Tokens

```
letter
digit
```

```
MP_AND
MP_BEGIN
MP_BOOLEAN
MP_DIV
MP_DO
MP_DOWNTO
MP_ELSE
MP_END
MP_FALSE
MP_FIXED
MP_FLOAT
MP_FOR
MP_FUNCTION
MP_IF
MP_INTEGER
MP_MOD
MP_NOT
MP_OR
MP_PROCEDURE
MP_PROGRAM
MP_READ
MP_REPEAT
MP_STRING
MP_THEN
MP_TRUE
MP_TO
MP_Type
MP_UNTIL
MP_VAR
MP_WHILE
MP_WRITE
MP_WRITELN
```

MP_IDENTIFIER
MP_INTEGER_LIT
MP_FIXED_LIT
MP_FLOAT_LIT
MP_STRING_LIT

MP_ASSIGN
MP_COLON
MP_COMMA
MP_EQUAL
MP_FLOAT_DIVIDE
MP_GEQUAL
MP_GTHAN
MP_LEQUAL
MP_LPAREN
MP_LTHAN
MP_MINUS
MP_NEQUAL
MP_PERIOD
MP_PLUS
MP_RPAREN
MP_SCOLON
MP_TIMES

MP_EOF
MP_RUN_COMMENT
MP_RUN_STRING
MP_ERROR

microPascal Tokens

Lexemes

Auxiliary Regular Expressions

```
a|b|c|...|z|A|B|C|...|Z
0|1|2|3|4|5|6|7|8|9
```

Reserved Words

```
"and"
"begin"
"Boolean"
"div"
"do"
"downto"
"else"
"end"
"false"
"fixed"
"float"
"for"
"function"
"if"
"integer"
"mod"
"not"
"or"
"procedure"
"program"
"read"
"repeat"
"string"
"then"
"true"
"to"
"type"
"until"
"var"
"while"
"write"
"writeln"
```

Identifiers and Literals

```
(letter | "_"(letter | digit)){["_"](letter | digit)}
digit{digit}
digit{digit} "." digit{digit}
(digit{digit} | digit{digit} "." digit{digit}) ("e"|"E")["+"|"-"]digit{digit}
"'" {"''" | AnyCharacterExceptApostropheOrEOL} "'"
```

Single String Tokens

```
":="
":"
","
"="
"/"
">="
">"
"<="
"("
"<"
"-"
"<>"
"."
"+"
")"
";"
"*"
```

Special Tokens

```
token for end-of-file character
token for run-on comment error
token for run-on string error
token for other scan errors
```

Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ

Adam
Tim
Tim
Tim
JJ

Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ
Adam
Tim
JJ

Adam
Tim
JJ
Adam