CS335 — Assignment 0

March 18, 2018

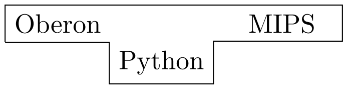
1 Group Members

Rohit Gupta (150593; rgupta@iitk.ac.in)

Shikhar Mahajan (150669; smahajan@iitk.ac.in)

Ankit Bhardwaj (150101; bhankit@iitk.ac.in)

2 T-Diagram



3 Grammar in BNF/EBNF

⟨*letter*⟩	:= "A" ... "Z" | "a" .... "z"
⟨*digit*⟩	:= "0" ... "9"
⟨*real*⟩	:= ⟨*digit*⟩ {⟨*digit*⟩} "." {⟨*digit*⟩}
⟨*integer*⟩	:= ⟨*digit*⟩ {⟨*digit*⟩}
⟨*char*⟩	:= '"'⟨*any ASCII character*⟩'"'
⟨*string*⟩	:= '"' {⟨*any ASCII character*⟩} '"'
⟨*number*⟩	:= ⟨*integer*⟩ | ⟨*real*⟩
⟨*element*⟩	:= ⟨*expression*⟩
⟨*set*⟩	:= "{" [⟨*element*⟩ "," ⟨*element*⟩] "}"

⟨*MulOperator*⟩	:=	`"*"` | `"/"` | `"%"` | `"&"`
⟨*AddOperator*⟩	:=	`"+"` | `"-"` | `"|"`
⟨*relation*⟩	:=	`"="` | `"!="` | `"<"` | `"<="` | `">"` | `">="` | `"IN"` | `"IS"`
⟨*ident*⟩	:=	⟨*letter*⟩ {⟨*letter*⟩ | ⟨*digit*⟩}
⟨*identdef*⟩	:=	[`@`] ⟨*ident*⟩
⟨*factor*⟩	:=	⟨*number*⟩ | ⟨*char*⟩ | ⟨*string*⟩ | `"NIL"` | ⟨*set*⟩ | ⟨*designator*⟩ [⟨*ActualParameters*⟩] | `"("` ⟨*expression*⟩ `")"` | `"!"` ⟨*factor*⟩ | `"ABS"` ⟨*factor*⟩ | `"CHR"` ⟨*factor*⟩ | `"ORD"` ⟨*factor*⟩ | `"INTEGER"` | `"BOOLEAN"` | `"CHAR"` | `"STRING"` | `" REAL "` | `" SET "`
⟨*expression*⟩	:=	⟨*SimpleExpression*⟩ [⟨*relation*⟩ ⟨*SimpleExpression*⟩]
⟨*SimpleExpression*⟩	:=	[`"+"`|`"-"`] ⟨*term*⟩ {⟨*AddOperator*⟩ ⟨*term*⟩}
⟨*term*⟩	:=	⟨*factor*⟩ {⟨*MulOperator*⟩ ⟨*factor*⟩}
⟨*qualident*⟩	=	[⟨*identdef*⟩ `.`] ⟨*identdef*⟩
⟨*designator*⟩	:=	⟨*qualident*⟩ {`"."` ⟨*identdef*⟩ | `"["` ⟨*ExpList*⟩ `"]"` | `"("` ⟨*qualident*⟩ `")"` }
⟨*ExpList*⟩	:=	⟨*expression*⟩ {`","` ⟨*expression*⟩}
⟨*ActualParameters*⟩	:=	`"("` [⟨*ExpList*⟩] `")"`
⟨*ConstantDeclaration*⟩	:=	⟨*ident*⟩ `":="` ⟨*expression*⟩
⟨*vartype*⟩	:=	`"INTEGER"` | `"BOOLEAN"` | `"CHAR"` | `"STRING"` | `" REAL "` | `" SET "`
⟨*ArrayType*⟩	:=	`"ARRAY"` ⟨*length*⟩ {`","` ⟨*length*⟩} OF ⟨*type*⟩
⟨*length*⟩	:=	⟨*expression*⟩
⟨*RecordType*⟩	=	`"RECORD"` [`"("` ⟨*BaseType*⟩ `")"`] ⟨*FieldListSequence*⟩ END
⟨*BaseType*⟩	=	⟨*qualident*⟩

```
⟨FieldListSequence⟩      := ⟨FieldList⟩ {";" ⟨FieldList⟩}

⟨FieldList⟩              := [⟨IdentList⟩ ":" ⟨type⟩]

⟨IdentList⟩              := ⟨ident⟩ {"," ⟨ident⟩}

⟨PointerType⟩            := "POINTER" "TO" ⟨type⟩

⟨StatementSequence⟩      := { ⟨statement⟩ ";"}

⟨statement⟩              := [⟨assignment⟩ | ⟨ProcedureCall⟩ | ⟨IfStatement⟩ | ⟨SwitchStatement⟩
                            | ⟨WhileStatement⟩ | ⟨ForStatement⟩ | ⟨DoWhileStatement⟩
                            |"EXIT" | "RETURN" [⟨expression⟩] | ⟨io_statement⟩
                            | ⟨FileStatement⟩ | "BREAK" | "CONTINUE"]

⟨assignment⟩             := ⟨designator⟩ ":=" ⟨expression⟩

⟨ProcedureCall⟩          := ⟨designator⟩ [⟨ActualParameters⟩]

⟨IfStatement⟩            := "IF" ⟨expression⟩ "THEN" ⟨StatementSequence⟩ {"ELSIF"
                            ⟨expression⟩ "THEN" ⟨StatementSequence⟩} ["ELSE"
                            ⟨StatementSequence⟩] "END"

⟨SwitchStatement⟩        := "SWITCH" ⟨expression⟩ "BEGIN" ⟨case⟩ {"|" ⟨case⟩}
                            ["ELSE" ":" ⟨StatementSequence⟩] "END"

⟨case⟩                   := "CASE" ":" ⟨expression⟩ ⟨StatementSequence⟩

⟨WhileStatement⟩         := "WHILE" ⟨expression⟩ "DO" "BEGIN" ⟨StatementSequence⟩
                            "END"

⟨ForStatement⟩           := "FOR" "(" ⟨assignment⟩ ";" ⟨expression⟩ ";" ⟨assignment⟩
                            ")" "BEGIN" ⟨StatementSequence⟩ "END"

⟨DoWhileStatement⟩       := "DO" ⟨StatementSequence⟩ "WHILE" ⟨expression⟩

⟨TypeDeclaration⟩        := ⟨ident⟩ "=" ⟨type⟩

⟨type⟩                   := ⟨ident⟩ | ⟨vartype⟩ |⟨ArrayType⟩ | ⟨RecordType⟩ |
                            ⟨PointerType⟩

⟨VariableDeclaration⟩    := ⟨IdentList⟩ ":" ⟨type⟩

⟨ProcedureDeclaration⟩   := ⟨ProcedureHeading⟩ ";" ⟨ProcedureBody⟩ ⟨ident⟩ ";"
```

⟨*ProcedureHeading*⟩ := "PROCEDURE" ⟨*ident*⟩ [⟨*FormalParameters*⟩] ":" ⟨*type*⟩

⟨*FormalParameters*⟩ := "(" [⟨*FPSection*⟩ {";" ⟨*FPSection*⟩}] ")"

⟨*FPSection*⟩ := ⟨*ident*⟩ {"," ⟨*ident*⟩} ":" ⟨*type*⟩

⟨*ProcedureBody*⟩ := ⟨*DeclarationSequence*⟩ "BEGIN" ⟨*StatementSequence*⟩ "END"

⟨*module*⟩ := "MODULE" ⟨*ident*⟩ ";" ⟨*DeclarationSequence*⟩ "BEGIN" ⟨*StatementSequence*⟩ "END" ⟨*ident*⟩ "."

⟨*DeclarationSequence*⟩ := { "CONST" {⟨*ConstantDeclaration*⟩ ";" } | "TYPE" {⟨*TypeDeclaration*⟩ ";" } | "VAR" {⟨*VariableDeclaration*⟩ ";" } | {⟨*ProcedureDeclaration*⟩ ";" } }

⟨*io_statement*⟩ := "WRITE" "("⟨*expression*⟩")" | "WRITEINT" "("⟨*expression*⟩")" | "WRITEREAL" "("⟨*expression*⟩")" | "WRITELN" "("⟨*expression*⟩")" | "READ" "("⟨*expression*⟩")"

⟨*FileStatement*⟩ := ⟨*identdef*⟩ "=" "FOPEN" "(" ⟨*string*⟩ "," ⟨*char*⟩ ")" | "FCLOSE" "(" ⟨*identdef*⟩ ")" | "FPRINTF" "(" ⟨*identdef*⟩ "," string ")" | "FREAD" "(" ⟨*identdef*⟩ "," ⟨*identdef*⟩ "," ⟨*integer*⟩ ")"

4 Deleted Grammar

⟨*module*⟩ := [ImportList]

⟨*ImportList*⟩ := "IMPORT" ⟨*import*⟩ {, ⟨*import*⟩} ";"

⟨*import*⟩ := ⟨*ident*⟩ [:= ⟨*ident*⟩]

⟨*integer*⟩ := ⟨*digit*⟩ ⟨*hexDigit*⟩ "H"

⟨*hexDigit*⟩ := ⟨*digit*⟩ | "A" | "B" | "C" | "D" | "E" | "F"

⟨*CharConstant*⟩ := ⟨*digit*⟩ ⟨*hexDigit*⟩ "X"

⟨*real*⟩ := ⟨*digit*⟩ {⟨*digit*⟩} . {⟨*digit*⟩} [⟨*ScaleFactor*⟩]

caleFactor := ("E" | "D") ["+" | "-"] ⟨*digit*⟩ {⟨*digit*⟩}

⟨*DeclarationSequence*⟩ := {⟨*ForwardDeclaration*⟩ `";"`}

⟨*ForwardDeclaration*⟩ = `"PROCEDURE"` `"^"` ⟨*ident*⟩ [`"*"`] [⟨*FormalParameters*⟩]

⟨*statement*⟩ := [WithStatement]

⟨*WithStatement*⟩ := `"WITH"` ⟨*qualident*⟩ `" "` ⟨*qualident*⟩ `"DO"` ⟨*StatementSequence*⟩ `"END"`

⟨*RepeatStatement*⟩ := `"REPEAT"` ⟨*StatementSequence*⟩ `"UNTIL"` ⟨*expression*⟩

⟨*LoopStatement*⟩ := `"LOOP"` ⟨*StatementSequence*⟩ `"END"`

5 Semantic Description of added constructs

We have introduces two new type of statements:

- io_statement: For printing and reading into standard output/input.
- FileStatement: We basically have included 4 basic file i/o statements in our grammar. Two of them are simply for opening and closing of files and the other two are for writing and reading into files. In FREAD statement the two identifiers taken into account correspond to file and character array where the copied characters will be stored, and the integer corresponds to how many characters we have to copy. Similarly, FPRINTF has two arguments as the file identifier and the string that has to be copied in the file.

6 Tools

PLY(Python Lex & Yacc).