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	– 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. ✓ **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. Assim, por exemplo, se o programa compilado for `teste01.txt` da pasta `c:\temp\`, deve ser gerado na pasta `c:\temp\` o arquivo `teste01.il`. 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.

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. Assim, tem-se alguns exemplos:

  Erro na linha 10 - tipos incompatíveis em comando de atribuição

  Erro na linha 15 - _iarea não declarado

  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 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 tarefa **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 o **código fonte**, o **executável** e o **arquivo com as especificações léxica e sintática** (no GALS, arquivo com extensão `.gals`) e demais recursos. Basta um integrante da equipe postar o trabalho. Caso não sejam postados todos os arquivos solicitados, será atribuída nota 0.0 (zero).

- 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. Além disso, trabalhos cujo código objeto não seja gerado em um arquivo texto com extensão `.il` na MESMA PASTA do programa fonte que está sendo compilado, receberão nota 0.0 (zero).

DATA LIMITE PARA ENTREGA: até às 23h do dia 11/12/2023 (segunda-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 fun main { 2 _ilado; 3 in ("digite um valor para lado: , _ilado} 4 _iarea = _ilado * _ilado; 5 out (_iarea); }	Erro na linha 3 - constante_string inválida

EXEMPLO 2: com erro sintático

ENTRADA	SAÍDA (na área de mensagens)
linha 1 fun main { 2 _ilado; 3 in ("digite um valor para lado: ", _ilado} 4 _iarea = _ilado * _ilado; 5 out (_iarea); }	Erro na linha 3 - encontrado } esperado , ; = :)

EXEMPLO 3: com erro semântico

ENTRADA	SAÍDA (na área de mensagens)
linha 1 fun main { 2 _ilado; 3 in ("digite um valor para lado: ", _ilado); 4 _iarea = _ilado * _ilado; 5 out (_iarea); }	Erro na linha 4 - _iarea 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 fun main { 2 _ilado, _iarea; 3 in ("digite um valor para lado: ", _ilado); 4 _iarea = _ilado * _ilado; 5 out (_iarea); }	programa compilado com sucesso

programa objeto (**`teste_01.il`**)

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

.class public _exemplo{

.method static public void _principal() {
.entrypoint
.locals(int64 _ilado)
.locals(int64 _iarea)
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 _ilado
ldloc _ilado
conv.r8
ldloc _ilado
conv.r8
mul
conv.i8
stloc _iarea
ldloc _iarea
conv.r8
conv.i8
call void [mscorlib]System.Console::WriteLine(int64)
ret
}
}
```