

TRABALHO FINAL – parte 4: implementação do analisador semântico e do gerador de código

Implementar as ações semânticas que constituem o **analisador semântico** e o **gerador de código**, gerando o código objeto, de acordo com o esquema de tradução ([esquema de tradução completo.pdf](#)) disponibilizado no AVA (aba **COMPILADOR**). Deve-se também implementar **tratamento de erros** semânticos, a partir das verificações semânticas especificadas no esquema de tradução.

Entrada	– A entrada é um conjunto de caracteres, isto é, o programa fonte do editor do compilador.
Saída	<p>– Caso o botão compilar seja pressionado, a ação deve ser: executar as análises léxica, sintática e semântica do programa fonte. Um programa pode ser compilado com sucesso ou apresentar erros. Em cada uma das situações a saída deve ser:</p> <p><u>1ª situação</u>: programa compilado com sucesso</p> <ul style="list-style-type: none">✓ mensagem (<i>programa compilado com sucesso</i>), na área reservada para mensagens, indicando que o programa não apresenta erros.✓ código objeto em <i>MicroSoft Intermediate Language</i>, corresponde ao programa fonte compilado. O código objeto deve ser gerado na <u>MESMA PASTA</u> do programa fonte que está sendo compilado e deve estar em um arquivo texto com extensão <code>.il</code> e nome igual ao nome do arquivo que contém o programa compilado. Assim, por exemplo, se o programa compilado for <code>teste01.txt</code> da pasta <code>c:\temp\</code>, deve ser gerado na pasta <code>c:\temp\</code> o arquivo <code>teste01.il</code>. Considere que só serão compilados arquivos já salvos. <p><u>2ª situação</u>: programa apresenta erros</p> <ul style="list-style-type: none">✓ mensagem, na área reservada para mensagens, indicando que o programa apresenta erro. O erro pode ser léxico, sintático ou semântico, cujas mensagens devem ser conforme descrito abaixo.

OBSERVAÇÕES:

- O tipo do analisador sintático a ser gerado é **LL (1)**.
- As mensagens para os **erros léxicos** devem estar conforme especificado na parte 2 do trabalho final, com as devidas correções, podendo ter alteração na nota do analisador léxico, segundo observado e indicado na avaliação do analisador léxico.
- As mensagens para os **erros sintáticos** devem ser conforme especificado na parte 3 do trabalho final, com as devidas correções, podendo ter alteração na nota do analisador sintático, segundo observado e indicado na avaliação do analisador sintático.
- As mensagens para os **erros semânticos** devem indicar a linha onde ocorreu o erro e a descrição do erro, conforme a especificação das verificações semânticas. As mensagens de erro devem ser geradas durante a execução das ações semânticas. Ao ser emitida uma mensagem de erro semântico, o processo de análise deve ser encerrado.
- No **esquema de tradução** disponibilizado, a gramática, possui a numeração das ações semânticas. A equipe deve colocar a numeração das ações semânticas na gramática usada para a implementação do analisador sintático. Observa-se que trabalhos desenvolvidos usando uma gramática diferente daquela utilizada pela equipe na implementação do analisador sintático receberão nota 0.0 (zero). Se a equipe achar necessário, pode incluir outras ações semânticas.
- Uma vez que a gramática esteja alterada e as ações semânticas corretamente colocadas, deve-se gerar novamente os analisadores léxico, sintático e semântico para refletir na implementação as alterações feitas. Observa-se que, em geral, o único código alterado pelo GALS é o das constantes (em Java - `ScannerConstants.java`, `ParserConstants.java`, `Constants.java`). As ações semânticas são executadas a partir do método `executeAction` (da classe que implementa o analisador semântico). Esse método recebe como parâmetros (do analisador sintático) o número da ação semântica reconhecida (`action`) e o `token` corrente (`token`).
- O **código objeto** para o programa fonte `nome_do_arquivo.txt` deve ser gerado no formato especificado.
- A implementação do analisador semântico e do gerador de código, juntamente com os analisadores léxico e sintático e a interface do compilador, deve ser disponibilizada no **AVA** na tarefa “**parte 4 - analisador semântico e gerador de código**”, aba **COMPILADOR**. Deve ser disponibilizado um **arquivo compactado** (com o nome: `compilador`, seguido do número da equipe), contendo: o projeto completo, incluindo **código fonte, executável** (ou `.jar`), **arquivo com as especificações léxica e sintática** e a numeração das ações semânticas (no GALS, arquivo com extensão `.gals`) e demais recursos. Basta um integrante da equipe postar o trabalho.
- Para que o analisador semântico e o gerador de código sejam **avaliados**, devem ser atendidos os seguintes requisitos:
 - (1) Todos os arquivos solicitados no item anterior devem ser postados no AVA, sendo que o **executável** (ou `.jar`) deve ser executado sem erro, caso contrário será atribuída nota 0.0 (zero) à parte 4.
 - (2) O código objeto deve ser gerado em um arquivo texto com extensão `.il` na MESMA PASTA do programa fonte que está sendo compilado, caso contrário será atribuída nota 0.0 (zero) à parte 4.

- Na avaliação do analisador semântico e do gerador de código serão levadas em consideração: a correta adequação da gramática com a inclusão das ações semânticas; a correta implementação das ações semânticas (as ações NÃO terão peso igual na avaliação) e das verificações semânticas; a qualidade das mensagens de erro, conforme descrito acima; o uso apropriado de ferramentas para construção de compiladores. Observa-se que esta implementação corresponderá a 80% da nota da **parte 4 - analisador semântico e gerador de código**.

DATA LIMITE PARA ENTREGA: até às 23h do dia 28/11/2024 (quinta-feira). Não serão aceitos trabalhos após data e hora determinadas.

EXEMPLOS DE ENTRADA / SAÍDA

EXEMPLO 1: com erro léxico

ENTRADA		SAÍDA (na área de mensagens)
linha	1 main 2 i_lado; 3 read ("digite um valor para lado: ", i_lado; 4 i_area = i_lado * i_lado; 5 writeln (i_area); 6 end	Erro na linha 3 - constante_string inválida

EXEMPLO 2: com erro sintático

ENTRADA		SAÍDA (na área de mensagens)
linha	1 main 2 i_lado; 3 read ("digite um valor para lado: ", i_lado; 4 i_area = i_lado * i_lado; 5 writeln (i_area); 6 end	Erro na linha 3 - encontrado ; esperado ,)

EXEMPLO 3: com erro semântico

ENTRADA		SAÍDA (na área de mensagens)
linha	1 main 2 i_lado; 3 read ("digite um valor para lado: ", i_lado); 4 i_area = i_lado * i_lado; 5 writeln (i_area); 6 end	Erro na linha 4 - i_area não declarado

EXEMPLO 4: sem erro – mensagem na **área de mensagens** e arquivo `.il` gerado na MESMA pasta do programa fonte

ENTRADA		SAÍDA (na área de mensagens)
linha	1 main 2 i_lado, i_area; 3 read ("digite um valor para lado: ", i_lado); 4 i_area = i_lado * i_lado; 5 writeln (i_area); 6 end	programa compilado com sucesso

programa objeto (teste_01.il)	
// cabeçalho .assembly extern mscorlib {} .assembly _codigo_objeto{} .module _codigo_objeto.exe .class public UNICA{ .method static public void _principal() { .entrypoint // cabeçalho .locals(int64 i_lado) .locals(int64 i_area) ldstr "digite um valor para lado: " call void [mscorlib]System.Console::Write(string) call string [mscorlib]System.Console::ReadLine() call int64 [mscorlib]System.Int64::Parse(string) stloc i_lado	ldloc i_lado conv.r8 ldloc i_lado conv.r8 mul conv.i8 stloc i_area ldloc i_area conv.r8 conv.i8 call void [mscorlib]System.Console::WriteLine(int64) // fim de programa ret } }