

Laboratório de Linguagens de Programação  
Prof. Andrei Rimsa Álvares

# Trabalho Prático I

## 1. Objetivo

O objetivo desse trabalho é desenvolver um interpretador para uma nova linguagem de programação que é um subconjunto de PHP, chamada de miniPHP. Ela possui suporte a tipos inteiros, strings e arranjos indexados. Além disso, suporta o conceito de variável-variável e permite iterar sobre arranjos através do comando `foreach`.

## 2. Contextualização

A seguir é dado um exemplo de utilização da linguagem que usa arranjos indexado por strings, `foreach` para navegação no mesmo e variável-variável:

---

```
/* Best Megadeth band formation */
$band = array("David Mustaine" => "guitars",
              "David Ellefson" => "bass",
              "Martin Friedman" => "guitars",
              "Nick Menza" => "drums");

echo "Members:\n";
foreach ($band as $member => $instrument) {
    echo " " . $member . ": " . $instrument . "\n";
    $$instrument++;
}

echo "Stats:\n";
echo " Guitars: " . $guitars . "\n";
echo " Bass: " . $bass . "\n";
echo " Drums: " . $drums . "\n";
```

---

*mega.mphp*

---

A linguagem possui escopo global para as variáveis. Operações aritméticas devem converter strings para inteiros (usar 0 se não for possível). Concatenação deve converter inteiros para string. Variáveis usadas sem ter valor previamente assinalado deve-se usar string vazia ("" ) por padrão. A linguagem possui somente comentários de múltiplas linhas onde são ignorados qualquer sequência de caracteres entre `/*` e `*/`. A linguagem possui as seguintes características:

### 1) Comandos:

- if**: executar comandos baseado em expressões condicionais.
- while**: repetir comandos enquanto a expressão for verdadeira.
- foreach**: iterar sobre um arranjo.
- echo**: imprimir na tela.

### 2) Constantes:

- Inteiro**: sequência de dígitos.
- String**: uma sequência de caracteres entre aspas duplas.
- Lógico**: operações de comparações que obtém um valor lógico (não podem ser armazenados).



Laboratório de Linguagens de Programação  
Prof. Andrei Rimsa Álvares

### 3) Valores:

- Variáveis (ex.: **\$x**).
- Variável-variável (ex.: **\$\$x**, **\$\$\$y**).
- Constantes (inteiro e string somente).
- read**: ler do teclado.

### 4) Operadores:

- Inteiro**: **+** (adição), **-** (subtração), **\*** (multiplicação), **/** (divisão), **%** (resto inteiro).
- String**: **.** (concatenação).
- Lógico**: **==** (igual), **!=** (diferença), **<** (menor), **>** (maior), **<=** (menor igual), **>=** (maior igual), **!** (negação).
- Conector**: **and** (E lógico), **or** (OU lógico).
- Atribuição**: **=** (atribuir), **+=** (incrementar e atribuir), **-=** (decrementar e atribuir), **\*=** (multiplicar e atribuir), **/=** (dividir e atribuir), **%=** (resto da divisão e atribuir), **.=** (concatenar e atribuir).
- Pré e pós-fixados**: **++** (incremento), **--** (decremento)

## 3. Gramática

A gramática da linguagem *miniPHP* é dada a seguir no formato de Backus-Naur estendida (EBNF):

```
<code>      ::= { <statement> }
<statement> ::= <if> | <while> | <foreach> | <echo> | <assign>

<if>        ::= if '(' <boolexpr> ')' '{' <code> '}'
              { elseif '(' <boolexpr> ')' '{' <code> '}' } [ else '{' <code> '}' ]
<while>     ::= while '(' <boolexpr> ')' '{' <code> '}'
<foreach>   ::= foreach '(' <expr> as <var> [ '=' <var> ] ')' '{' <code> '}'
<echo>      ::= echo <expr> ';'
<assign>    ::= <value> [ ('=' | '+=' | '-=' | '.=' | '*=' | '/=' | '%=') <expr> ] ';'

<boolexpr>  ::= [ '!' ] <cmpexpr> [ (and | or) <boolexpr> ]
<cmpexpr>   ::= <expr> ('==' | '!=' | '<' | '>' | '<=' | '>=') <expr>

<expr>      ::= <term> { ('+' | '-' | '.') <term> }
<term>      ::= <factor> { ('*' | '/' | '%') <factor> }
<factor>    ::= <number> | <string> | <array> | <read> | <value>

<array>     ::= array '(' [ <expr> '=' <expr> { ',' <expr> '=' <expr> } ] ')'
<read>      ::= read <expr>
<value>     ::= [ ('++' | '--') ] <access> | <access> [ ('++' | '--') ]
<access>    ::= ( <varvar> | '(' <expr> ')' ) [ '[' <expr> ']' ]
<varvar>    ::= '$' <varvar> | <var>
```



Laboratório de Linguagens de Programação

Prof. Andrei Rimsa Álvares

## 4. Instruções

Deve ser desenvolvido um interpretador em linha de comando que recebe um programa-fonte na linguagem *miniPHP* como argumento e executa os comandos especificados pelo programa. Por exemplo, para o programa *mega.mphp* deve-se produzir uma saída semelhante a:

```
$ ./mphpi
Usage: ./mphpi [miniPHP File]
$ ./mphpi mega.mphp
Members:
  David Mustaine: guitars
  David Ellefson: bass
  Martin Friedman: guitars
  Nick Menza: drums
Stats:
  Guitars: 2
  Bass: 1
  Drums: 1
```

O programa deverá abortar sua execução, em caso de qualquer erro léxico, sintático ou semântico, indicando uma mensagem de erro. As mensagens são padronizadas indicando o número da linha (2 dígitos) onde ocorreram:

Tipo de Erro	Mensagem
Léxico	Lexema inválido [ <i>lexema</i> ]
	Fim de arquivo inesperado
Sintático	Lexema não esperado [ <i>lexema</i> ]
	Fim de arquivo inesperado
Semântico	Operação inválida

Exemplo de mensagem de erro:

```
$ ./mphpi erro.mphp
03: Lexema não esperado [;]
```

## 5. Avaliação

O trabalho deve ser feito em grupo de até dois alunos, sendo esse limite superior estrito. O trabalho será avaliado em 15 pontos, onde essa nota será multiplicada por um fator entre 0.0 e 1.0 para compor a nota de cada aluno individualmente. Esse fator poderá estar condicionado a apresentações presenciais a critério do professor.

Trabalhos copiados, parcialmente ou integralmente, serão avaliados com nota **ZERO**, sem direito a contestação. Você é responsável pela segurança de seu código, não podendo alegar que outro grupo o utilizou sem o seu consentimento.

Laboratório de Linguagens de Programação  
Prof. Andrei Rimsa Álvares

## **6. Submissão**

O trabalho deverá ser submetido até as 23:59 do dia 19/10/2020 (segunda-feira) via sistema acadêmico em pasta específica. Não serão aceitos, em hipótese alguma, trabalhos enviados por e-mail ou por quaisquer outras fontes.



