**ESQUEMA DE TRADUÇÃO nº1:** determinar o tipo de uma <expressão>

|  |
| --- |
| <expressão> ::= <termo> <expressão\_>  <expressão\_> ::= **+** <termo> **#1** <expressão\_>  | **-** <termo> **#2** <expressão\_>  | ε  <termo> ::= <elemento> <termo\_>  <termo\_> ::= **\*** <elemento> **#3** <termo\_>  | **/** <elemento> **#4** <termo\_>  | ε  <elemento> ::= constante\_int **#5**  | constante\_ float **#6**  | **(** <expressão> **)** |

**DESCRIÇÃO DOS REGISTROS SEMÂNTICOS:** para executar a análise semântica e a geração de código é necessário fazer uso de registros semânticos:

* **pilha\_tipos** (inicialmente vazia): usada para determinar o tipo de uma expressão durante a compilação do programa.

**TABELA DE TIPOS**: o tipo de uma <expressão> deve ser determinado da seguinte forma:

| operando1 | operando2 | operador | tipo resultante |
| --- | --- | --- | --- |
| constante\_int |  |  | **int64** |
| constante\_float |  |  | **float64** |
| **int64** | **int64** | operadores binários**: + - \*** | **int64** |
| **int64**  **float64**  **float64** | **float64**  **int64**  **float64** | operadores binários**: + - \*** | **float64** |
| **int64** ou **float64** | **int64** ou **float64** | **/** | **float64** |

**DESCRIÇÃO DA SEMÂNTICA**:

A semântica para determinar o tipo de uma <expressão> é a seguinte:

Para constantes (constante\_int: ação **#5**, constante\_float: ação **#6**): empilhar na **pilha\_tipos** o tipo correspondente, conforme indicado na TABELA DE TIPOS.

Para os operadores (aritméticos binários: ações **#1**, **#2**, **#3**, **#4**): desempilhar dois tipos da **pilha\_tipos**; implementar a verificação de tipos conforme indicado na TABELA DE TIPOS; empilhar o tipo correspondente.

**EXEMPLO - compilando a expressão: 3 + 4**

|  |  |
| --- | --- |
| **ação** | **pilha\_tipos** (atributo do analisador semântico, durante a compilação do programa) |
|  |  |
|  |  |
|  |  |

**compilando a expressão: 3 - 4 / 2**

|  |  |
| --- | --- |
| **ação** | **pilha\_tipos** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**compilando a expressão: 3.5 \* (4 + 4)**

|  |  |
| --- | --- |
| **ação** | **pilha\_tipos** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**ESQUEMA DE TRADUÇÃO nº2:** gerar código objeto para uma <expressão> em instruções em MSIL (MicroSoft Intermediate Language)

|  |
| --- |
| <expressão> ::= <termo> <expressão\_>  <expressão\_> ::= **+** <termo> **#1** <expressão\_>  | **-** <termo> **#2** <expressão\_>  | ε  <termo> ::= <elemento> <termo\_>  <termo\_> ::= **\*** <elemento> **#3** <termo\_>  | **/** <elemento> **#4** <termo\_>  | ε  <elemento> ::= constante\_int **#5**  | constante\_ float **#6**  | **(** <expressão> **)** |

**DESCRIÇÃO DOS REGISTROS SEMÂNTICOS:**

* **código**: usado para armazenar o código objeto gerado.

**DESCRIÇÃO DA SEMÂNTICA**:

A semântica para gerar código para uma <expressão> é a seguinte:

Para constante\_int (ação **#5**): gerar código objeto para carregar o valor da constante (código: ldc.i8 token.getLexeme), observando que a constante\_int da linguagem fonte deve ser tratada como float64 em MSIL, portanto deve ser convertida para float64 (código: conv.r8).

Para constante\_float (ação **#6**): gerar código objeto para carregar o valor da constante (código: ldc.r8 token.getLexeme).

Para os operadores (aritméticos binários: ações **#1**, **#2**, **#3**, **#4**): gerar código objeto para efetuar a operação correspondente (código: add, sub, mul ou div, respectivamente).

**EXEMPLO - compilando a expressão: 3 + 4**

|  |  |
| --- | --- |
| **ação** | **código** (atributo do analisador semântico, durante a compilação do programa) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**compilando a expressão: 3 - 4 / 2**

|  |  |
| --- | --- |
| **ação** | **código** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**compilando a expressão: 3.5 \* (4 + 4)**

|  |  |
| --- | --- |
| **ação** | **código** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**ESQUEMA DE TRADUÇÃO nº3:** determinar o tipo e gerar código objeto para uma <expressão>

|  |
| --- |
| <expressão> ::= <termo> <expressão\_>  <expressão\_> ::= **+** <termo> **#1** <expressão\_>  | **-** <termo> **#2** <expressão\_>  | ε  <termo> ::= <elemento> <termo\_>  <termo\_> ::= **\*** <elemento> **#3** <termo\_>  | **/** <elemento> **#4** <termo\_>  | ε  <elemento> ::= constante\_int **#5**  | constante\_ float **#6**  | **(** <expressão> **)** |

**DESCRIÇÃO DOS REGISTROS SEMÂNTICOS:**

* **código**: usado para armazenar o código objeto gerado.
* **pilha\_tipos** (inicialmente vazia): usada para determinar o tipo de uma expressão durante a compilação do programa.

**TABELA DE TIPOS**: o tipo de uma <expressão> deve ser determinado da seguinte forma:

| operando1 | operando2 | operador | tipo resultante |
| --- | --- | --- | --- |
| constante\_int |  |  | **int64** |
| constante\_float |  |  | **float64** |
| **int64** | **int64** | operadores binários**: + - \* /** | **int64** |
| **int64**  **float64**  **float64** | **float64**  **int64**  **float64** | operadores binários**: + - \*** | **float64** |
| **float64** | **float64** | **/** | **float64** |

Operadores e tipos não previstos na tabela acima indicam situação de erro.

**DESCRIÇÃO DA SEMÂNTICA**:

A semântica para gerar código para uma <expressão> é a seguinte:

Para constante\_int (ação **#5**):

empilhar na **pilha\_tipos** o tipo correspondente, conforme indicado na TABELA DE TIPOS;

gerar código objeto para carregar o valor da constante (código: ldc.i8 token.getLexeme), observando que a constante\_int da linguagem fonte deve ser tratada como float64 em MSIL, portanto deve ser convertida para float64 (código: conv.r8).

Para constante\_float (ação **#6**):

empilhar na **pilha\_tipos** o tipo correspondente, conforme indicado na TABELA DE TIPOS;

gerar código objeto para carregar o valor da constante (código: ldc.r8 token.getLexeme).

Para os operadores (aritméticos binários: ações **#1**, **#2**, **#3**, **#4**):

desempilhar dois tipos da **pilha\_tipos**; implementar a verificação de tipos conforme indicado na TABELA DE TIPOS; empilhar o tipo correspondente;

gerar código objeto para efetuar a operação correspondente (código: add, sub, mul ou div, respectivamente).

**EXEMPLO - compilando a expressão: 3.5 \* (4 + 4)**

|  |  |  |
| --- | --- | --- |
| **ação** | **pilha\_tipos** | **código** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**compilando a expressão: 3 - 4 / 2**

|  |  |  |
| --- | --- | --- |
| **ação** | **pilha\_tipos** | **código** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**ESQUEMA DE TRADUÇÃO nº4:** determinar o tipo e gerar código objeto para operadores unários

|  |
| --- |
| <expressão> ::= <termo> <expressão\_>  <expressão\_> ::= **+** <termo> **#1** <expressão\_>  | **-** <termo> **#2** <expressão\_>  | ε  <termo> ::= <elemento> <termo\_>  <termo\_> ::= **\*** <elemento> **#3** <termo\_>  | **/** <elemento> **#4** <termo\_>  | ε  <elemento> ::= constante\_int **#5**  | constante\_ float **#6**  | **(** <expressão> **)**  | **+** <elemento> **#7**  | **-** <elemento> **#8** |

**(complementando esquema de tradução nº3)**

**TABELA DE TIPOS**:

| operando1 | operando2 | operador | tipo resultante |
| --- | --- | --- | --- |
| **int64** |  | operadores unários: **+ -** | **int64** |
| **float64** |  | operadores unários: **+ -** | **float64** |

Operadores e tipos não previstos na tabela acima indicam situação de erro.

**DESCRIÇÃO DA SEMÂNTICA**:

Para o operador (aritmético unário **+**: ação **#7**):

desempilhar um tipo da **pilha\_tipos**; implementar a verificação de tipos conforme indicado na TABELA DE TIPOS; empilhar o tipo correspondente.

Para o operador (aritmético unário **-**: ação **#8**):

desempilhar um tipo da **pilha\_tipos**; implementar a verificação de tipos conforme indicado na TABELA DE TIPOS; empilhar o tipo correspondente.

gerar código objeto para efetuar a operação correspondente.

**EXEMPLO - compilando a expressão: -3 + 4**

|  |  |  |
| --- | --- | --- |
| **ação** | **pilha\_tipos** | **código** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**compilando a expressão: -(3 + 4)**

|  |  |  |
| --- | --- | --- |
| **ação** | **pilha\_tipos** | **código** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**ESQUEMA DE TRADUÇÃO nº5:** determinar o tipo e gerar código objeto para operadores relacionais

|  |
| --- |
| <relacional> ::= <expressão> <operador> **#9**  <expressão> **#10**  | <expressão>  <operador> ::= **<** | **>** | **==**  <expressão> ::= <termo> <expressão\_>  <expressão\_> ::= **+** <termo> **#1** <expressão\_>  | **-** <termo> **#2** <expressão\_>  | ε  <termo> ::= <elemento> <termo\_>  <termo\_> ::= **\*** <elemento> **#3** <termo\_>  | **/** <elemento> **#4** <termo\_>  | ε  <elemento> ::= constante\_int **#5**  | constante\_ float **#6**  | **(** <expressão> **)**  | **+** <elemento> **#7**  | **-** <elemento> **#8** |

**(complementando esquema de tradução nº3)**

**DESCRIÇÃO DOS REGISTROS SEMÂNTICOS:**

* **operador** (inicialmente igual a ""): usado para armazenar o operador relacional reconhecido pela ação **#9**, para uso posterior na ação **#10**.

**TABELA DE TIPOS**:

| operando1 | operando2 | operador | tipo resultante |
| --- | --- | --- | --- |
| **int64** | **int64** | operadores binários: **< > ==** | **bool** |
| **float64** | **float64** | operadores binários: **< > ==** | **bool** |

Operadores e tipos não previstos na tabela acima indicam situação de erro.

**DESCRIÇÃO DA SEMÂNTICA**:

Para os operadores (relacionais binários: ação **#9**): guardar o operador relacional reconhecido.

Para os operadores (relacionais binários: ação **#10**):

desempilhar dois tipos da **pilha\_tipos**; implementar a verificação de tipos conforme indicado na TABELA DE TIPOS; empilhar o tipo correspondente (**bool**);

gerar código objeto para efetuar a operação correspondente ao operador relacional armazenado em **operador**.

**compilando a expressão: 3 == -4**

|  |  |  |  |
| --- | --- | --- | --- |
| **ação** | **pilha\_tipos** | **código** | **operador** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**ESQUEMA DE TRADUÇÃO nº6:** determinar o tipo e gerar código objeto para constantes lógicas e operador **not**

|  |
| --- |
| <lógica> ::= **true** **#11**  | **false** **#12**  | **not (** <lógica> **) #13**  | <relacional>  <relacional> ::= <expressão> <operador> **#9** <expressão> **#10**  | <expressão>  <operador> ::= **<** | **>** | **==**  <expressão> ::= <termo> <expressão\_>  <expressão\_> ::= **+** <termo> **#1** <expressão\_>  | **-** <termo> **#2** <expressão\_>  | ε  <termo> ::= <elemento> <termo\_>  <termo\_> ::= **\*** <elemento> **#3** <termo\_>  | **/** <elemento> **#4** <termo\_>  | ε  <elemento> ::= constante\_int **#5**  | constante\_ float **#6**  | **(** <expressão> **)**  | **+** <elemento> **#7**  | **-** <elemento> **#8** |

**(complementando esquema de tradução nº3)**

**TABELA DE TIPOS**:

| operando1 | operando2 | operador | tipo resultante |
| --- | --- | --- | --- |
| **true** |  |  | **bool** |
| **false** |  |  | **bool** |
| **bool** | **bool** | operador unário**: not** | **bool** |

Operadores e tipos não previstos na tabela acima indicam situação de erro.

**DESCRIÇÃO DA SEMÂNTICA**:

Para constante **true** (ação **#11**) e constante **false** (ação **#12**):

empilhar na **pilha\_tipos** o tipo correspondente, conforme indicado na TABELA DE TIPOS;

gerar código objeto para carregar o valor da constante.

Para o operador (lógico unário **not**: ação **#13**):

desempilhar um tipo da **pilha\_tipos**; implementar a verificação de tipos conforme indicado na TABELA DE TIPOS; empilhar o tipo correspondente.

gerar código objeto para efetuar a operação correspondente.

**ESQUEMA DE TRADUÇÃO nº7:** gerar código objeto para comando de saída

|  |
| --- |
| <saída> ::= **write**  **(** <lista expressão> **)**  <lista expressão>::= <lógica>  | <lógica>  **,** <lista expressão>  <lógica> ::= **true** **#11**  | **false** **#12**  | **not (** <lógica> **) #13**  | <relacional>  <relacional> ::= <expressão> <operador> **#9** <expressão> **#10**  | <expressão>  <operador> ::= **<** | **>** | **==**  <expressão> ::= <termo> <expressão\_>  <expressão\_> ::= **+** <termo> **#1** <expressão\_>  | **-** <termo> **#2** <expressão\_>  | ε  <termo> ::= <elemento> <termo\_>  <termo\_> ::= **\*** <elemento> **#3** <termo\_>  | **/** <elemento> **#4** <termo\_>  | ε  <elemento> ::= constante\_int **#5**  | constante\_ float **#6**  | **(** <expressão> **)**  | **+** <elemento> **#7**  | **-** <elemento> **#8** |

**(complementando esquema de tradução nº3)**

**DESCRIÇÃO DA SEMÂNTICA**:

A semântica do comando <saída> é a seguinte (ação **#14**):

desempilhar um tipo da **pilha\_tipos**;

valores do tipo int da linguagem fonte são tratados como float64 em MSIL, portanto devem ser primeiramente convertidos para int64 (código: conv.i8).

gerar código objeto para escrever um valor conforme o tipo desempilhado (código: call void [mscorlib]System.Console::Write(<tipo>), onde <tipo> pode se **int64**, **float64**, **string** ou **bool**).

**compilando a expressão: write (not (3 + 4 == 7), true, 3.5)**

|  |  |  |  |
| --- | --- | --- | --- |
| **ação** | **pilha\_tipos** | **código** | **operador** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |