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 nº2.pdf`) disponibilizado no AVA (pasta avaliações). 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	– Caso o botão **compile** (ou a tecla de atalho associada) 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: 1ª situação: programa compilado com sucesso ✓ **mensagem** (*programa compilado com sucesso*), na área reservada para mensagens, indicando que o programa não apresenta erros. ✓ **código objeto** em *MicroSoft Intermediate Language*, corresponde ao programa fonte compilado. O código objeto deve ser gerado na MESMA PASTA do programa fonte que está sendo compilado e deve estar em um arquivo texto com extensão `.il` e nome igual ao nome do arquivo que contém o programa compilado. 2ª situação: programa apresenta erros ✓ **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. Observa-se que, se o programa não tiver sido salvo (for um programa novo), antes de efetuar a ação correspondente ao botão pressionado, deve ser solicitado que o programa seja salvo.

OBSERVAÇÕES:

- O tipo do analisador sintático a ser gerado é **LL (1)**.
- As mensagens para os **erros léxicos** devem ser conforme especificado na parte 2 do trabalho final, com as devidas correções, conforme indicado na avaliação do analisador léxico. Caso contrário, a nota do analisador léxico poderá ser alterada. Observa-se que no tratamento de erros léxicos, a mensagem para constante caractere incorreta deve ser alterada para: `constante str inválida ou não finalizada`.
- As mensagens para os **erros sintáticos** devem ser conforme especificado na parte 3 do trabalho final, com as devidas correções, conforme indicado na avaliação do analisador sintático. Caso contrário, a nota do analisador sintático poderá ser alterada.
- 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. Caso o erro envolva o uso de identificadores, deve também ser apresentado o identificador que causou o erro. As mensagens de erro devem ser geradas durante a execução das ações semânticas. Assim, tem-se alguns exemplos:

  Erro na linha 1 – **`nota`** já declarado

  Erro na linha 10 – **`idade`** não declarado

  Erro na linha 15 – tipos incompatíveis em comando de atribuição

  Erro na linha 20 – tipo incompatível em operação unária

  Ao ser emitida uma mensagem de erro semântico, o processo de análise deve ser encerrado.
- No **esquema de tradução** disponibilizado, a gramática, com não determinismo e recursão à esquerda, 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 devem ser 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** gerado deve estar 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 **pasta da sua equipe**. Deve ser disponibilizado um

arquivo compactado (com o nome: `compilador`), contendo: 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`).

- 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 e no esquema de tradução nº2; o uso apropriado de ferramentas para construção de compiladores.

DATA LIMITE PARA ENTREGA: até às 23h do dia 11/07 (quarta-feira). Não serão aceitos trabalhos após data e hora determinadas.

EXEMPLOS DE ENTRADA / SAÍDA

EXEMPLO 1: com erro léxico

ENTRADA: programa fonte `teste_02.txt`		**SAÍDA (na área de mensagens)**
linha		`Erro na linha 3 - constante str inválida ou não finalizada`
1	def	
2	execute	
3	print("digite um valor para lado:)	
4	input lado)	
5	area:= lado * lado	
6	print(area)	

EXEMPLO 2: com erro sintático

ENTRADA: programa fonte `teste_02.txt`		**SAÍDA (na área de mensagens)**
linha		`Erro na linha 4 - encontrado lado esperado (`
1	def	
2	execute	
3	print("digite um valor para lado:")	
4	input lado)	
5	area:= lado * lado	
6	print(area)	

EXEMPLO 3: com erro semântico

ENTRADA: programa fonte `teste_02.txt`		**SAÍDA (na área de mensagens)**
linha		`Erro na linha 4 - lado não declarado`
1	def	
2	execute	
3	print("digite um valor para lado:")	
4	input(lado)	
5	area:= lado * lado	
6	print(area)	

EXEMPLO 4: sem erro

ENTRADA: programa fonte `teste_02.txt`		**SAÍDA (na área de mensagens)**
linha		`programa compilado com sucesso`
1	def	
2	var lado, area: float	**SAÍDA (na pasta do arquivo contendo o programa fonte `teste_02.txt`):**
3	execute	
4	print("digite um valor para lado:")	arquivo **`teste_02.il`**, contendo o código objeto correspondente ao código fonte, VER NOS EXEMPLOS.
5	input(lado)	
6	area:= lado * lado	
7	print(area)	