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

Implementar as ações semânticas que constituem o **analisador semântico** e o **gerador de código intermediário**, gerando o código objeto IL, de acordo com o esquema de tradução (`esquema.pdf`) disponibilizado no AVA (aba COMPILADOR). Deve-se também implementar **tratamento de erros** semânticos, a partir da descrição da semântica especificada 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`. Considere que só serão compilados arquivos já salvos. 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 descrição da semântica especificada no esquema de tradução. 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.
- A implementação do analisador semântico e do gerador de código intermediário, juntamente com os analisadores léxico e sintático e a interface do compilador, deve ser disponibilizada no **AVA**, na aba **COMPILADOR**, na tarefa "**parte 4 - analisador semântico e gerador de código**". Deve ser disponibilizado um **arquivo compactado** (com o nome: `compilador`, seguido do número da equipe), contendo o projeto completo, incluindo:
  (1) **código fonte**;
  (2) **executável** (ou **.jar**);
  (3) **arquivo com as especificações léxica e sintática** e a numeração das ações semânticas (no GALS, arquivo com extensão `.gals`);
  (4) demais recursos.

  Basta um integrante da equipe postar o trabalho.
- Para que o analisador semântico e o gerador de código intermediário 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 no formato especificado, em um arquivo texto com extensão `.il` com nome igual ao nome do arquivo que contém o programa fonte compilado e na MESMA PASTA do programa fonte compilado, caso contrário, será atribuída nota 0.0 (zero) à parte 4.
- Na avaliação do analisador semântico e do gerador de código intermediário 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); a qualidade das mensagens de erro, conforme descrito acima; o uso apropriado de ferramentas para construção de compiladores.

DATA LIMITE PARA ENTREGA: até às 23h do dia 05/12/2025 (sexta-feira). Não serão aceitos trabalhos após data e hora determinadas.

EXEMPLOS DE ENTRADA / SAÍDA

EXEMPLO 1: com erro léxico

ENTRADA: programa fonte - arquivo **teste_01.txt**

```
linha
   1:
   2: begin
   3:   int lado;
   4:   read  ("digite um valor para lado: , lado
   5:   if lado  print ("invalido");
   6:   else     print ("valor digitado: ", lado);
   7:   end;
   8: end
```

SAÍDA (na área de mensagens)

```
linha 3: constante_string inválida
```

EXEMPLO 2: com erro sintático

ENTRADA: programa fonte - arquivo **teste_01.txt**

```
linha
   1:
   2: begin
   3:   int lado;
   4:   read  ("digite um valor para lado: ", lado
   5:   if lado print ("invalido");
   6:   else    print ("valor digitado: ", lado);
   7:   end;
   8: end
```

SAÍDA (na área de mensagens)

```
linha 4: encontrado if esperado ,  )
```

EXEMPLO 3: com erro semântico

ENTRADA: programa fonte - arquivo **teste_01.txt**

```
linha
   1:
   2: begin
   3:   int lado;
   4:   read  ("digite um valor para lado: ", lado);
   5:   if lado print ("invalido");
   6:   else    print ("valor digitado: ", lado);
   7:   end;
   8: end
```

SAÍDA (na área de mensagens)

```
linha 4: expressão incompatível em comando de seleção
```

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

ENTRADA: programa fonte - arquivo **teste_01.txt**

```
linha
   1:
   2: begin
   3:   int lado;
   4:   read  ("digite um valor para lado: ", lado);
   5:   if lado < 1 print ("invalido");
   6:   else        print ("valor digitado: ", lado);
   7:   end;
   8: end
```

SAÍDA (na área de mensagens)

```
programa compilado com sucesso
```

SAÍDA (na mesma parta do programa fonte): programa objeto - arquivo **teste_01.il**

```
// cabeçalho
.assembly extern mscorlib {}
.assembly _programa{}
.module _programa.exe

.class public _unica{
  .method static public void _principal(){
   .entrypoint
// cabeçalho

.locals(int64 lado)
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 lado
ldloc lado
conv.r8
ldc.i8 1
conv.r8
clt
brfalse L1
ldstr "invalido"
call void [mscorlib]System.Console::Write(string)
ldstr "\n"
call void [mscorlib]System.Console::Write(string)
br L2
L1:
ldstr "valor digitado: "
call void [mscorlib]System.Console::Write(string)
ldloc lado
conv.r8
conv.i8
call void [mscorlib]System.Console::Write(int64)
ldstr "\n"
call void [mscorlib]System.Console::Write(string)
L2:
// fim de programa
ret
}
}
```