

# 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 <b>compilar</b> 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"><li>✓ <b>mensagem</b> (<i>programa compilado com sucesso</i>), na área reservada para mensagens, indicando que o programa não apresenta erros.</li><li>✓ <b>código objeto</b> 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 <u>.il</u> 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 o programa fonte compilado está previamente salvo em um arquivo.</li></ul> <p><u>2ª situação</u>: programa apresenta erros</p> <ul style="list-style-type: none"><li>✓ <b>mensagem</b>, na área reservada para mensagens, indicando que o programa apresenta erro. O erro pode ser <b>léxico</b>, <b>sintático</b> ou <b>semântico</b>, cujas mensagens devem ser conforme descrito abaixo.</li></ul>

## 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 ([do esquema de tradução completo.pdf](#)). As mensagens de erro devem ser geradas durante a execução das ações semânticas. Assim, tem-se alguns exemplos:  
    Erro na linha 10 – tipos incompatíveis em expressão lógica  
    Erro na linha 15 – tipos incompatíveis em expressão aritmética  
Ao ser emitida uma mensagem de erro semântico, o processo de análise deve ser encerrado.
- No **esquema de tradução** disponibilizado ([esquema de tradução completo.pdf](#)), 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 e pode ser validado utilizando `ilasm nome_do_arquivo.il` e, em seguida, executando o arquivo `nome_do_arquivo.exe`
- 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 aba COMPILADOR / analisador semântico), na **“pasta” da sua equipe**. Deve ser disponibilizado **um arquivo compactado** (com o nome: `compilador`, seguido do número da equipe), contendo: o projeto completo, incluindo o **código fonte**, o **executável** e o **arquivo com as especificações** léxica e sintática e a numeração das ações semânticas (no GALS, arquivo com extensão `.gals`) e demais recursos.

- 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; a entrega de todos os artefatos relacionados anteriormente (a não entrega de algum dos artefatos implicará na não correção do trabalho).

**DATA LIMITE PARA ENTREGA:** até às 23h do dia 09/12/2022 (sexta-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	<pre> 1 fun main { 2   var lado, area: int; 3   readln ("digite o lado do quadrado: , lado); 4   area = lado * "lado"; 5   print area); 6 }</pre>	Erro na linha 3 - constante string inválida ou não finalizada

### EXEMPLO 2: com erro semântico

ENTRADA		SAÍDA (na área de mensagens)
linha	<pre> 1 fun main { 2   var lado, area: int; 3   readln ("digite o lado do quadrado: ", lado); 4   area = lado * "lado"; 5   print area); 6 }</pre>	Erro na linha 4 - tipos incompatíveis em expressão aritmética

### EXEMPLO 3: com erro sintático

ENTRADA		SAÍDA (na área de mensagens)
linha	<pre> 1 fun main { 2   var lado, area: int; 3   readln ("digite o lado do quadrado: ", lado); 4   area = lado * lado; 5   print area); 6 }</pre>	Erro na linha 5 - encontrado area esperado (

### 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	<pre> 1 fun main { 2   var lado, area: int; 3   readln ("digite o lado do quadrado: ", lado); 4   area = lado * lado; 5   print (area); 6 }</pre>	programa compilado com sucesso

programa objeto (teste_01.il)	
<pre> .assembly extern mscorlib {} .assembly _codigo_objeto{} .module _codigo_objeto.exe  .class public _UNICA{ .method static public void _principal() { .entrypoint .locals (int64 lado) .locals (int64 area) ldstr "digite o lado do quadrado: " call void [mscorlib]System.Console::Write(string) call string [mscorlib]System.Console::ReadLine() call int64 [mscorlib]System.Int64::Parse(string) stloc lado</pre>	<pre> ldloc lado conv.r8 ldloc lado conv.r8 mul conv.i8 stloc area ldloc area conv.r8 conv.i8 call void [mscorlib]System.Console::Write(int64) ret }</pre>