unibh

CENTRO UNIVERSITTÁRIO DE BELO HORIZONTE

Trabalho Prático

Belo Horizonte – 2018

Stella Oliveira Moreira
CIC4AN-ESA

Índice

Stella Oliveira Moreira
CIC4AN-ESA

Introdução

Segunda parte do trabalho prático de Compiladores, foi implementada o analisador sintático cuja utiliza a regra descendente *(top-down)*, é o processo de se determinar se uma cadeia de símbolos léxicos pode ser gerada por uma gramática. Ela transforma um texto na entrada em uma estrutura de dados, em geral uma árvore, o que é conveniente para o processamento posterior e captura a hierarquia implícita desta entrada. Através da análise léxica é obtida o grupo de *tokens*, para que o analisador sintático use em conjunto de regras para construir uma árvore sintática da estrutura.

A tarefa do analisador é determinar se a entrada de dados pode ser derivada de um símbolo inicial com as regras de uma gramática formal.

A regra que utilizamos *(top-down)*, o analisador inicia com um símbolo inicial e tenta transforma-lo na entrada de dados. De maneira intuitiva o analisador inicia dos maiores elementos e a quebra em elementos menores.

Stella Oliveira Moreira
CIC4AN-ESA

Desenvolvimento

O desenvolvimento dessa segunda parte, ainda foi implementada em C#, foram adicionadas novas duas classes no projeto (Program e AnalisadorSint)

- A Classe Program:

É responsável por chamar os métodos iniciando o programa, abrindo o arquivo de leitura e finalizando o mesmo.

- A Classe AnalisadorSint:

É onde ocorre o tratamento da tabela preditiva, e todos os outros processos como ir para o próximo token, o eat e até mesmo o skip, entre outros métodos que serão explicados no relatório.

Desempenho Global:

Houve diversas complicações no desenvolvimento do trabalho, muitas pesquisas sobre o tema relacionado e não houve 100% de cumprimento da tarefa proposta, na próxima entrega correções serão realizadas.

Fases de Elaboração:

1. Pesquisa do método
2. Esquematização
3. Desenvolvimento das novas classes
4. Implementação
5. Teste

ID - Etapa	Descrição de Etapa	ID próxima etapa	Tipo de forma	Data de Início	Data de Término
P700	Estudo do método	P800	Início	30/05/2018	01/06/2018
P800	Esquematização	P900	Esquematização	02/06/2018	02/06/2018
P900	Desenvolvimento das Classes	P1000	Documento	03/06/2018	03/06/2018
P1000	Implementação	P2000	Decisão	03/06/2018	03/06/2018
P3000	Teste	P4000	Subprocesso	03/06/2018	03/06/2018
P4000	Termino		Termino	03/06/2018	

Stella Oliveira Moreira
CIC4AN-ESA

Explicação do Código

Toda estrutura das classes foi desenvolvida com base no exemplo disponibilizado pelo professor, o código implementado visa realizar o processo de *(top-down)*, e segue com o trabalho anterior.

No uso do código é necessário especificar o caminho do arquivo na classe “AnalisadorLex.cs” e “Program” na região especificada como (leitura de arquivo).

```
#region Leitura de Arquivo
public AnalisadorLex(String arquivoEntrada)
{
    tabelaSimbolos = new TabelaSimbolos(nLinhas, nColunas);
    try
    {
        arquivo = new FileStream("C:\\Users\\stell\\Documents\\Arquivos\\teste.txt", FileMode.Open);
    }
    catch (IOException excep)
    {
        Console.WriteLine("Erro de abertura do arquivo " + arquivoEntrada + "\nException: " + excep);
        Environment.Exit(1);
        Console.ReadLine();
    }
    catch (Exception excep)
    {
        Console.WriteLine("Erro do programa ou falha da tabela de símbolos \nException: " + excep);
        Environment.Exit(2);
    }
}
#endregion
```

Class – AnalisadorLex.cs

```
public static void Main(string[] args)
{
    AnalisadorLex lexico = new AnalisadorLex("teste.txt");
    AnalisadorSint sintatico = new AnalisadorSint(lexico);
```

Class – Program.cs

Foram realizados testes para que a implementação seja validada, foram realizadas no total de 6 testes, onde três não possuem erros, e outros três onde existem erros de sintáticos.

Testes com sucesso:

```
[DEBUG]: <COMENTARIO , "
 ">
[DEBUG]: <KW_PROGRAM , "program ">
[DEBUG]: <ID , "metodo ">
[DEBUG]: <SMB_OBC , "{ ">
[DEBUG]: <KW_IF , "if ">
[DEBUG]: <SMB_OPA , "( ">
[DEBUG]: <ID , "numero ">
[DEBUG]: <OP_EQ , "== ">
[DEBUG]: <NUM_CONST , "2 ">
[DEBUG]: <SMB_CPA , ") ">
[DEBUG]: <SMB_SEM , "; ">
[DEBUG]: <SMB_CBC , "} ">
[DEBUG]: <SMB_SEM , "; ">

Arquivo Finalizado
```

```
[DEBUG]: <KW_PROGRAM , "program ">
[DEBUG]: <ID , "metodo ">
[DEBUG]: <KW_NUM , "num ">
[DEBUG]: <ID , "teste ">
[DEBUG]: <SMB_COM , ", ">
[DEBUG]: <ID , "numero1 ">
[DEBUG]: <SMB_SEM , "; ">
[DEBUG]: <KW_CHAR , "char ">
[DEBUG]: <ID , "varia123 ">
[DEBUG]: <SMB_SEM , "; ">
[DEBUG]: <SMB_OBC , "{ ">
[DEBUG]: <SMB_CBC , "} ">

Arquivo Finalizado
```

```
[DEBUG]: <COMENTARIO , "
 ">
[DEBUG]: <KW_PROGRAM , "program ">
[DEBUG]: <ID , "metodo ">
[DEBUG]: <SMB_OBC , "{ ">
[DEBUG]: <KW_WRITE , "write ">
[DEBUG]: <STRING , "teste string ">
[DEBUG]: <SMB_SEM , "; ">
[DEBUG]: <SMB_CBC , "} ">

Arquivo Finalizado
```

Stella Oliveira Moreira
CIC4AN-ESA

Testes sem sucesso:

```
[ERRO SINTATICO]:       Linha: 1 Coluna: 1
Esperado "PROGRAM", encontrado: public


[ERRO SINTATICO]:       Linha: 1 Coluna: 1
[MODO PÂNICO] Esperado "EOF", encontrado: public

[DEBUG]: <ID , "public ">

[ERRO SINTATICO]:       Linha: 1 Coluna: 2
[MODO PÂNICO] Esperado "EOF", encontrado: public

[DEBUG]: <ID , "static ">

[ERRO SINTATICO]:       Linha: 1 Coluna: 3
[MODO PÂNICO] Esperado "EOF", encontrado: public

[DEBUG]: <ID , "void ">

[ERRO SINTATICO]:       Linha: 1 Coluna: 4
[MODO PÂNICO] Esperado "EOF", encontrado: public

[DEBUG]: <ID , "main ">

[ERRO SINTATICO]:       Linha: 1 Coluna: 5
[MODO PÂNICO] Esperado "EOF", encontrado: public

[DEBUG]: <SMB_OPA , "( ">

[ERRO SINTATICO]:       Linha: 1 Coluna: 6
[MODO PÂNICO] Esperado "EOF", encontrado: public

[DEBUG]: <SMB_CPA , ") ">


                Arquivo Finalizado
```

```
[DEBUG]: <KW_PROGRAM , "program ">
[DEBUG]: <ID , "p1 ">
[DEBUG]: <SMB_OBC , "{ ">
[DEBUG]: <ID , "variavel ">

[ERRO SINTATICO]:       Linha: 2 Coluna: 2
Esperado "!=", encontrado: ,


[ERRO SINTATICO]:       Linha: 2 Coluna: 2
Esperado "ID ou NUM_CONST ou CHAR_CONST ou ( ou NOT", encontrado: ,


[ERRO SINTATICO]:       Linha: 2 Coluna: 2
[MODO PÂNICO] Esperado "; ou == ou > ou >= ou < ou <= ou != ou )", encontrado: ,

[DEBUG]: <SMB_COM , ", ">

[ERRO SINTATICO]:       Linha: 3 Coluna: 1
[MODO PÂNICO] Esperado "; ou == ou > ou >= ou < ou <= ou != ou )", encontrado: ,

[DEBUG]: <SMB_CBC , "} ">

[ERRO SINTATICO]:       Linha: 3 Coluna: 1
Esperado "ID ou IF ou WHILE ou READ ou WRITE", encontrado: EOF


[ERRO SINTATICO]:       Linha: 3 Coluna: 1
Esperado ";", encontrado: EOF


                Arquivo Finalizado
```

```
[DEBUG]: <KW_PROGRAM , "program ">
[DEBUG]: <ID , "metodo ">
[DEBUG]: <SMB_OBC , "{ ">
[DEBUG]: <KW_IF , "if ">
[DEBUG]: <SMB_OPA , "( ">
[DEBUG]: <ID , "variavel ">

[ERRO SINTATICO]:       Linha: 2 Coluna: 4
Esperado "* ou / ou AND", encontrado 1


[ERRO SINTATICO]:       Linha: 2 Coluna: 4
[MODO PÂNICO] Esperado "+ ou - ou OR ou ; ou == ou > ou >= ou < ou <= ou != ou )", encontrado 1

[DEBUG]: <NUM_CONST , "1 ">
[DEBUG]: <SMB_CPA , ") ">

[ERRO SINTATICO]:       Linha: 2 Coluna: 6
[MODO PÂNICO] Esperado: ")", encontrado: {

[DEBUG]: <SMB_OBC , "{ ">

[ERRO SINTATICO]:       Linha: 3 Coluna: 1
[MODO PÂNICO] Esperado: ")", encontrado: {

[DEBUG]: <SMB_CBC , "} ">

[ERRO SINTATICO]:       Linha: 4 Coluna: 1
[MODO PÂNICO] Esperado: ")", encontrado: {

[DEBUG]: <SMB_CBC , "} ">

[ERRO SINTATICO]:       Linha: 4 Coluna: 1
Esperado ")", encontrado: EOF


[ERRO SINTATICO]:       Linha: 4 Coluna: 1
Esperado "ELSE ou , ou ;", encontrado: EOF


[ERRO SINTATICO]:       Linha: 4 Coluna: 1
Esperado "ID ou IF ou WHILE ou READ ou WRITE", encontrado: EOF


[ERRO SINTATICO]:       Linha: 4 Coluna: 1
Esperado ";", encontrado: EOF


                Arquivo Finalizado
```

Stella Oliveira Moreira
CIC4AN-ESA

unibh

Classe AnaliseSint

```
public AnalisadorSint(AnalisadorLex lexer)
{
    this.lexer = lexer;
    token = lexer.proximoToken();
    sincroniza = new List<EnumTab>();
    if (token.getClasse() == EnumTab.COMENTARIO)
    {
        proximoToken();
    }else if (!eat(EnumTab.KW_PROGRAM))
    {
        erroSintatico("Esperado \"PROGRAM\", encontrado: " + token.getLexema());
        sincroniza.Add(EnumTab.STRING);
        sincronizaToken("[MODO PÂNICO] Esperado \"EOF\", encontrado: " + token.getLexema());
    }
}
```

- Construtor do método da classe AnalisadorSint

Responsável por iniciar o processo de análise na tabela preditiva.

```
public void erroSintatico(String mensagem)
{
    Console.WriteLine("\n[ERRO SINTATICO]: " + token.LinhaPercorrida());
    Console.WriteLine(mensagem + "\n");
}
```

- Método que visa sinalizar o erro sintático.

```
public void proximoToken()
{
    Console.WriteLine("[DEBUG]: " + token.ToString());
    token = lexer.proximoToken();
}
```

- Método responsável por imprimir o token lido corretamente e ir para o próximo token.

```
#region Implementação do Skip and Eat
public void skip(String mensagem)
{
    erroSintatico(mensagem);
    proximoToken();
}

public Boolean eat(EnumTab tab)
{
    if (token.getClasse() == tab)
    {
        proximoToken();
        return true;
    }
    else
    {
        return false;
    }
}

#endregion
```

- Métodos que fazem parte do formato (top-down) visam seguir em frente com base no erro sintático encontrado ou recuar.

Stella Oliveira Moreira
CIC4AN-ESA

```
public void sincronizaToken(String mensagem)
{
    Boolean casaToken = false;

    while (!casaToken && token.getClasse() != EnumTab.EOF)
    {
        if (sincroniza.Contains(token.getClasse()))
        {
            casaToken = true;
        }
        else
        {
            skip(mensagem);
        }
    }
    sincroniza.Clear();
}
```

- Método responsável por pegar o token atual e fazer com que ele seja consumido não interferindo no compilador, ignorando o token anterior caso erro.

```
#region Tabela Preditiva
//prog -> "program" "id" body |Fol| $
public void prog()
{
    if (token.getClasse() == EnumTab.KW_PROGRAM)
    {
        program();
    }
    proximoToken();
    if (!eat(EnumTab.ID))
    {
        body();
    }
}
```

- Parte inicial do corpo da análise da tabela preditiva, nessa região é feito todo o processo de análise dos tokens sendo assim informada se o processo ocorrerá conforme passado ou não.

Classe Program

```
public class Program
{
    public static void Main(string[] args)
    {
        AnalisadorLex lexico = new AnalisadorLex("teste.txt");
        AnalisadorSint sintatico = new AnalisadorSint(lexico);

        //Inicia o processo
        sintatico.prog();
        sintatico.fecharArquivo();

        // lexico.imprimeTabelaSimbolos();
        Versão 01

        Console.WriteLine("\n\n\t\tArquivo Finalizado");
        Console.ReadLine();
    }
}
```

Classe responsável por coordenar o funcionamento do programa.

Stella Oliveira Moreira
CIC4AN-ESA

Conclusão

Novamente a linguagem C# atendeu as expectativas necessárias para o desenvolvimento do projeto.

O desenvolvimento dessa segunda parte foi totalmente com base no conteúdo passado pelo professor, logo não houve muita complexidade.

Stella Oliveira Moreira
CIC4AN-ESA

unibh

Referências bibliográficas

1. http://www.di.uern.br/epoca2010/artigos/78448_1.pdf
2. https://pt.wikipedia.org/wiki/Abordagem_top-down_e_bottom-up
3. https://www.dcce.ibilce.unesp.br/~aleardo/cursos/compila/cap03.pdf

Stella Oliveira Moreira
CIC4AN-ESA

GITHUB DESENVOLVIMENTO DO TRABALHO

https://github.com/stellaoliveirabertt/compilador

Stella Oliveira Moreira
CIC4AN-ESA