Centro Universitário de Belo Horizonte – UniBH
UC: Teoria da Computação e Compiladores
Professor: Gustavo Alves Fernandes

Exercício Prático II – Análise Sintática

Nesta etapa, você deverá implementar um analisador sintático para a linguagem ***Mini-Cond***, cuja descrição encontra-se nas próximas páginas. **Atenção**: A gramática foi corrigida para que seja possível a implementação de um parser *top-down*. A implementação segue a gramática corrigida.

Seu analisador sintático deverá ser implementado conforme visto em sala de aula. Ele deverá ser um analisador de uma única passada, dessa forma, ele deverá interagir com o analisador léxico para obter os *tokens* do arquivo-fonte. Você deve implementar seu analisador sintático utilizando o algoritmo de Parser Preditivo Recursivo. O parser está parcialmente implementado, logo, você deve complementar as partes faltantes.

O analisador sintático deverá reportar o primeiro erro ocorrido no programa-fonte e abortar a compilação. O analisador deverá informar qual o erro encontrado e sua localização no arquivo-fonte.

O que entregar?

Você deverá entregar nesta etapa:

- Um vídeo comentando a sua implementação e mostrando testes comprovando a implementação.
- Submeter seu código fonte para apreciação do professor.

Para avaliar a correção, o programa deverá exibir os *tokens* reconhecidos e o local de sua ocorrência, os *tokens* armazenados na tabela de símbolos, bem como os erros léxicos e sintáticos gerados, se acontecerem.

Regras:

- O trabalho poderá ser realizado individualmente, em dupla ou trio.
- Não é permitido o uso de ferramentas para geração do analisador sintático.
- A implementação poderá ser realizada em uma das linguagens C, C++, C#, Java, Ruby, Python, Javascript ou PHP.
- Trabalhos total ou parcialmente iguais receberão avaliação nula.
- Se o analisador sintático não compilar ou apresentar erros de execução durante a fase de teste realizada pelo professor, a avaliação será nula.
- Após a data definida para entrega, nenhum trabalho será recebido.

A linguagem *Mini-Cond*

```
PROGRAMA  → CMD EOF
CMD       → if E then { CMD } |
            if E then { CMD } else { CMD } |
            print T; | ATRIB CMD
ATRIB     → id = T;
E         → T OP T | T
OP        → < | <= | == | != | > | >=
T         → id | num
```

A linguagem *Mini-Cond (corrigida)*

```
PROGRAMA  → CMD EOF
CMD       → if E then { CMD } CMD'|
            print T; | ATRIB CMD
CMD'      → else { CMD } | Ɛ
ATRIB     → id = T;
E         → T E'
E'        → OP T | Ɛ
OP        → < | <= | == | != | > | >=
T         → id | num
```

Padrões:

```
id: [A – Za – z]([A – Za – z] | [0 – 9])*
num: [0 – 9]+
```

Os demais nomes de token são apresentados no código.

Atenção: EOF significa “Fim de Arquivo”. A identificação de fim de arquivo já está implementada.

Exemplo: Programa válido para essa linguagem:

```
var1 = 10;
var2 = 10;
if var1 != 10 then {
     print var1;
}
else {
     if 10 == var2 then {
          print var2;
     }
     else {
          print 10;
     }
}
```