatement: 'default' ':' declarationsAndStatements ;
- deleteExpression: 'delete' unaryExpression ;
- destructor: ' ' 'this' '(' ')' memberFunctionAttribute* (functionBody | ';');
- doStatement: 'do' statementNoCaseNoDefault 'while' '(' expression ')' ';';
- enumBody: '' enumMember (',' enumMember?)* '' ;

- anonymousEnumMember: type identifier '=' assignExpression | identifier '=' assignExpression | identifier ;
- anonymousEnumDeclaration: 'enum' (':' type)? '' anonymousEnumMember+ '' ;
- enumDeclaration: 'enum' Identifier (':' type)? ';' | 'enum' Identifier (':' type)? enumBody ;
- enumMember: Identifier | Identifier '=' assignExpression ;
- equalExpression: shiftExpression ('==' | '!=') shiftExpression ;
- expression: assignExpression (',' assignExpression)* ;
- expressionStatement: expression ';' ;
- forStatement: 'for' '(' (declaration | statementNoCaseNoDefault) expression? ';' expression? ')' declarationOrStatement ;
- foreachStatement: ('foreach' | 'foreach_reverse') '(' foreachTypeList ';' expression ')' declarationOrStatement | ('foreach' | 'foreach_reverse') '(' foreachType ';' expression '..' expression ')' declarationOrStatement ;
- foreachType: typeConstructors? type? Identifier | typeConstructors? type? Identifier ;
- foreachTypeList: foreachType (',' foreachType)* ;
- functionBody: blockStatement ;
- functionCallExpression: symbol arguments unaryExpression arguments | type arguments ;
- functionDeclaration: (storageClass+ | type) Identifier parameters memberFunctionAttribute* (functionBody | ';') | (storageClass+ | type) Identifier templateParameters parameters memberFunctionAttribute* constraint? (functionBody | ';') ;
- functionLiteralExpression: | 'function' type? (parameters functionAttribute*)? functionBody | parameters functionAttribute* functionBody | functionBody | Identifier '=>' assignExpression | 'function' type? parameters functionAttribute* '=>' assignExpression | parameters functionAttribute* '=>' assignExpression ;
- gotoStatement: 'goto' (Identifier | 'default' | 'case' expression?) ';' ;
- identifierChain: Identifier ('.' Identifier)* ;
- identifierList: Identifier (',' Identifier)* ;
- identifierOrTemplateChain: identifierOrTemplateInstance ('.' identifierOrTemplateInstance)* ;
- identifierOrTemplateInstance: Identifier | templateInstance ;
- identityExpression: shiftExpression ('is' | ('!' 'is')) shiftExpression ;

- ifStatement: 'if' '(' ifCondition ')' declarationOrStatement ('else' declarationOrStatement)? ifCondition: 'auto' Identifier '=' expression | type Identifier '=' expression | expression ;
- importBind: Identifier ('=' Identifier)? ;
- importBindings: singleImport ':' importBind (',' importBind)* ;
- importDeclaration: 'import' singleImport (',' singleImport)* (',' importBindings)? ';' | 'import' importBindings ';' ;
- importExpression: 'import' '(' assignExpression ')' ;
- index: assignExpression ('..' assignExpression)? ;
- indexExpression: unaryExpression '[' ']' | unaryExpression '[' index (',' index)* ']' ;
- initializer: 'void' | nonVoidInitializer ; ;
- isExpression: 'is' '(' type identifier? ')' 'is' '(' type identifier? ':' typeSpecialization ')' 'is' '(' type identifier? '=' typeSpecialization ')' 'is' '(' type identifier? ':' typeSpecialization ',' templateParameterList ')' 'is' '(' type identifier? '=' typeSpecialization ',' templateParameterList ')' ;
- labeledStatement: Identifier ':' declarationOrStatement? ; ; ;
- memberFunctionAttribute: functionAttribute | 'immutable' | 'const' | 'return' ;
- mixinDeclaration: mixinExpression ';' | templateMixinExpression ';' ;
- mixinExpression: 'mixin' '(' assignExpression ')' ;
- mulExpression: powExpression | mulExpression ('*' | '/' | ';
- newAnonClassExpression: 'new' arguments? 'class' arguments? baseClassList? structBody ;
- newExpression: 'new' type (('[' assignExpression ']') | arguments)? | newAnonClassExpression ;
- nonVoidInitializer: assignExpression | arrayInitializer | structInitializer ;
- orExpression: xorExpression | orExpression '|' xorExpression ;
- orOrExpression: andAndExpression | orOrExpression '||' andAndExpression ;
-
- parameter: parameterAttribute* type parameterAttribute* type Identifier? '...' parameterAttribute* type Identifier? ('=' assignExpression)?;
- parameterAttribute: typeConstructor | 'final' | 'out' | 'auto' | 'return' ;
- parameters: '(' parameter (',' parameter)* (',' '...')? ')' | '(' '...' ')' | '(' ')' ;

- postblit: 'this' '(' 'this' ')' memberFunctionAttribute* (functionBody | ';');
- powExpression: unaryExpression | powExpression '^
item^ unaryExpression ;
- primaryExpression: identifierOrTemplateInstance | '.' identifierOrTemplateInstance | typeConstructor '(' basicType ')' '.' Identifier | basicType '.' Identifier | basicType arguments | arrayLiteral | '(' expression ')' | functionLiteralExpression | traitsExpression | mixinExpression | importExpression | '$' | 'this' | 'null' | 'true' | 'false' | IntegerLiteral | FloatLiteral | StringLiteral+ | CharacterLiteral ;
- register: Identifier | Identifier '(' IntegerLiteral ')' ;
- relExpression: shiftExpression | relExpression relOperator shiftExpression;
- relOperator: '<' | '<=' | '>' | '>=' | '!<>=' | '!<>' | '<>' | '<>=' | '!>' | '!>=' | '!<' | '!<=' ;
- returnStatement: 'return' expression? ';' ;
- shiftExpression: addExpression | shiftExpression ('«' | '»') addExpression;
- singleImport: (Identifier '=')? identifierChain;
- statement: statementNoCaseNoDefault | caseStatement | caseRangeStatement | defaultStatement ;
- statementNoCaseNoDefault: labeledStatement | blockStatement | ifStatement | whileStatement | doStatement | forStatement | foreachStatement | switchStatement | finalSwitchStatement | continueStatement | breakStatement | returnStatement | gotoStatement | withStatement | conditionalStatement | expressionStatement ;
- storageClass: | typeConstructor | 'auto' | 'enum' | 'extern' | 'final' | 'static'
- structBody: '{' declaration* '}' ;
- switchStatement: 'switch' '(' expression ')' statement ; symbol: '.'? identifierOrTemplateChain ;
- ternaryExpression: orOrExpression ('?' expression ':' ternaryExpression)? ; ;
- type: typeConstructors? type2 typeSuffix* ;
- type2: builtinType | symbol | typeofExpression ('.' identifierOrTemplateChain)? | typeConstructor '(' type ')' ;
- typeConstructor: 'const' | 'immutable' ;
- typeConstructors: typeConstructor+ ;
- typeSpecialization: type | 'union' | 'class' | 'enum' | 'function' | 'const' | 'immutable' | 'return' | 'typedef' | '__parameters' ;

- typeSuffix: '*' | '[' type? ']' | '[' assignExpression ']' | '[' assignExpression '..' assignExpression ']' | ('delegate' | 'function') parameters memberFunctionAttribute* ; typeidExpression: 'typeid' '(' (type | expression) ')' ; typeofExpression: 'typeof' '(' (expression | 'return') ')' ;
- unaryExpression: primaryExpression | '&' unaryExpression | '!' unaryExpression | '*' unaryExpression | '+' unaryExpression | '-' unaryExpression | '~' unaryExpression | '++' unaryExpression | '–' unaryExpression | newExpression | deleteExpression | castExpression | functionCallExpression | indexExpression | '(' type ')' '.' identifierOrTemplateInstance | unaryExpression '.' newExpression | unaryExpression '.' identifierOrTemplateInstance | unaryExpression '–' | unaryExpression '++' ;
- unionDeclaration: 'union' Identifier templateParameters constraint? structBody | 'union' Identifier (structBody | ';') | 'union' structBody ;
- variableDeclaration: storageClass* type declarator (',' declarator)* ';' | storageClass* type identifier '=' functionBody ';' | autoDeclaration ;
- whileStatement: 'while' '(' expression ')' declarationOrStatement ;
- withStatement: 'with' '(' expression ')' statementNoCaseNoDefault ;
- xorExpression: andExpression | <xorExpression> '^' <andExpression> ;

4 Syntax rules deleted from your base language

Deleted Features

Double, byte, ubyte, wchar, dchar, real, ifloat, idouble, ireal, cfloat, cdouble, creal

Associative Arrays, Tuples, Ranges, Alias, pragma, pure, reg, override, abstract, synchronised

Exception Handling(try, catch, throw, nothrow), Conditional Compilation(debug, static if, version)

Struct, Function Templates, Interfaces, Modules, invariant, version, finally finalSwitchStatement, invariant, unittest

Assert, asm, inout, shared, _gshared, Vector, super, delegate, export
linkage Attribute, deprecated Attribute, atAttribute, alignAttribute
Static Constructor, Static Deconstructor, Static Assert,inStatement,OutStatement,unittest,asm operands

Rules for deleted features:

- alignAttribute: 'align' ('(' IntegerLiteral ')')? ;
- asmAddExp: asmMulExp | asmAddExp ('+' | '-') asmMulExp ;
- asmAndExp: asmEqualExp | asmAndExp '&' asmEqualExp ;
- asmBrExp: asmUnaExp | asmBrExp? '[' asmExp ']' ;
- asmEqualExp: asmRelExp | asmEqualExp ('==' | '!=') asmRelExp ;
- asmExp: asmLogOrExp ('?' asmExp ':' asmExp)? ;
- asmInstruction: Identifier | 'align' IntegerLiteral | 'align' Identifier | Identifier ':' asmInstruction | Identifier operands | 'in' operands | 'out' operands | 'int' operands ;
- asmLogAndExp: asmOrExp asmLogAndExp '&&' asmOrExp ;
- asmLogOrExp: asmLogAndExp | asmLogOrExp '||' asmLogAndExp ;
- asmMulExp: asmBrExp | asmMulExp ('*' | '/' | ';
- asmOrExp: asmXorExp | asmOrExp '|' asmXorExp ;
- asmPrimaryExp: IntegerLiteral | FloatLiteral | StringLiteral | register | identifierChain | '$' ;
- asmRelExp: asmShiftExp | asmRelExp (('<' | '<=' | '>' | '>=') asmShiftExp)? ;

- asmShiftExp: asmAddExp asmShiftExp ('«' | '»' | '»>') asmAddExp ;
- asmStatement: 'asm' functionAttributes? '' asmInstruction+ '' ;
- asmTypePrefix: Identifier Identifier? | 'byte' Identifier? | 'short' Identifier? | 'int' Identifier? | 'float' Identifier? | 'double' Identifier? | 'real' Identifier? ;
- asmUnaExp: asmTypePrefix asmExp | Identifier asmExp | '+' asmUnaExp | '-' asmUnaExp | '!' asmUnaExp | ' ' asmUnaExp | asmPrimaryExp ;
- asmXorExp: asmAndExp | asmXorExp '^' asmAndExp ;
- assertExpression: 'assert' '(' assignExpression (',' assignExpression)? ')' ;
- assignOperator: | '»>=' ;
- assocArrayLiteral: '[' keyValuePairs ']' ;
- atAttribute: '@' Identifier | '@' Identifier '(' argumentList? ')' | '@' '(' argumentList ')' | '@' TemplateInstance ;
- attribute: | pragmaExpression | alignAttribute | deprecated | atAttribute | linkageAttribute | 'export' | 'package' | 'abstract' | 'override' | 'synchronized' | 'scope' | 'inout' | 'shared' | '$_g$*shared*' | 'nothrow' | 'pure' | 'ref' ;
- builtinType: | 'byte' | 'ubyte' | 'wchar' | 'dchar' | 'double' | 'real' | 'ifloat' | 'idouble' | 'ireal' | 'cfloat' | 'cdouble' | 'creal' ;
- castQualifier: | 'const' 'shared' | 'inout' | 'inout' 'shared' | 'shared' | 'shared' 'const' | 'shared' 'inout' ;
- catch: 'catch' '(' type Identifier? ')' declarationOrStatement ;
- catches: catch+ | catch* lastCatch ;
- classDeclaration: | 'class' Identifier templateParameters constraint? (structBody | ';') | 'class' Identifier templateParameters constraint? (':' baseClassList)? structBody | 'class' Identifier templateParameters (':' baseClassList)? constraint? structBody ;
- cmpExpression: | inExpression ;
- compileCondition: versionCondition | debugCondition | staticIfCondition ;
- conditionalDeclaration: compileCondition declaration | compileCondition '' declaration* '' | compileCondition ':' declaration+ | compileCondition declaration 'else' ':' declaration* | compileCondition declaration 'else' declaration | compileCondition declaration 'else' '' declaration* '' | compileCondition '' declaration* '' 'else' declaration | compileCondition '' declaration* '' 'else' '' declaration* '' | compileCondition '' declaration* '' 'else' ':' declaration* | compileCondition ':' declaration+ 'else' declaration | compileCondition ':' declaration+ 'else' '' declaration* '' | compileCondition ':' declaration+ 'else' ':' declaration* ;

- conditionalStatement: compileCondition declarationOrStatement ('else' declarationOrStatement)? ;
- debugCondition: 'debug' ('(' (IntegerLiteral | Identifier) ')')? ;
- debugSpecification: 'debug' '=' (Identifier | IntegerLiteral) ';' ;
- declaration2: | debugSpecification | eponymousTemplateDeclaration | interfaceDeclaration | invariant | mixinTemplateDeclaration | pragmaDeclaration | sharedStaticConstructor | sharedStaticDestructor | staticAssertDeclaration | staticConstructor | staticDestructor | structDeclaration | templateDeclaration | versionSpecification ;
- deprecated: 'deprecated' ('(' StringLiteral+ ')')? ;
- finalSwitchStatement: 'final' switchStatement ;
- finally: 'finally' declarationOrStatement ; ;
- foreachType: typeConstructors? type? Identifier | typeConstructors? type? Identifier ;
- functionAttribute: atAttribute | 'pure' | 'nothrow' ;
- functionBody: (inStatement | outStatement | outStatement inStatement | inStatement outStatement)? bodyStatement? ;
- functionLiteralExpression: | 'delegate' type? (parameters functionAttribute*)? functionBody | 'delegate' type? parameters functionAttribute* '=>' assignExpression ;
- inExpression: shiftExpression ('in' | ('!' 'in')) shiftExpression ;
- inStatement: 'in' blockStatement ;
- interfaceDeclaration: 'interface' Identifier ';' | 'interface' Identifier (':' baseClassList)? structBody | 'interface' Identifier templateParameters constraint? (':' baseClassList)? structBody | 'interface' Identifier templateParameters (':' baseClassList)? constraint? structBody ;
- invariant: 'invariant' ('(' ')')? blockStatement ;
- keyValuePair: assignExpression ':' assignExpression ;
- keyValuePairs: keyValuePair (',' keyValuePair)* ','? ;
- lastCatch: 'catch' statementNoCaseNoDefault ;
- linkageAttribute: 'extern' '(' Identifier ('++' (',' identifierChain)?)? ')' ;
- memberFunctionAttribute: functionAttribute | 'inout' | 'shared' ;
- mixinTemplateDeclaration: 'mixin' templateDeclaration ;
- mixinTemplateName: symbol | typeofExpression '.' identifierOrTemplateChain ;

- module: moduleDeclaration? declaration* ;
- moduleDeclaration: deprecated? 'module' identifierChain ';' ;
- operands: asmExp | asmExp ',' operands ;
- outStatement: 'out' ('(' Identifier ')')? blockStatement ;
- parameterAttribute: | 'in' | 'ref' | 'lazy' | 'scope' ;
- pragmaDeclaration: pragmaExpression ';' ;
- pragmaExpression: 'pragma' '(' Identifier (',' argumentList)? ')' ;
- primaryExpression: | typeofExpression | typeidExpression | vector | assocArrayLiteral | isExpression | '$' | 'super' | '__DATE__' | '__TIME__' | '__TIMESTAMP__' | '__VENDOR__' | '__VERSION__' | '__FILE__' | '__LINE__' | '__MODULE__' | '__FUNCTION__' | '__PRETTY_FUNCTION__' ;
- scopeGuardStatement: 'scope' '(' Identifier ')' statementNoCaseNoDefault ;
- sharedStaticConstructor: 'shared' 'static' 'this' '(' ')' functionBody ;
- sharedStaticDestructor: 'shared' 'static' ' ' 'this' '(' ')' functionBody ;
- statementNoCaseNoDefault: | synchronizedStatement | tryStatement | throwStatement | scopeGuardStatement | asmStatement | staticAssertStatement | versionSpecification | debugSpecification
- staticAssertDeclaration: staticAssertStatement ;
- staticAssertStatement: 'static' assertExpression ';' ;
- staticConstructor: 'static' 'this' '(' ')' functionBody ;
- staticDestructor: 'static' '~' 'this' '(' ')' functionBody ;
- staticIfCondition: 'static' 'if' '(' assignExpression ')' ;
- storageClass: alignAttribute | linkageAttribute | atAttribute | deprecated | 'abstract' | 'nothrow' | 'override' | 'pure' | 'ref' | '__gshared' | 'scope'; | 'synchronized'
- structDeclaration: 'struct' Identifier? (templateParameters constraint? structBody | (structBody | ';')) ;
- structInitializer: '' structMemberInitializers? '' ;
- structMemberInitializer: (Identifier ':')? nonVoidInitializer ;
- structMemberInitializers: structMemberInitializer (',' structMemberInitializer?)* ;
- synchronizedStatement: 'synchronized' ('(' expression ')')? statementNoCaseNoDefault ;

- templateAliasParameter: 'alias' type? Identifier (':' (type | assignExpression))? ('=' (type | assignExpression))? ;
- templateArgument: type | assignExpression ;
- templateArgumentList: templateArgument (',' templateArgument?)* ;
- templateArguments: '!' ('(' templateArgumentList? ')') | templateSingleArgument ;
- templateDeclaration: 'template' Identifier templateParameters constraint? '' declaration* '' ;
- templateInstance: Identifier templateArguments ;
- templateMixinExpression: 'mixin' mixinTemplateName templateArguments? Identifier? ;
- templateParameter: templateTypeParameter | templateValueParameter | templateAliasParameter | templateTupleParameter | templateThisParameter ;
- templateParameterList: templateParameter (',' templateParameter?)* ;
- templateParameters: '(' templateParameterList? ')' ;
- templateSingleArgument: builtinType | Identifier | CharacterLiteral | StringLiteral | IntegerLiteral | FloatLiteral | 'true' | 'false' | 'null' | 'this' | '__DATE__' | '__TIME__' | '__TIMESTAMP__' | '__VENDOR__' | '__VERSION__' | '__FILE__' | '__LINE__' | '__MODULE__' | '__FUNCTION__' | '__PRETTY_FUNCTION__' ;
- templateThisParameter: 'this' templateTypeParameter ;
- templateTupleParameter: Identifier '...' ;
- templateTypeParameter: Identifier (':' type)? ('=' type)? ;
- templateValueParameter: type Identifier (':' assignExpression)? templateValueParameterDefault? ;
- templateValueParameterDefault: '=' ('__FILE__' | '__MODULE__' | '__LINE__' | '__FUNCTION__' | '__PRETTY_FUNCTION__' | assignExpression) ;
- throwStatement: 'throw' expression ';' ;
- traitsExpression: '__traits' '(' Identifier ',' TemplateArgumentList ')' ;
- tryStatement: 'try' declarationOrStatement (catches | catches finally | finally) ;
- type2: | vector ;
- typeConstructor: | 'inout' | 'shared' ;
- typeSpecialization: | 'struct' | 'interface' | 'delegate' | 'super' | 'inout' | 'shared' ;

- unaryExpression: | assertExpression ;
- unittest: 'unittest' blockStatement ;
- vector: '__vector' '(' type ')' ;
- versionCondition: 'version' '(' (IntegerLiteral | Identifier | 'unittest' | 'assert') ')' ;
- versionSpecification: 'version' '=' (Identifier | IntegerLiteral) ';' ;

5 Description of the new constructs added

No new constructs were added to the language grammar

6 Tools for the Project

Lexer, Parse Generator