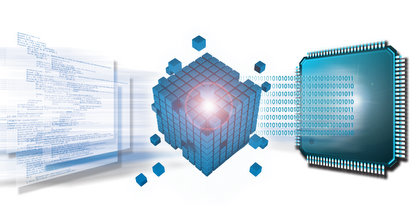
 

Compiladores

2º Semestre



**Relatório do Projecto**

Trabalho realizado por:

* Daniel Coelho Bastos 2009109605
* Tiago Martins Andrade 2010165760

2014/2015

# Índice

[Introdução 2](#_Toc418122474)

[Fase 1 – Análise Lexical 3](#_Toc418122475)

[Fase 2 – Análise Sintática 4](#_Toc418122476)

[Fase 3 – Análise Semântica 5](#_Toc418122477)

[Fase 4 – Geração de Código 6](#_Toc418122478)

[Conclusão 7](#_Toc418122479)

# Introdução

Este projeto consiste no desenvolvimento de um compilador para a linguagem “mili-Pascal,” que é um pequeno subconjunto da linguagem Pascal Standard ISO 7185:1990 com extensões relativas à passagem de parâmetros através da linha de comandos.

O compilador desenvolvido neste trabalho foi implementado em C com recurso às ferramentas lex, yacc e LLVM.

Este projeto está dividido nas seguintes fases:

1. Analisador lexical (verifica se os tokens introduzidos são válidos)
2. Analisador gramatical (verifica se os tokens seguem a gramática definida)
3. Construção da árvore de sintaxe abstrata e da tabela de símbolos
4. Geração de código (cria um cheiro em código LLVM que implementa as mesmas funcionalidades do programa de entrada)

# Fase 1 – Análise Lexical

O analisador lexical deve ser implementado em C utilizando a ferramenta lex. Os tokens da linguagem são apresentados de seguida.

**ASSIGN** = ":="

**BEGIN** = "begin"

**COLON** = ":"

**COMMA** = ","

**DO** = "do"

**DOT** = "."

**ELSE** = "else"

**END** = "end"

**FORWARD** = "forward"

**FUNCTION** = "function"

**IF** = "if"

**LBRAC** = "("

**NOT** = "not"

**OUTPUT** = "output"

**PARAMSTR** = "paramstr"

**PROGRAM** = "program"

**RBRAC** = ")"

**REPEAT** = "repeat"

**SEMIC** = ";"

**THEN** = "then"

**UNTIL** = "until"

**VAL** = "val"

**VAR** = "var"

**WHILE** = "while"

**WRITELN** = "writeln"

**OP1** = "and" | "or"

**OP2** = "<" | ">" | "=" | "<>" | "<=" | ">="

**OP3** = "+" | "-"

**OP4** = "\*" | "/" | "mod" | "div"

**RESERVED** : palavras reservadas e identificadores requeridos do Pascal standard não usados. NOTA: os identificadores requeridos boolean, false, integer, real e true serão usados em fases posteriores do projeto, e não deverão ser RESERVED.

No caso de o interpretador detetar um caractere inválido é feita uma contagem de linhas e de colunas para o mostrar não só o caractere inválido mas também a linha e a coluna onde se encontra o erro. Também é mantida em memória a linha e coluna do início do token actual. Isto permite no caso de uma expressão não passar a análise lexical mostrar não só o tipo de erro mas também a linha e a coluna onde ocorre.

# Fase 2 – Análise Sintática

O analisador sintático foi implementado através do interpretador lexical descrito anteriormente e com recurso à ferramenta yacc. A seguinte gramática em notação EBNF define a sintaxe utilizada para o desenvolvimento deste compilador da linguagem miliPascal.

**Prog** → ProgHeading SEMIC ProgBlock DOT

**ProgHeading** → PROGRAM ID LBRAC OUTPUT RBRAC

**ProgBlock** → VarPart FuncPart StatPart

**VarPart** → [ VAR VarDeclaration SEMIC { VarDeclaration SEMIC } ]

**VarDeclaration** → IDList COLON ID

**IDList** → ID { COMMA ID }

**FuncPart** → { FuncDeclaration SEMIC }

**FuncDeclaration** → FuncHeading SEMIC FORWARD

**FuncDeclaration** → FuncIdent SEMIC FuncBlock

**FuncDeclaration** → FuncHeading SEMIC FuncBlock

**FuncHeading** → FUNCTION ID [ FormalParamList ] COLON ID

**FuncIdent** → FUNCTION ID

**FormalParamList** → LBRAC FormalParams { SEMIC FormalParams } RBRAC

**FormalParams** → [ VAR ] IDList COLON ID

**FuncBlock** → VarPart StatPart

**StatPart** → CompStat

**CompStat** → BEGIN StatList END

**StatList** → Stat { SEMIC Stat }

**Stat** → CompStat

**Stat** → IF Expr THEN Stat [ ELSE Stat ]

**Stat** → WHILE Expr DO Stat

**Stat** → REPEAT StatList UNTIL Expr

**Stat** → VAL LBRAC PARAMSTR LBRAC Expr RBRAC COMMA ID RBRAC

**Stat** → [ ID ASSIGN Expr ]

**Stat** → WRITELN [ WritelnPList ]

**WritelnPList** → LBRAC ( Expr | STRING ) { COMMA ( Expr | STRING ) } RBRAC

**Expr** → Expr (OP1 | OP2 | OP3 | OP4) Expr

**Expr** → (OP3 | NOT) Expr

**Expr** → LBRAC Expr RBRAC

**Expr** → INTLIT | REALLIT

**Expr** → ID [ ParamList ]

**ParamList** → LBRAC Expr {COMMA Expr} RBRAC

# Fase 3 – Análise Semântica

A análise semântica da linguagem miliPascal deve ser implementada em C utilizando as ferramentas lex e yacc tendo por base o analisador sintático desenvolvido na fase anterior. O analisador deverá chamar-se mpasemantic, ler o ficheiro a processar através do stdin, e detetar a existência de quaisquer erros (lexicais, de sintaxe, ou de semântica) no ficheiro de entrada.

# Fase 4 – Geração de Código

# Conclusão