Implementação de um Interpretador em Haskell (Projeto)

A nota projeto será distribuída da seguinte forma: 40% para a documentação e 60% para a implementação do interpretador em Haskell.

Documentação

A sintaxe abstrata da linguagem imperativa encontra-se no final desse documento. Com base nessa sintaxe, o grupo deve escrever, em linguagem natural, uma semântica para a linguagem. Após essa etapa, deve-se acrescentar, na documentação, pelo menos três programas que use vários de seus recursos e que esteja dentro da sintaxe definida.

Outro ponto importante diz respeito à clareza, à organização e à precisão do documento, pois o mesmo deve estar de tal forma que possa ser passado para um programador, para a implementação de um interpretador/compilador da linguagem.

Implementação

Nesse ponto o grupo deve implementar um interpretador para a linguagem em Haskell utilizando a ferramenta Hugs. Os programas que foram descritos na documentação devem ser interpretados pelo interpretador construído. É importante que o código fonte esteja devidamente comentado. Use de forma explícita conceitos com valores, bindings, memória e abstração na estruturação do interpretador.

Características da linguagem

- É uma linguagem imperativa;
- Possui procedimentos parametrizados e recursivos;
- Possui ponteiros;
- O corpo de um procedimento é um comando e a chamada de um procedimento também;
- Procedimentos não possuem status de valor (retorno);
- Expressões podem resultar em valores inteiros, booleanos, strings ou endereços;
- Um programa é um comando;
- Variáveis e procedimentos devem ser declarados antes do uso;
- É inspirada em Pascal e C;

Sintaxe Abstrata da Linguagem

```
Programa        ::= Comando

Comando         ::= ComandoDeclaracao
                  | Atribuicao
                  | While
                  | For
                  | IfThenElse
                  | RepeatUntil
                  | IO
                  | Skip
                  | ChamadaProcedimento
                  | Comando ";" Comando

Skip            ::= "skip"

Atribuicao      ::= Id ":=" Expressao

Expressao       ::= Valor
                  | ExpUnaria
                  | ExpBinaria
                  | Id
                  | "&"Id
                  | "*"Id

Valor           ::= ValorInteiro
                  | ValorBooleano
                  | ValorString

ExpUnaria       ::= "-" Expressao
                  | "!" Expressao
                  | "#" Expressao

ExpBinaria      ::= Expressao "+"  Expressao
                  | Expressao "-"  Expressao
                  | Expressao "&&" Expressao
                  | Expressao "||" Expressao
                  | Expressao "==" Expressao
                  | Expressao "++" Expressao


ComandoDeclaracao   ::= "{" Declaracao ";" Comando "}"

Declaracao      ::= DeclaracaoVariavel
                  | DeclaracaoProcedimento
                  | Declaracao "," Declaracao

DeclaracaoVariavel    ::= "var" Id "=" Expressao
                  | "pointer" Id "=" "^"Tipo

DeclaracaoProcedimento    ::= "proc" Id "(" [ ListaDeclaracaoParametro ] ")" "{" Comando "}"

ListaDeclaracaoParametro  ::= Id ["^"]Tipo
                  | ListaDeclaracaoParametro "," ListaDeclaracaoParametro

Tipo            ::= "string"
                  | "int"
                  | "boolean"
```

```
While              ::= "while" Expressao "do" Comando

For                ::= "for" Atribuicao "to" ValorInteiro "do" Comando

IfThenElse         ::= "if" Expressao "then" Comando "else" Comando

RepeatUntil        ::= “repeat” Comando “until” Expressao

IO                 ::= "write" "(" Expressao ")"
                     | "read" "(" Id ")"

ChamadaProcedimento    ::= "call" Id "(" ListaExpressao ")"

ListaExpressao ::= Expressao
                     | Expressao "," ListaExpressao
```