Nas aulas de laboratório vamos criar um compilador para uma linguagem parecida com o Java (MicroJava). Vamos aprender como colocar as técnicas vistas no curso de construção de compiladores em prática e estudar alguns detalhes envolvidos na implementação de um compilador real.

Este documento descreve a linguagem MicroJava que será usada nas aulas práticas do curso de compiladores. MicroJava tem alguns aspectos similares ao Java, mas bem mais simples.

Características gerais

- Um programe em MicroJava consiste em um único arquivo com métodos e variáveis estáticas. Não há classes externas, mas somente classes internas (*inner classes*) que podem ser usadas como tipos de dados.
- O método principal de um programa em MicroJava é sempre chamado main(). Quando um programa em MicroJava é invocado esse método é executado.
- Existem constantes do tipo int (e.g. 3) e char (e.g. 'x') mas não constantes do tipo string
- Todas as variáveis do programa são estáticas
- Os tipos primitivos são int e char
- Tipos referenciáveis: arrays unidimensionais como em Java e classes com atributos, mas sem métodos.
- Métodos estáticos na classe principal
- Não tem coletor de lixo (*garbage collector*): os objetos alocados são somente desalocados quando o programa termina.
- Tem três procedimentos pré-declarados: ord, chr e len

Programa exemplo

```
program P
  final int size = 10;
  class Table {
    int[] pos;
    int[] neg;
  }
  Table val;
{
  void main()
    int x, i;
  { //---------- Initialize val ------------
    val = new Table;
    val.pos = new int[size];
    val.neg = new int[size];
    i = 0;
    while (i < size) {
      val.pos[i] = 0; val.neg[i] = 0;
      i = i + 1;
    }
    //------------ Read values -------------
    read(x);
```

```
    while (x != 0) {
      if (x >= 0) {
        val.pos[x] = val.pos[x] + 1;
      } else if (x < 0) {
        val.neg[-x] = val.neg[-x] + 1;
      }
      read(x);
    }
  }
}
```

Sintaxe

```
Program    = "program" ident {ConstDecl | VarDecl | ClassDecl}
             "{" {MethodDecl} "}".
ConstDecl  = "final" Type ident "=" (number | charConst) ";".
VarDecl    = Type ident {"," ident } ";".
ClassDecl  = "class" ident "{" {VarDecl} "}".
MethodDecl = (Type | "void") ident "(" [FormPars] ")" {VarDecl}
Block.
FormPars   = Type ident {"," Type ident}.
Type       = ident ["[" "]"].
Block      = "{" {Statement} "}".
Statement  = Designator ("=" Expr | ActPars) ";"
           | "if" "(" Condition ")" Statement ["else" Statement]
           | "while" "(" Condition ")" Statement
           | "return" [Expr] ";"
           | "read" "(" Designator ")" ";"
           | "print" "(" Expr ["," number] ")" ";"
           | Block
           | ";".
ActPars    = "(" [ Expr {"," Expr} ] ")".
Condition  = Expr Relop Expr.
Relop      = "==" | "!=" | ">" | ">=" | "<" | "<=".
Expr       = ["-"] Term {Addop Term}.
Term       = Factor {Mulop Factor}.
Factor     = Designator [ActPars]
           | number
           | charConst
           | "new" ident ["[" Expr "]"]
           | "(" Expr ")".
Designator = ident {"." ident | "[" Expr "]"}.
Addop      = "+" | "-".
Mulop =      "*" | "/" | "%".
```

Estrutura léxica

```
Classes de caracteres: letter     = 'a'..'z' | 'A'..'Z'.
                       digit      = '0'..'9'.
                       whiteSpace = ' ' | '\t' | '\r' | '\n'.
Classes dos terminais: ident      = letter {letter | digit}.
                       number     = digit {digit}.
                       charConst  = "'" char "'". // including
'\r', '\t', '\n'

Palavras reservadas:  program class
                      if     else    while   read    print  return
                      void    final   new
Operadores:          +       -       *       /       %
                     ==      !=      >       >=      <      <=
                     (       )       [       ]       {      }
                             ;       ,       .
                     =
Comentários:         // to the end of line
```