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 disponibilizado no AVA (repositório avaliações, arquivo **esquema de tradução**). Deve-se também implementar **tratamento de erros** semânticos bem como as 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 **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: 1ª situação: programa compilado com sucesso ✓ **mensagem** (*programa compilado com sucesso*), na área reservada para mensagens, indicando que o programa não apresenta erros. 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. • Caso o botão **gerar código** 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** (*código objeto gerado 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 no mesmo diretório 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, em todas as situações, 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, se necessário.
- As mensagens para os **erros sintáticos** devem ser conforme especificado na parte 3 do trabalho final, com as devidas correções, se necessário.
- 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 2 – identificador (**`f_nota`**) já declarado

  Erro na linha 10 – identificador (**`i_idade`**) não declarado

  Erro na linha 15 – tipos incompatíveis em comando de atribuição (`int`, `string`)

  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.exe 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, no **repositório 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 o uso apropriado de ferramentas para construção de compiladores.

DATA LIMITE PARA ENTREGA: até às 23h do dia 28/06/2013 (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		Erro na linha 1 - $ símbolo inválido
1	main module [$	
2	:	
3		
4	in (i_lado);	
5	i_area <- i_lado * i_lado;	
6	out (i_area);	
7	]	

EXEMPLO 2: com erro sintático

ENTRADA		SAÍDA (na área de mensagens)
linha		Erro na linha 4 - encontrado in esperado identificador
1	main module [	
2	:	
3		
4	in (i_lado);	
5	i_area <- i_lado * i_lado;	
6	out (i_area);	
7	]	

EXEMPLO 3: com erro semântico

ENTRADA		SAÍDA (na área de mensagens)
linha		Erro na linha 5 - identificador (i_area) não declarado
1	main module [	
2	: i_lado;	
3		
4	in (i_lado);	
5	i_area <- i_lado * i_lado;	
6	out (i_area);	
7	]	

EXEMPLO 4: sem erro – botão **compilar**

ENTRADA		**SAÍDA (na área de mensagens)**
linha		programa compilado com sucesso
1	main module [	
2	: i_lado, i_area;	
3		
4	in (i_lado);	
5	i_area <- i_lado * i_lado;	
6	out (i_area);	
7	]	

EXEMPLO 5: sem erro – botão **gerar código**

ENTRADA: arquivo **teste_01.txt**		**SAÍDA (na área de mensagens)**
linha		código objeto gerado com sucesso
1	main module [	
2	: i_lado, i_area;	
3		
4	in (i_lado);	
5	i_area <- i_lado * i_lado;	
6	out (i_area);	
7	]	

SAÍDA (no diretório do arquivo teste_01.txt): teste_01.il

```
.assembly extern mscorlib {}
.assembly teste_01{}
.module teste_01.exe

.class public teste_01{
  .method static public void principal()
  { .entrypoint
    .locals (int64 i_lado, int64 i_area)
    call string [mscorlib]System.Console::ReadLine()
    call int64 [mscorlib]System.Int64::Parse(string)
    stloc i_lado
    ldloc i_lado
    ldloc i_lado
    mul
    stloc i_area
    ldloc i_area
    call void [mscorlib]System.Console::Write(int64)
    ret
  }
}
```