Construção de Compiladores
Período Especial
Aula 9: Comando If

Bruno Müller Junior

Departamento de Informática
UFPR

2020

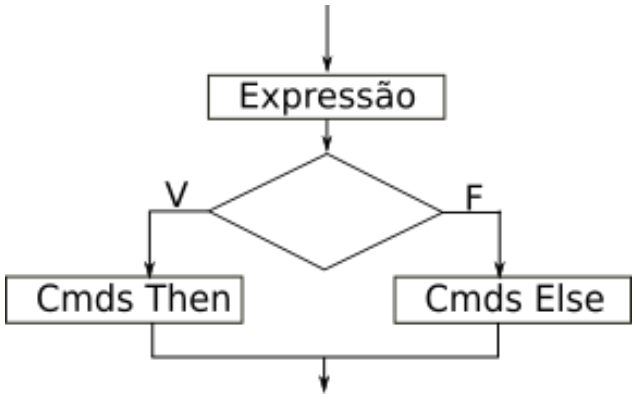
Sintaxe

- O comando `if` (regra 22) é uma das possibilidades de `<comando sem rótulo>` (regra 18).
- O analisador sintático irá escolher a regra 22 quando estiver na regra 18 e encontrar o token `IF`.
- É importante destacar que o comando `ELSE` é opcional, cuja gramática em formato ascendente (bison) é ambígua.

```
18.  <comando sem rótulo> ::= <atribuição>           |
                              ...                    |
                              <comando condicional>
22. <comando condicional>  ::= IF <expressão>
                                   THEN <comando sem rótulo>
                                   [ELSE <comando sem rótulo>]
```

Fluxo

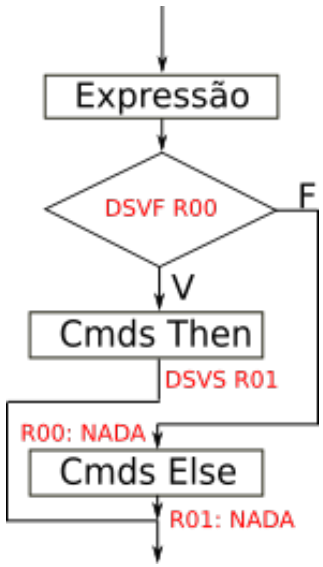
- Assim como no comando `WHILE`, o fluxo de execução do comando `IF` não é sequencial.
- Também existem desvios do fluxo para locais específicos.
- Porém, a figura ao lado não ajuda muito a ver onde colocar estes desvios de fluxo.



Esquema de Tradução

- A maneira mais simples de ver onde inserir os comandos é no esquema de tradução.

IF	
E	*Traduz Expressão* `DSVF R00`
THEN	
<comandos>	*Traduz Comandos* `DSVS R01`
ELSE	`R00: NADA`
<comandos>	*Traduz Comandos* `R01: NADA`

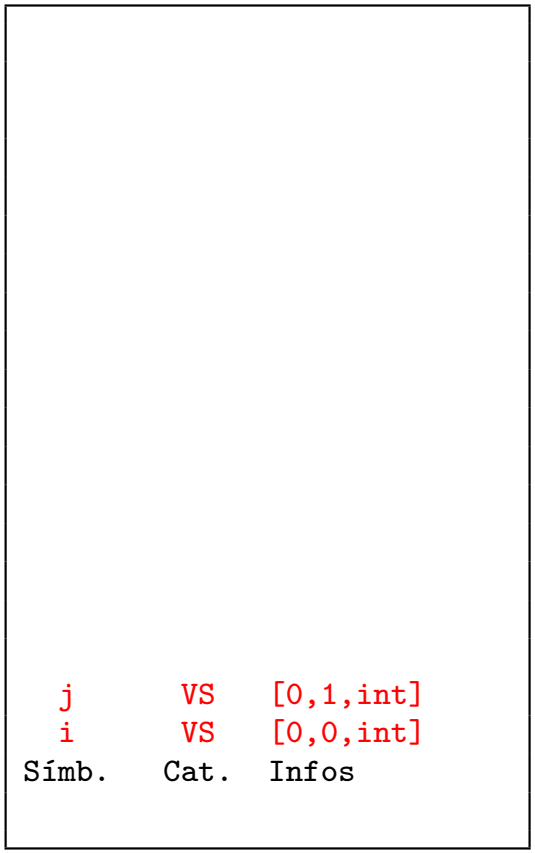


```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
program cmdIf (input, output);        INPP
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
INPP
AMEM 2
```



```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
INPP
AMEM 2
LEIT
ARMZ 0, 1
```

```
  j       VS    [0,1,int]
  i       VS    [0,0,int]
Símb.   Cat.   Infos
```

```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
INPP
AMEM 2
LEIT
ARMZ 0, 1
CRCT 0
ARMZ 0, 0
```

Símb.	Cat.	Infos
j	VS	[0,1,int]
i	VS	[0,0,int]

```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
      INPP
      AMEM 2
      LEIT
      ARMZ 0, 1
      CRCT 0
      ARMZ 0, 0
R00:NADA
      CRVL 0,0
      CRVL 0,1
      CMME
      DSVF R01
```

```
R00
R01

  j       VS    [0,1,int]
  i       VS    [0,0,int]
Símb.   Cat.   Infos
```

```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
      INPP
      AMEM 2
      LEIT
      ARMZ 0, 1
      CRCT 0
      ARMZ 0, 0
R00:  NADA
      CRVL 0,0
      CRVL 0,1
      CMME
      DSVF R01
      CRVL 0,0
      CRCT 2
      DIVI
      CRCT 2
      MULT
      CRVL 0,0
      CMIG
      DSVF R02
```

```
R00
R01
R02
R03

   j       VS    [0,1,int]
   i       VS    [0,0,int]
Símb.    Cat.   Infos
```

```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
      INPP            CRCT 0
      AMEM 2          IMPR
      LEIT
      ARMZ 0, 1
      CRCT 0
      ARMZ 0, 0
R00:  NADA
      CRVL 0,0
      CRVL 0,1
      CMME
      DSVF R01
      CRVL 0,0
      CRCT 2
      DIVI
      CRCT 2
      MULT
      CRVL 0,0
      CMIG
      DSVF R02
      CRVL 0,0
      IMPR
```

```
R00
R01
R02
R03

  j       VS    [0,1,int]
  i       VS    [0,0,int]
Símb.   Cat.   Infos
```

```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
      INPP            CRCT 0
      AMEM 2          IMPR
      LEIT            DSVS R03
      ARMZ 0, 1   R02:NADA
      CRCT 0          CRVL 0,0
      ARMZ 0, 0       IMPR
R00:  NADA            CRCT 1
      CRVL 0,0        IMPR
      CRVL 0,1    R03:NADA
      CMME
      DSVF R01
      CRVL 0,0
      CRCT 2
      DIVI
      CRCT 2
      MULT
      CRVL 0,0
      CMIG
      DSVF R02
      CRVL 0,0
      IMPR
```

```
R00
R01
R02
R03

  j      VS    [0,1,int]
  i      VS    [0,0,int]
Símb.   Cat.   Infos
```

```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
      INPP
      AMEM 2
      LEIT
      ARMZ 0, 1
      CRCT 0
      ARMZ 0, 0
R00:NADA
      CRVL 0,0
      CRVL 0,1
      CMME
      DSVF R01
      CRVL 0,0
      CRCT 2
      DIVI
      CRCT 2
      MULT
      CRVL 0,0
      CMIG
      DSVF R02
      CRVL 0,0
      IMPR
      CRCT 0
      IMPR
      DSVS R03
R02:NADA
      CRVL 0,0
      IMPR
      CRCT 1
      IMPR
R03:NADA
      CRVL 0,0
      CRCT 1
      SOMA
      ARMZ 0,0
```

```
R00
R01

  j      VS    [0,1,int]
  i      VS    [0,0,int]
Símb.   Cat.   Infos
```

```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
      INPP
      AMEM 2
      LEIT
      ARMZ 0, 1
      CRCT 0
      ARMZ 0, 0
R00:  NADA
      CRVL 0,0
      CRVL 0,1
      CMME
      DSVF R01
      CRVL 0,0
      CRCT 2
      DIVI
      CRCT 2
      MULT
      CRVL 0,0
      CMIG
      DSVF R02
      CRVL 0,0
      IMPR
      CRCT 0
      IMPR
      DSVS R03
R02:  NADA
      CRVL 0,0
      IMPR
      CRCT 1
      IMPR
R03:  NADA
      CRVL 0,0
      CRCT 1
      SOMA
      ARMZ 0,0
      DSVS R00
R01:  NADA
```

```
R00
R01
  j       VS    [0,1,int]
  i       VS    [0,0,int]
Símb.   Cat.   Infos
```

```
program cmdIf (input, output);
var i, j: integer;
begin
   read(j);
   i:=0;
   while (i < j) do
   begin
      if (i div 2 * 2 = i)
        then   write(i,0)
        else   write(i,1);
      i := i+1
   end;
end.
```

```
      INPP
      AMEM 2
      LEIT
      ARMZ 0, 1
      CRCT 0
      ARMZ 0, 0
R00:NADA
      CRVL 0,0
      CRVL 0,1
      CMME
      DSVF R01
      CRVL 0,0
      CRCT 2
      DIVI
      CRCT 2
      MULT
      CRVL 0,0
      CMIG
      DSVF R02
      CRVL 0,0
      IMPR
      CRCT 0
      IMPR
      DSVS R03
R02:NADA
      CRVL 0,0
      IMPR
      CRCT 1
      IMPR
R03:NADA
      CRVL 0,0
      CRCT 1
      SOMA
      ARMZ 0,0
      DSVS R00
R01:NADA
      DMEM 2
      PARA
```

```
Símb.   Cat.   Infos
```

Ambiguidade

- A gramática do comando `if-then-else` é ambígua.
- Considere gramática:

```
comando condicional  : IF expressão THEN comSemRot |
                       IF expressão THEN comSemRot ELSE comSemRot
```

- e o trecho de código:

```
if E1 then
   if E2 then C1
else C2 (* A quem pertence este else? *)
```

Convenção

- Os primeiros compiladores da linguagem `Algol 60` não observaram este fato.
- Alguns associaram o `ELSE` ao primeiro `IF` e outros ao segundo (um mesmo programa gerava dois códigos diferentes).
- Na bibliografia, este problema é conhecido como *dangling else*, e foi convencionado que o `ELSE` deve ser associado ao `IF` mais próximo.
- Como "amarrar" uma convenção no bison???

Bison

- Existem várias soluções para isto em bison. A mais simples de e explicar é associar precedências aos operadores.
- Assim, ao encontrar um ELSE, o bison deverá priorizar a conclusão da árvore sintática do ELSE.

```
// Precedências são crescentes, logo "lower_than_else" < "else".
%nonassoc "lower_then_else"
%nonassoc "else"
%%
stm: "if" "(" exp ")" stm  %prec "lower_then_else" |
     "if" "(" exp ")" stm "else" stm
```

- Uma alternativa detalhada pode ser encontrado em `http://www.inf.ufpr.br/bmuller/CI211/Recursos/IF.y`

- Página para anotações

Licença

- Slides desenvolvidos somente com software livre:
  - LaTeX usando beamer;
  - Inkscape.
- Licença:
  - Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License. `http://creativecommons.org/licenses/by-nc-nd/2.5/br/`