

Unidade III

3 TOMADA DE DECISÕES

Na vida, tudo depende da tomada de decisões. Para tomarmos uma decisão, uma série de condições deverá ser avaliada, e, então, decidiremos se uma ação deverá ou não ser realizada. A vida é repleta de tomadas de decisões: começar um namoro, estudar análise de sistemas, casar-se, pagar à vista, enfim, uma infinidade de situações necessita de uma análise anterior.

3.1 Teoria

Até agora, os programas foram executados em sequência, de cima para baixo, mas podem acontecer situações em que alguns comandos não sejam adequados e, por isso, não necessitem ser executados. Em outros casos, pode ser necessário optar por executar diferentes blocos de comandos, conforme a situação. Essas estruturas são chamadas de condicionais.

3.2 Condicionais

As estruturas condicionais são utilizadas quando há a necessidade de tomar decisões, ou quando é preciso efetuar comparações.

Nesse tipo de estrutura, uma operação lógica (<condição>) é avaliada. Se o resultado dessa avaliação for verdadeiro (V), então um determinado conjunto de instruções será executado. Se o resultado da avaliação for falso (F), um comando diferente será executado.

3.2.1 Se então

A primeira situação é quando uma condição é testada e, caso seja verdadeira, um bloco de comando é executado. Essa estrutura se chama **se então**.

Sintaxe:

```
se <condição> então  
    comando 1..... (neste caso a <condição> é verdadeira)  
fimse
```

Ao encontrar a condição **se**, o programa entende que virá uma expressão cujo resultado é lógico. Caso essa expressão resulte em Verdadeiro, é executado o bloco de comandos entre o **então** e o **fimse**. Caso a condição resulte em Falso, o bloco todo é ignorado, prosseguindo após o **fimse**.

A estrutura no fluxograma é representada da seguinte forma:

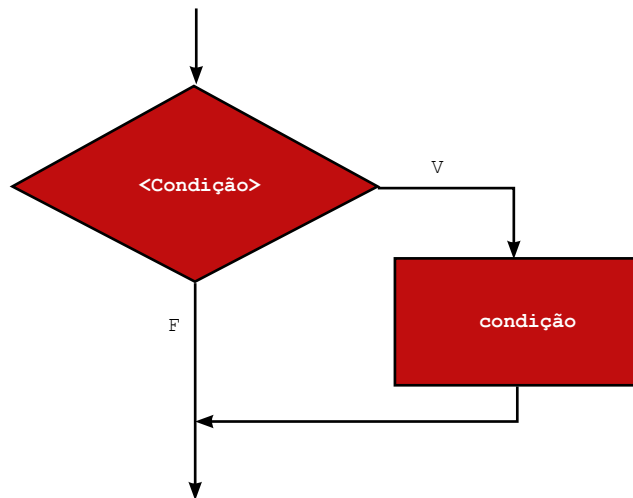


Figura 32 –Símbolo de condicional apenas com o **então**

Exemplo de aplicação

Faça um programa que leia um número e diga se é negativo.

```
algoritmo "negativo"
var
    numero:inteiro
inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    fimse
finalgoritmo
```

O fluxograma ficará:

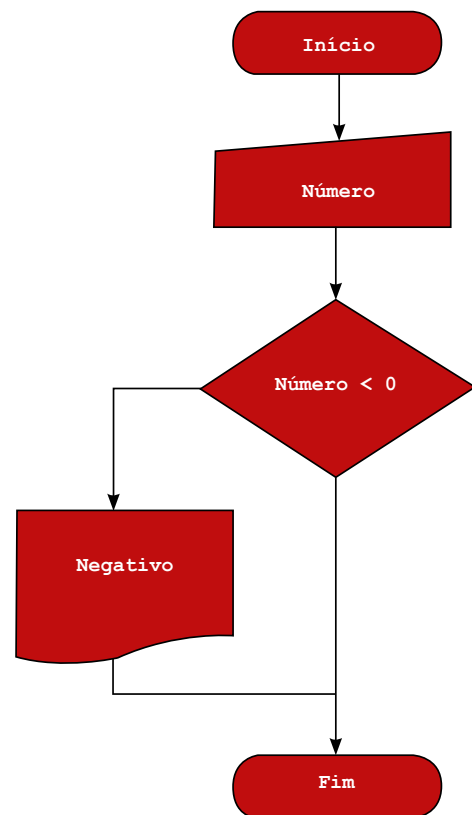


Figura 33 – Programa para avisar se um número é negativo

3.2.1.1 Fazendo o teste de mesa

Simulando o papel do computador para duas possibilidades: inicialmente com a entrada 5 e depois com a entrada -5.

Primeiro, um espaço de memória é inicializado para o programa "negativo".

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    Fimse
Fimalgoritmo
```

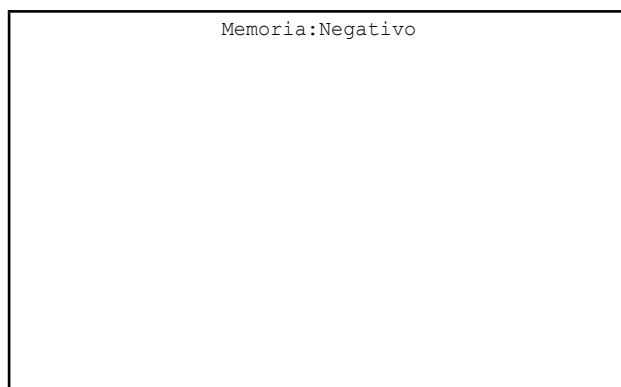


Figura 34 – Espaço de memória do programa "negativo"

As variáveis são separadas na memória. No caso, apenas a variável inteira *numero*.

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    Fimse
Fimalgoritmo
```

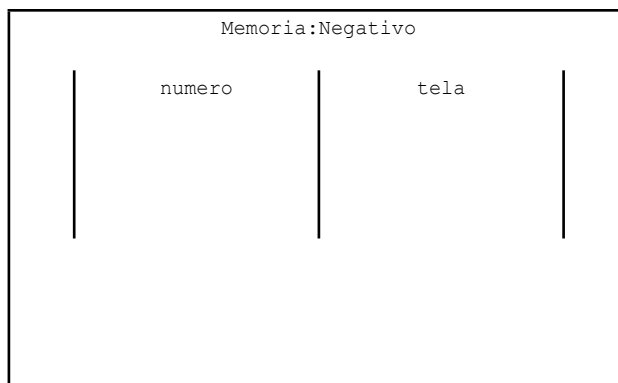


Figura 35 – Espaço de memória mostrando a variável *numero*

A primeira linha do programa mostra, na tela, o parâmetro do comando de saída **escreva**, no caso, **escreva um *numero***.

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    Fimse
Fimalgoritmo
```

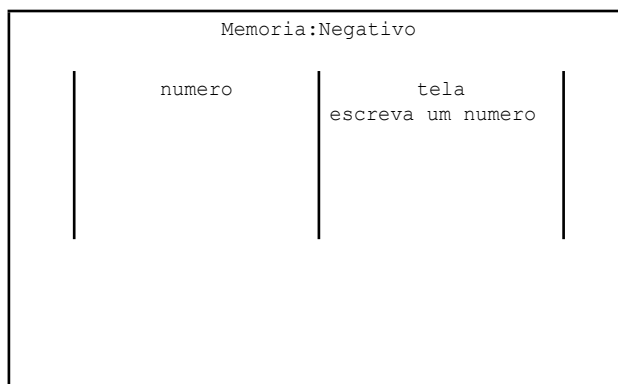


Figura 36 – Memória com tela mostrando o parâmetro do comando de saída

A seguir, é dada a entrada do valor 5, para testar a entrada de um número positivo.

Ao ser dada a entrada, o valor é atribuído na variável *numero*. Assim, o conteúdo de *numero* fica 5.

```
algoritmo "negativo"
Var
    numero:inteiro
```

```

Início
  escreva("Entre com um número")
  leia(numero)
  se numero<0 entao
    escreva(" negativo ")
  Fimse
Fimalgoritmo

```

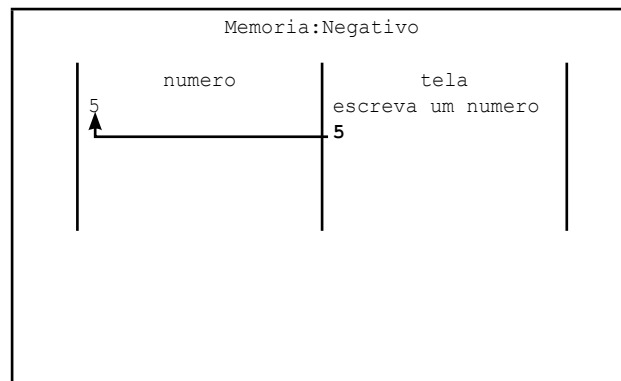


Figura 37 – Memória com tela mostrando a atribuição do valor 5 à variável *numero*

O comando seguinte é o que estamos estudando. Ao encontrar o **se**, o programa avaliará se a operação seguinte resultará em Verdadeiro. No caso, a operação *numero<0* resulta em Falso; assim, o fluxo não executará as instruções contidas no bloco do **então**.

```

algoritmo "negativo"
Var
  numero:inteiro
Início
  escreva("Entre com um número")
  leia(numero)
  se numero<0 entao
    escreva(" negativo ")
  Fimse
Fimalgoritmo

```

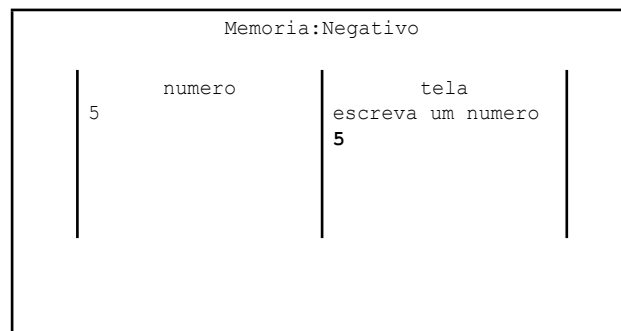


Figura 38 – Nesse caso, como a operação não resulta em Verdadeiro, as instruções no bloco do **então** não serão executadas

A seguir, o programa chega ao final, encerrando a simulação.

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    Fimse
Fimalgoritmo
```

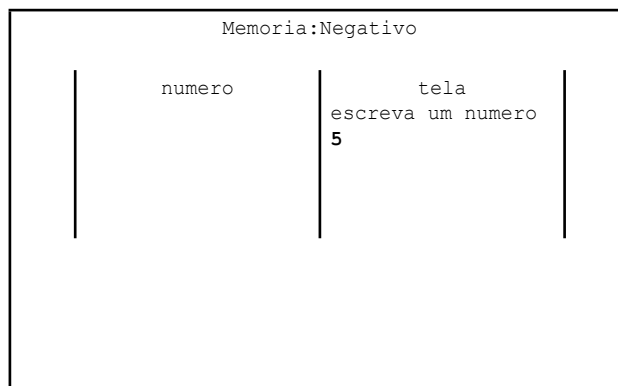


Figura 39 – Fim da simulação

Como esperado, ao entrarmos com o valor positivo, o programa não mostra nenhum resultado, pois somente o fará se o valor for negativo.

A seguir, faremos o teste de mesa para a entrada -5. Como o processamento é o mesmo até a entrada do valor, iniciaremos a partir da linha da entrada desse valor (leia).

O valor -5 é digitado no teclado, e, ao aplicarmos a tecla do *enter*, a entrada será transferida para a variável *numero*.

```
algoritmo "negativo"
var
    numero:inteiro
inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    Fimse
fimalgoritmo
```

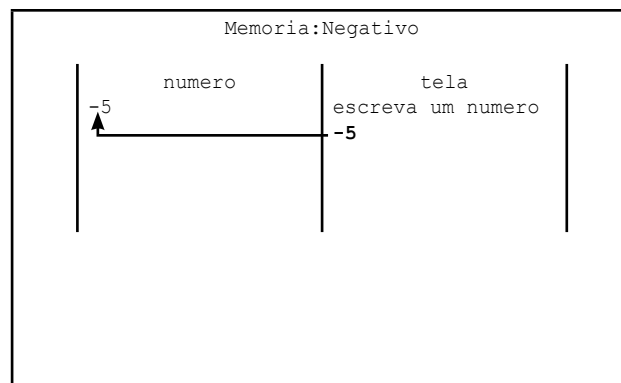


Figura 40 – Atribuição do valor -5 à variável *numero*

Na condicional, o resultado de '*numero*<0' é Verdadeiro, pois, como a variável contém -5, a comparação é $-5 < 0$, que é verdadeira. Como o resultado é Verdadeiro, o programa irá executar o bloco de comandos do **então**.



Lembrete

Na comparação $x < 0$, se o x valer 0, o resultado será falso; assim, o zero não será considerado negativo.

```

algoritmo "negativo"
var
    numero:inteiro
inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 então
        escreva(" negativo ")
    Fimse
finalgoritmo
    
```

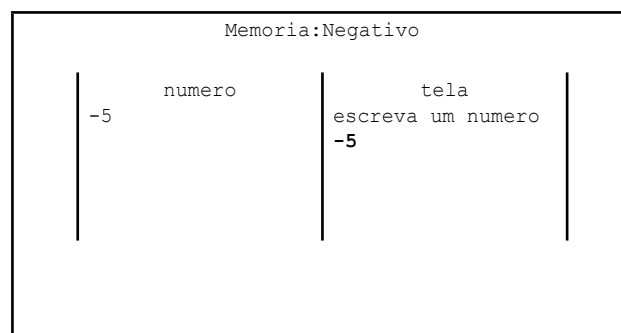


Figura 41 – Nesse caso, o bloco de comandos do **então** será executado, pois o resultado é Verdadeiro

No bloco do **então** há apenas um comando, no caso, a saída do texto "negativo".

```
algoritmo "negativo"
var
    numero:inteiro
inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    Fimse
finalgoritmo
```

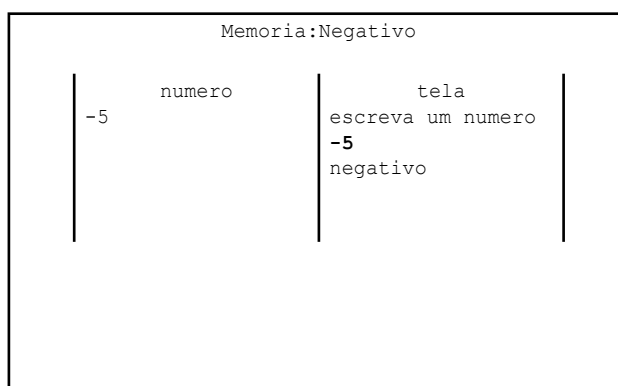


Figura 42 – Memória com tela mostrando o comando *negativo* no bloco do **então**

A seguir, o programa é encerrado.

```
algoritmo "negativo"
var
    numero:inteiro
inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    Fimse
finalgoritmo
```

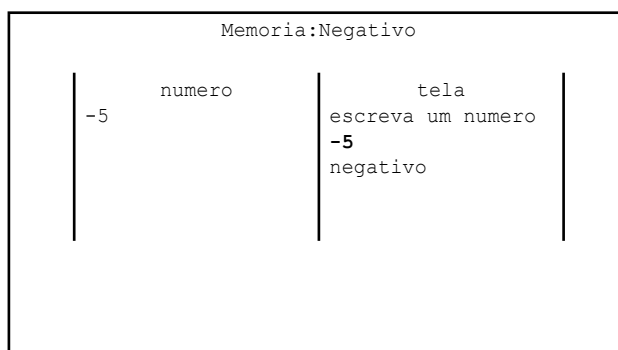


Figura 43 – Fim do programa

Assim, ao entrar com o valor -5, o programa mostra que o *numero* é negativo, indicando que a lógica do programa está de acordo com o proposto.

3.2.2 Se então senão

A segunda situação é quando uma condição é testada e, caso seja verdadeira, um bloco de comando é executado; e, caso seja falsa, um outro bloco de comando é executado. Essa estrutura se chama **se então senão**.

Sintaxe:

```
se <condição> então  
    comando 1..... (neste caso a <condição> é verdadeira)  
senão  
    comando 2..... (neste caso a <condição> é falsa)  
fimse
```

O programa, ao encontrar a condição **se**, entende que virá uma expressão cujo resultado é lógico. Caso essa expressão resulte em Verdadeiro, é executado o bloco de comandos entre o **então** e o **senão**, passando a execução para o primeiro comando após o **fimse**. Caso a condição resulte em Falso, é executado o bloco entre o **senão** e o **fimse**.

A estrutura no fluxograma é representada da seguinte forma:

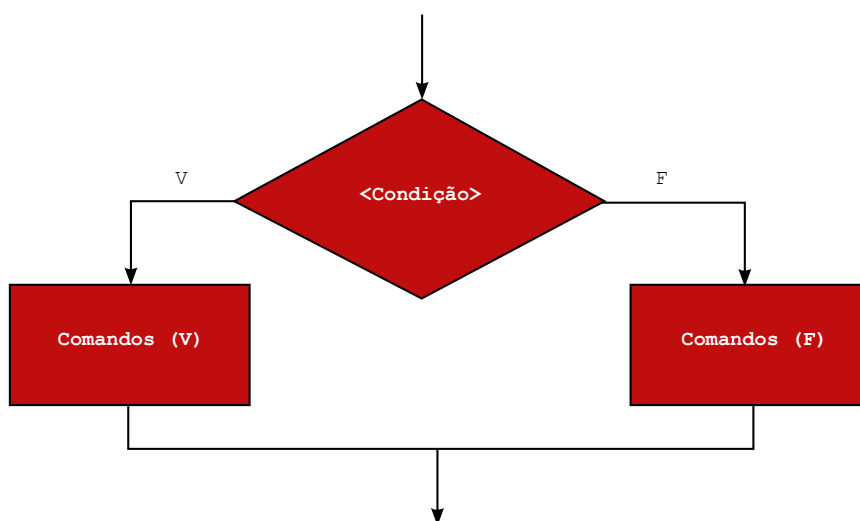


Figura 44 – Fluxograma de decisão com **então** e **senão**

Exemplo de aplicação

Faça um programa que leia um número e diga se é positivo ou negativo.

```
algoritmo "negativo"  
var  
    numero:inteiro  
inicio  
    escreva("Entre com um número")  
    leia(numero)  
    se numero<0 entao  
        escreva(" negativo ")  
    senão  
        escreva(" positivo ")  
    fimse  
finalgoritmo
```

No fluxograma fica claro que, conforme o resultado do teste, o programa toma caminhos diferentes:

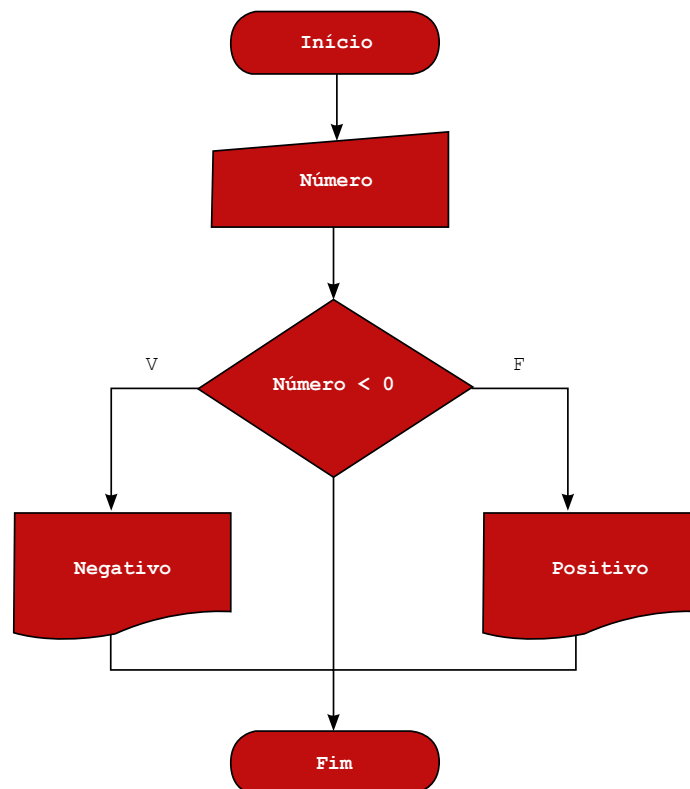


Figura 45 – Decisão com **então** e **senão**

3.2.2.1 Fazendo o teste de mesa

Simulando o papel do computador para duas possibilidades: inicialmente com a entrada 5 e depois com a entrada -5.

Primeiro, um espaço de memória é inicializado para o programa "negativo".

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    senao
        escreva(" positivo ")
    Fimse
Fimalgoritmo
```

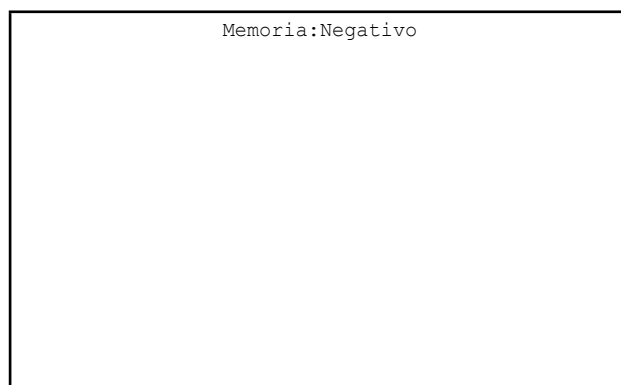


Figura 46 – Espaço de memória do programa *negativo*

As variáveis são separadas na memória. No caso, apenas a variável inteira *numero*.

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    senao
        escreva(" positivo ")
    Fimse
Fimalgoritmo
```

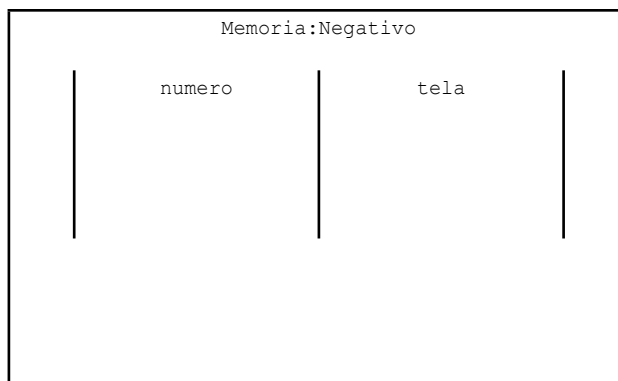


Figura 47 – Espaço de memória mostrando a variável *numero*

A primeira linha do programa apresenta, na tela, o parâmetro do comando de saída **escreva**, no caso, **escreva um *numero***.

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    senao
        escreva(" positivo ")
    Fimse
Fimalgoritmo
```

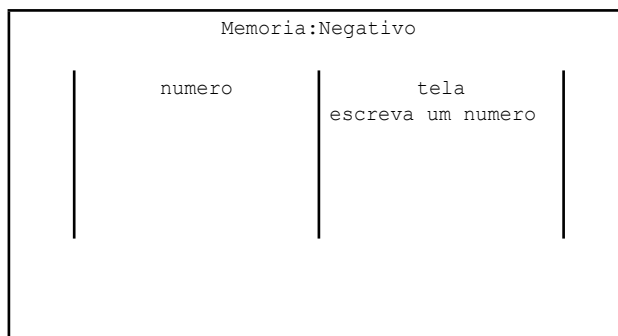


Figura 48 – Memória com tela mostrando o parâmetro do comando de saída do programa

A seguir, é dada a entrada do valor 5, para testar a entrada de um número positivo.

Ao ser dada a entrada, o valor é atribuído na variável *numero*. Assim, o conteúdo de *numero* fica 5.

```
algoritmo "negativo"
Var
    numero:inteiro
```

```
Inicio
  escreva("Entre com um número")
  leia(numero)
  se numero<0 entao
    escreva(" negativo ")
  senao
    escreva(" positivo ")
  Fimse
Fimalgoritmo
```

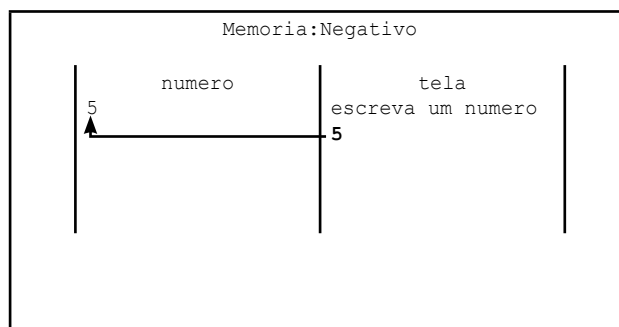


Figura 49 – Atribuição do valor 5 à variável *numero*

Ao encontrar o **se**, o programa avaliará se a operação seguinte resultará em Verdadeiro. No caso, a operação *numero<0* resulta em Falso; assim, o fluxo executará as instruções contidas no bloco do **senão**.

```
algoritmo "negativo"
Var
  numero:inteiro
Inicio
  escreva("Entre com um número")
  leia(numero)
  se numero<0 entao
    escreva(" negativo ")
  senao
    escreva(" positivo ")
  Fimse
Fimalgoritmo
```

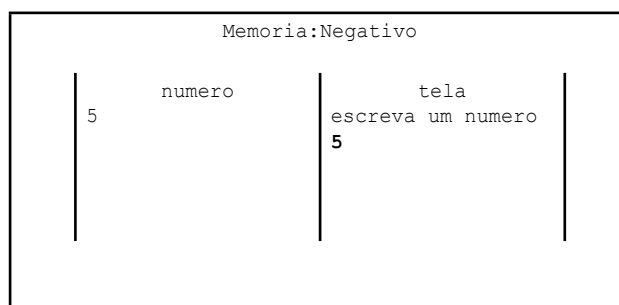


Figura 50 – Como o resultado é Falso, serão executadas as instruções do bloco do **senão**

Executa-se o bloco do **senão**.

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    senao
        escreva(" positivo ")
    Fimse
Fimalgoritmo
```

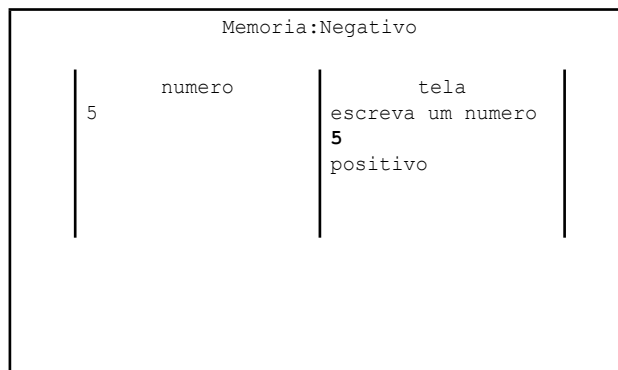


Figura 51 – Memória com tela mostrando o comando *positivo* no bloco do **senão**

A seguir, o programa chega ao final, encerrando a simulação.

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    senao
        escreva(" positivo ")
    Fimse
Fimalgoritmo
```

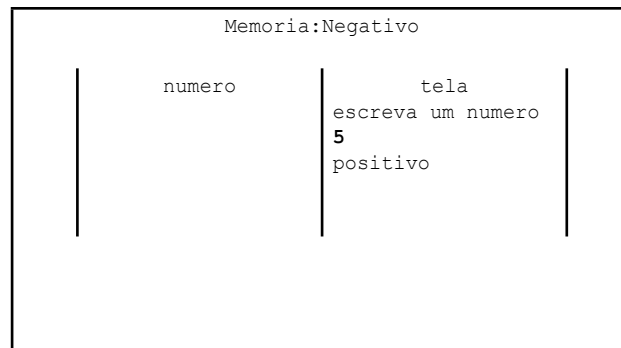


Figura 52 – Fim da simulação

Como esperado, ao entrar com o valor positivo, o programa mostra o resultado *positivo*, pois o bloco executado foi para a avaliação falsa no comando **se**.

A seguir, faremos o teste de mesa para a entrada -5. Como o processamento é o mesmo até a entrada do valor, iniciaremos a partir da linha da entrada do valor (leia).

O valor -5 é digitado no teclado, e, ao aplicarmos a tecla *enter*, a entrada é transferida para a variável *numero*.

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    senao
        escreva(" positivo ")
    Fimse
finalgoritmo
```

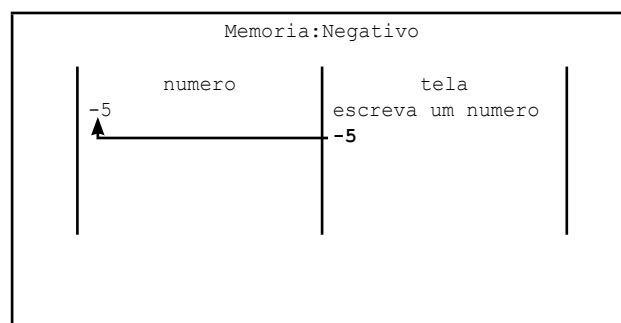


Figura 53 – Atribuição do valor -5 à variável *numero*

Na condicional, o resultado de *numero<0* é Verdadeiro, pois, como a variável contém -5, a comparação é $-5 < 0$, que é verdadeira. Como o resultado é Verdadeiro, o programa irá executar o bloco de comandos do **então**.

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    senao
        escreva(" positivo ")
    Fimse
finalgoritmo
```

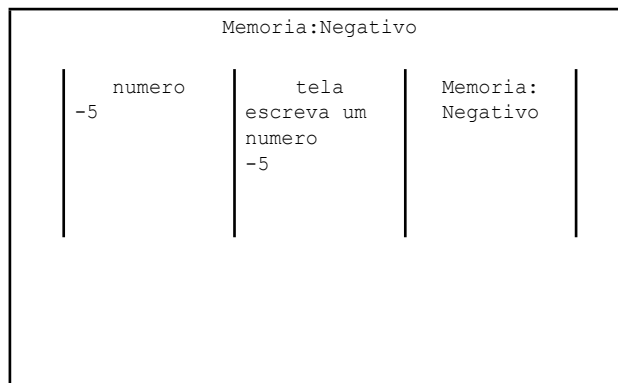


Figura 54 – O bloco de comandos do **então** será executado, pois o resultado é Verdadeiro

No bloco do **então** há apenas um comando, no caso, a saída do texto *negativo*.

```
algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    senao
        escreva(" positivo ")
    Fimse
finalgoritmo
```

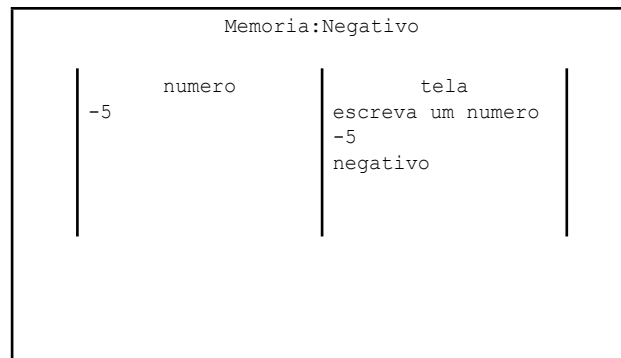



Figura 55 – Memória com tela mostrando o comando *negativo*, no bloco do **então**

A seguir, o programa é encerrado.

```

algoritmo "negativo"
Var
    numero:inteiro
Inicio
    escreva("Entre com um número")
    leia(numero)
    se numero<0 entao
        escreva(" negativo ")
    senao
        escreva(" positivo ")
    Fimse
Fimalgoritmo
    
```

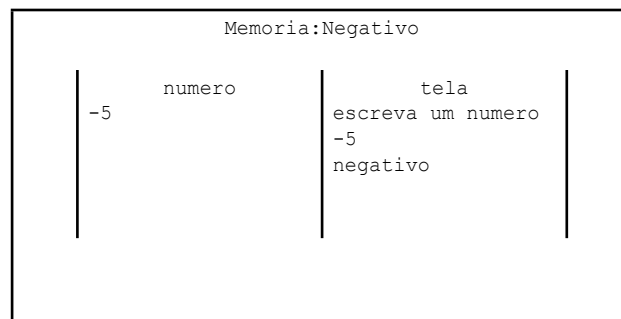


Figura 56 – Encerramento do programa

Assim, ao entrar com o valor -5, o programa mostra que o *numero* é negativo, e ao entrar com o valor maior ou igual a zero, o programa mostra que o número é positivo, portanto a lógica do programa está de acordo com o proposto.

3.2.3 Estrutura condicional se aninhada

Num consultório médico, como se chega a um diagnóstico? O médico inicia fazendo uma série de perguntas; conforme a resposta, vai mudando o seu repertório de perguntas e, com base também nos exames, chega a uma decisão.

Caso o paciente esteja com dor de cabeça, ele pergunta o tipo de dor. Caso seja na nuca, ele pede que o paciente abaixe a cabeça; se este não conseguir, o médico passará a suspeitar de meningite; caso consiga, passará a fazer outras perguntas. Caso a dor seja na região dos olhos, o médico suspeita de sinusite e passa a fazer outras perguntas. Assim, pelo aninhamento de uma série de condicionais, o médico chega a um diagnóstico.

Na programação, dentro de um bloco, os comandos (que, por enquanto, são executados entre o **então** e o **senão**, ou entre o **senão** e o **fimse**) podem ser quaisquer dos aprendidos até agora, inclusive, o próprio **se**.

O aninhamento das condicionais é muito comum e exige um nível maior de raciocínio, portanto uma lógica mais apurada no momento da programação.

Sintaxe:

```
se <condição> então
    se <condição> então
        se <condição> então
            se <condição> então
                .
                .
                .
            fimse
        senão
            .
            .
        fimse
    senão
        .
        .
    fimse
senão
    se <condição> então
        se <condição> então
            .
            .
            .
        fimse
    senão
        .
        .
    fimse
fimse
```

3.2.3.1 Os comandos se dentro de blocos de outros comandos se

Para ilustrar a estrutura do caso que foi apresentado no início, o do médico seria semelhante ao apresentado a seguir:

O paciente reclama de dor de cabeça

```
se dói a nuca então
  se abaixa a cabeça então
    se a boca está torta então
      .
      .
      .
      fimse
    senão
      .
      .
      fimse
  senão
    fazer exame de liquor.
    .
    .
    .
  fimse
senão
  se dói a região dos olhos então
    se já teve sinusite então
      .
      .
      .
    fimse
  senão
    .
    .
  fimse
fimse
```

Assim, a indentação passa a ser fundamental para o entendimento do programa. A seguir, o mesmo código, desta vez, sem a indentação:

O paciente reclama de dor de cabeça

```
se dói a nuca então
se abaixa a cabeça então
```

se a boca está torta então

.

.

.

fimse

senão

fimse

senão

fazer exame de liquor.

.

.

.

fimse

senão

se dói a região dos olhos então

se já teve sinusite então

.

.

.

fimse

senão

.

.

fimse

fimse

No caso de um fluxograma:

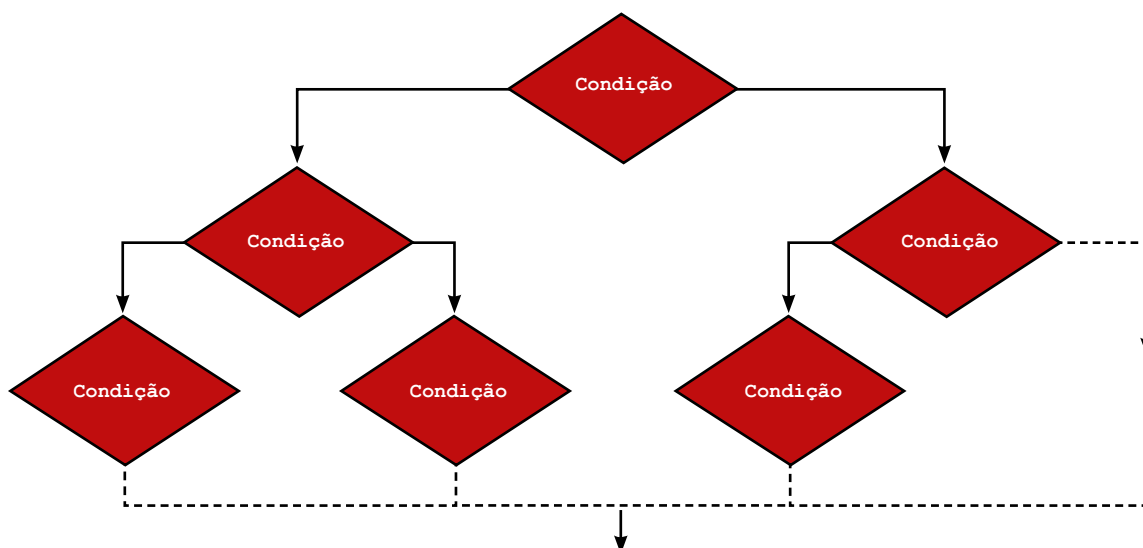


Figura 57 – Fluxograma com condicionais aninhadas

Exemplo de aplicação

Faça um programa que leia três números inteiros diferentes e mostre qual o maior.

```
algoritmo "o maior"
var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    senao
        se n2>n3 entao
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    fimse
fimse
finalgoritmo
```

O fluxograma correspondente ao pseudocódigo será:

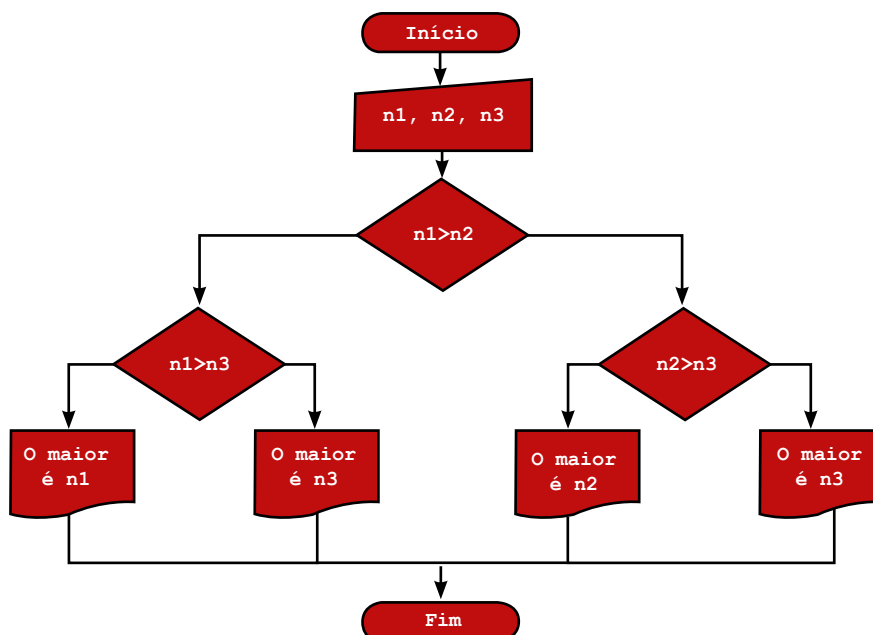


Figura 58 – Programa para encontrar o maior número, de três digitados

Como para todo código é necessário fazer o teste de mesa, faremos os testes para todas as situações possíveis para n_1 , n_2 e n_3 :

- $n_1 = 1$, $n_2 = 2$ e $n_3 = 3$;
- $n_1 = 1$, $n_2 = 3$ e $n_3 = 2$;
- $n_1 = 2$, $n_2 = 1$ e $n_3 = 3$;
- $n_1 = 2$, $n_2 = 3$ e $n_3 = 1$;
- $n_1 = 3$, $n_2 = 1$ e $n_3 = 2$;
- $n_1 = 3$, $n_2 = 2$ e $n_3 = 1$.

Iniciamos o teste com a memória do programa *o maior* limpa.

```
algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
    Fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
    Fimse
Fimse
Fimalgoritmo
```

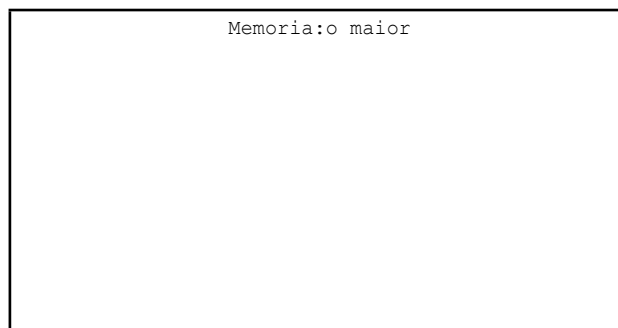


Figura 59 – Início da memória do programa *o maior*

A seguir, criamos as variáveis na memória.

```
algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    Fimse
finalgoritmo
```

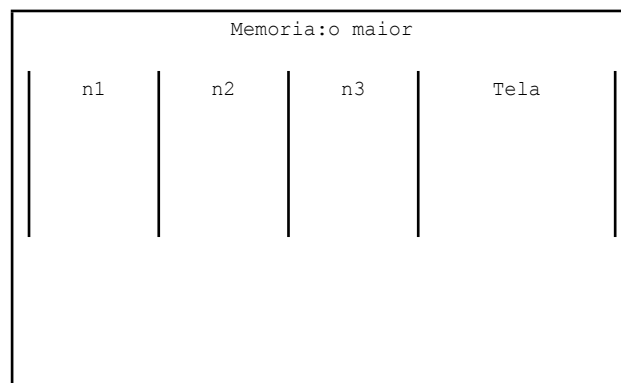


Figura 60 – Espaço de memória com as variáveis criadas

Com o processamento se iniciando, a primeira instrução é a leitura de n1, n2 e n3. No caso, como é o primeiro teste, o valor 1 é armazenado na variável n1, o valor 2, na n2, e o valor 3, na variável n3.

```
algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    Fimse
```

```

Senão
  se n2>n3 entao
    escreva("o maior e ", n2)
  senao
    escreva("o maior e ", n3)
Fimse
Fimse
Fimalgoritmo
    
```

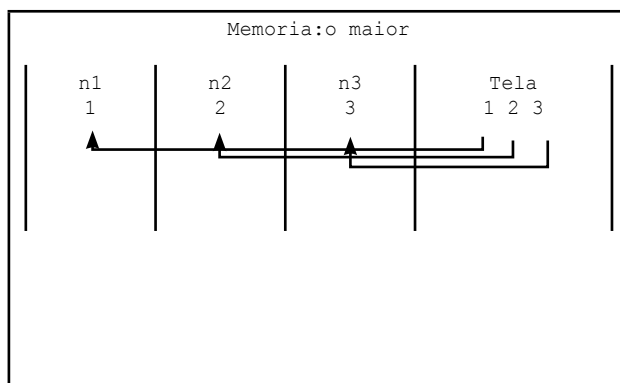


Figura 61 – Atribuição de valores às variáveis

Em seguida é executado o comando condicional. Como o conteúdo de n1 (1) é menor que o conteúdo de n2 (2), a operação $n1 > n2$ resulta em Falso, desviando o curso do programa para o **senão** correspondente ao **se** inicial.

```

algoritmo "o maior"
Var
  n1,n2,n3:inteiro
Inicio
  leia(n1,n2,n3)
  se n1>n2 entao
    se n1>n3 entao
      escreva("o maior e ",n1)
    senao
      escreva("o maior e ",n3)
    fimse
  Senão
    se n2>n3 entao
      escreva("o maior e ", n2)
    senao
      escreva("o maior e ", n3)
    fimse
  Fimse
fimalgoritmo
    
```

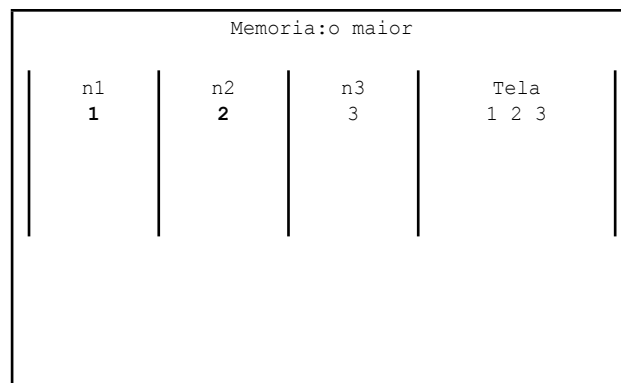



Figura 62 – Execução do comando condicional

Seguindo para o **senão** do primeiro **se**, acontece uma nova condicional. Como o conteúdo de n2 (2) é menor que o conteúdo de n3 (3), a operação $n2 > n3$ resulta em Falso, desviando para o **senão**.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    Fimse
fimalgoritmo
    
```

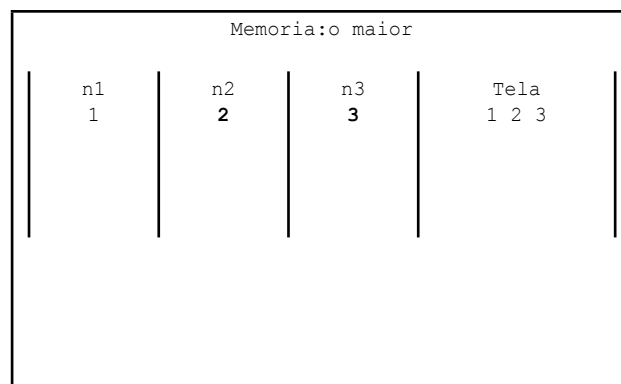


Figura 63 – Execução do segundo comando condicional

No bloco que segue, é dada a saída do valor armazenado na variável n3, ou seja, o valor 3.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    Fimse
Fimalgoritmo
    
```

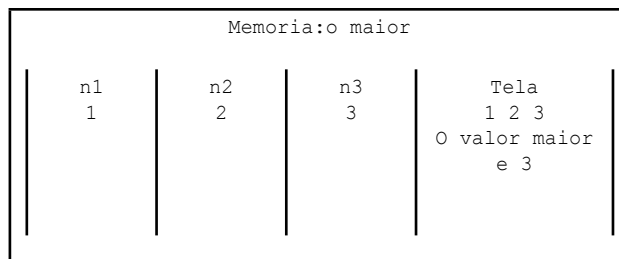


Figura 64 – Saída do valor 3, armazenado na variável n3

Em seguida, o programa somente encontra finais de blocos, encerrando o processamento.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    senao
        se n2>n3 entao
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    Fimse
fimalgoritmo
    
```

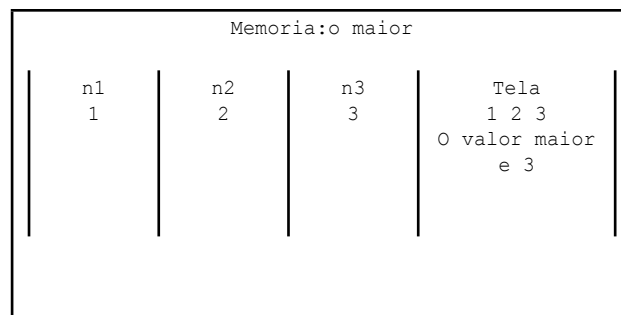


Figura 65 – Fim do processamento

Assim, ao encerrar o programa, foi mostrado o maior valor, conforme o proposto, porém o teste está incompleto, pois as sequências de entrada podem variar de acordo com o arranjo dos números digitados na entrada. Assim, são necessários outros testes.

Agora, fazendo o teste com n1 recebendo 1, n2 recebendo 3 e n3 recebendo 2, a entrada maior está na variável n2.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
        Fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    Fimse
Fimalgoritmo
    
```

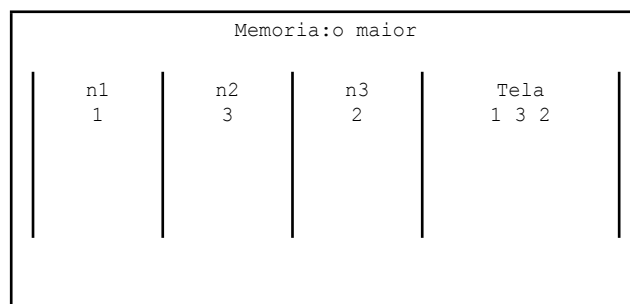


Figura 66 – Programa com a entrada maior atribuída à variável n2

A primeira condicional resulta em Falso, pois o conteúdo de n1 é menor que o conteúdo da variável n2, portanto o fluxo é desviado para o **senão**.

```
algoritmo "o maior"
var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    senao
        se n2>n3 entao
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
        Fimse
    Fimse
fimalgoritmo
```

Memoria:o maior			
n1	n2	n3	Tela
1	3	2	1 3 2

Figura 67 – Como a primeira condicional resultou em Falso, o fluxo vai para o **senão**

No segundo **se**, como o conteúdo de n2 é maior que o de n3, a operação é verdadeira, executando o bloco do **então**.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
        Fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
        Fimse
    Fimse
finalgoritmo
    
```

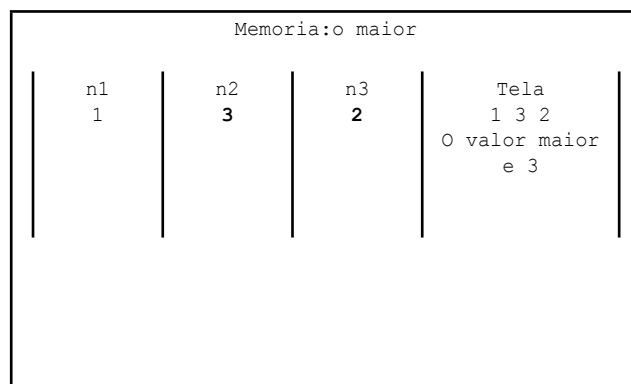


Figura 68 – Como a segunda condicional resultou em Verdadeiro, será executado o bloco do **então**

No bloco do **então** é executado o comando de saída, mostrando o conteúdo da variável n2, no caso, o número 3.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
        Fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    Fimse
Fimalgoritmo
    
```

Memoria:o maior			
n1	n2	n3	Tela
1	3	2	1 3 2
			O valor maior
			e 3

Figura 69 – Memória com tela mostrando o valor 3, da variável n2

Ao encontrar o fim do bloco no **senão**, o programa é encaminhado apenas para os finais de blocos, encerrando-se. Ao final, o programa mostrou novamente o maior número digitado na entrada.

Alterando-se novamente a sequência de entrada, agora na ordem 2, 1 e 3, vamos fazer o teste de mesa. Assim, ao realizar a leitura, a variável n1 recebe o valor 2, a variável n2 recebe 1 e a variável n3 recebe 3.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
    
```

```

        escreva("o maior e ",n1)
    senao
        escreva("o maior e ",n3)
    fimse
Senão
    se n2>n3 entao
        escreva("o maior e ", n2)
    senao
        escreva("o maior e ", n3)
    fimse
Fimse
finalgoritmo

```

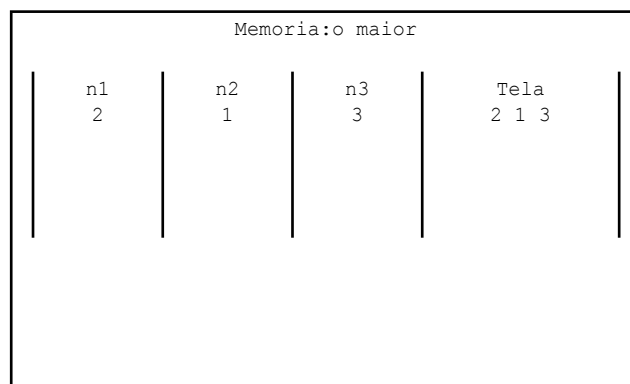


Figura 70 – Atribuição de valores a cada variável

No primeiro **se**, o conteúdo de n1 é maior que o de n2; assim, desta vez, o bloco a ser executado será o **então**.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    Fimse
finalgoritmo

```

Memória:o maior			
n1	n2	n3	Tela
2	1	3	2 1 3

Figura 71 – Execução do bloco do **então**

Com essa sequência de entrada, ao contrário dos testes anteriores, o bloco executado será o do primeiro **então**.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
    se n1>n2 então
        se n1>n3 então
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    Senão
        se n2>n3 então
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    Fimse
finalgoritmo
    
```

Memória:o maior			
n1	n2	n3	Tela
2	1	3	2 1 3

Figura 72 – Execução do bloco do primeiro **então**

No primeiro bloco, o programa encontra a próxima condicional testando as variáveis $n1$ e $n3$. Como o conteúdo da variável $n1$ é maior que o da $n2$, agora é necessário testar com $n3$. No caso, o conteúdo de $n3$ (3) é maior que o de $n1$ (2), assim a operação $n1 > n3$ é falsa, remetendo ao **senão**.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
    ┌───┐
    se n1>n2 entao
    ┌───┐
    se n1>n3 entao
        escreva("o maior e ",n1)
    └───┘
    senao
        escreva("o maior e ",n3)
    fimse
Senão
    se n2>n3 entao
        escreva("o maior e ", n2)
    senao
        escreva("o maior e ", n3)
    fimse
Fimse
finalgoritmo
    
```

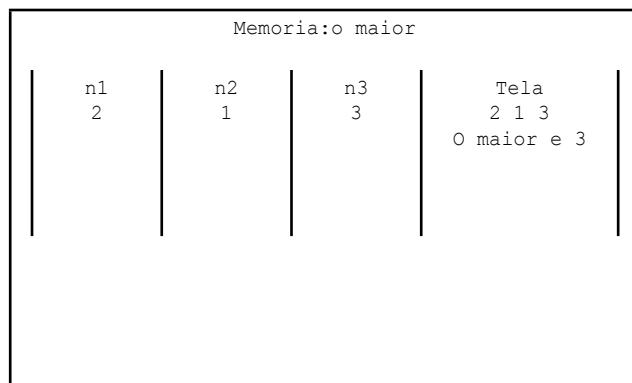


Figura 73 – O resultado Falso leva ao **senão**

Como o conteúdo da variável $n3$ é maior que o de $n1$, que, por sua vez, é maior que o de $n2$, o programa mostra o valor contido em $n3$, no caso, o número 3.

Ao escrever na tela, o bloco é encerrado no **fimse**, pulando do **senão** para o **fimse** e encerrando o programa.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
        ↘
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    Fimse
Fimalgoritmo
    
```

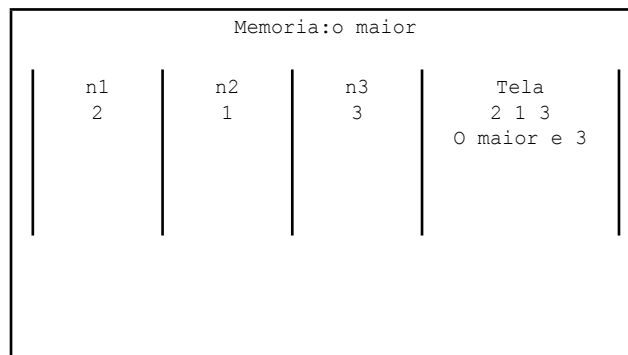


Figura 74 – Término do programa

O programa mostrou o conteúdo da variável n3 que tinha o maior valor digitado na entrada de dados.

O próximo teste será com as entradas 2, 3 e 1, respectivamente, para n1, n2 e n3.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
        Fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        senao
            escreva("o maior e ", n3)
        fimse
    Fimse
Fimalgoritmo
    
```

```

se n2>n3 entao
  escreva("o maior e ", n2)
Senão
  escreva("o maior e ", n3)
Fimse
Fimse
finalgoritmo

```

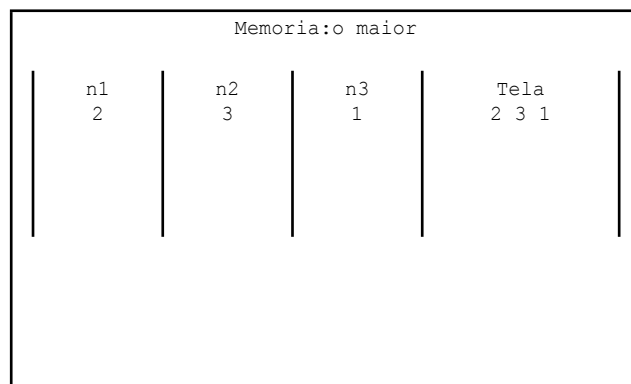


Figura 75 – Definição dos valores para as variáveis

Novamente, como o conteúdo de $n1$ (2) é menor que o conteúdo de $n2$ (3), gerando Falso na operação $n1 > n2$, o programa desvia-se para o **senão**.

```

algoritmo "o maior"
Var
  n1,n2,n3:inteiro
Inicio
  leia(n1,n2,n3)
  se n1>n2 entao
    se n1>n3 entao
      escreva("o maior e ",n1)
    senao
      escreva("o maior e ",n3)
    fimse
  senao
    se n2>n3 entao
      escreva("o maior e ", n2)
    senao
      escreva("o maior e ", n3)
    fimse
  Fimse
finalgoritmo

```

Memoria:o maior			
n1	n2	n3	Tela
2	3	1	2 3 1

Figura 76 – O resultado Falso remete ao **senão**

No bloco do **senão**, como n2 contém 3 e n3 contém 2, a operação $n2 > n3$ fica verdadeira, passando a executar o bloco do **então**.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    senao
        se n2>n3 entao
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
        Fimse
    Fimse
fimalgoritmo
    
```

Memoria:o maior			
n1	n2	n3	Tela
2	3	1	2 3 1

Figura 77 – O resultado Verdadeiro leva à execução do bloco do **então**

No bloco do **então** é executado o comando para escrever que n2 é o maior número.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Início
    leia(n1,n2,n3)
    se n1>n2 então
        se n1>n3 então
            escreva("o maior e ",n1)
        senão
            escreva("o maior e ",n3)
        Fimse
    senão
        se n2>n3 então
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
        Fimse
    Fimse
Fimalgoritmo
    
```

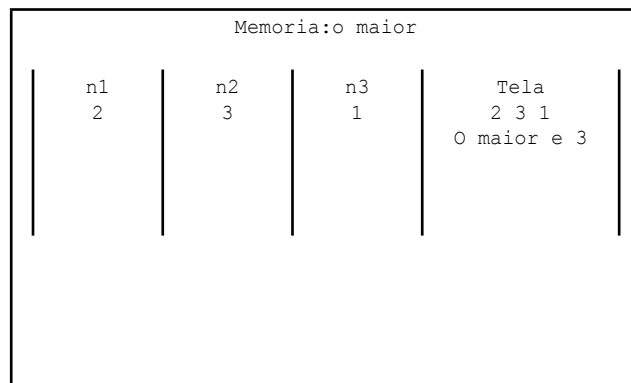


Figura 78 – Execução do comando no bloco do **então**

Revendo a sequência de condicionais, n2 é maior que n1 e n3, portanto n2 é o maior de todos.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
        Fimse
    senao
        se n2>n3 entao
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
        Fimse
    Fimse
fim algoritmo
    
```

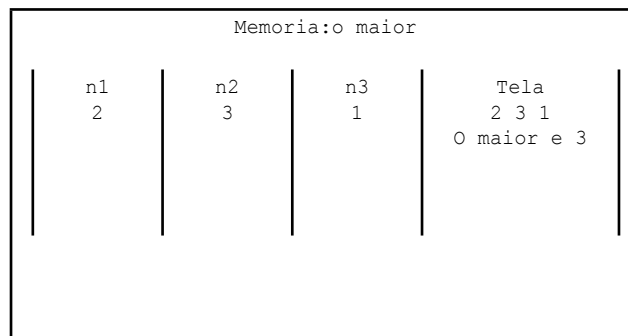


Figura 79 – Memória com tela demonstrando que n2 é a variável com maior valor

Assim, o programa se encerra após todos os blocos serem fechados.

O penúltimo teste é para n1 valendo 3, n2 valendo 1 e n3 valendo 2.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        senao
            escreva("o maior e ",n3)
        fimse
    Senão
    
```

```

se n2>n3 entao
  escreva("o maior e ", n2)
senao
  escreva("o maior e ", n3)
fimse
Fimse
Fimalgoritmo

```

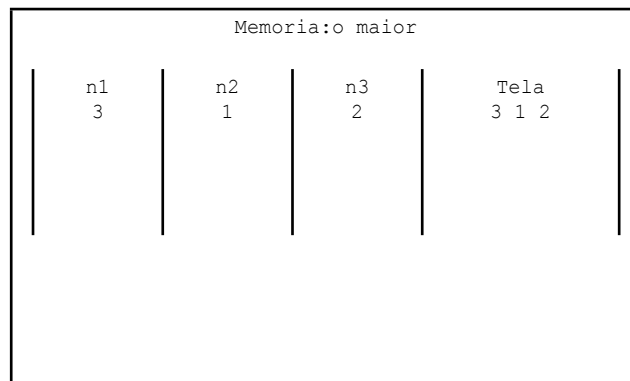


Figura 80 – Definição dos valores para cada variável

Como n1 (3) é maior que n2 (1), a primeira condicional é verdadeira, prosseguindo no bloco do **então**.

```

algoritmo "o maior"
Var
  n1,n2,n3:inteiro
inicio
  leia(n1,n2,n3)
  ┌───┴───>
se n1>n2 entao
  se n1>n3 entao
    escreva("o maior e ",n1)
  senao
    escreva("o maior e ",n3)
  fimse
Senão
  se n2>n3 entao
    escreva("o maior e ", n2)
  senao
    escreva("o maior e ", n3)
  fimse
Fimse
Fimalgoritmo

```

Memoria:o maior			
n1	n2	n3	Tela
3	1	2	3 1 2

Figura 81 – O resultado Verdadeiro da primeira condicional faz o programa prosseguir no bloco do **então**

A segunda condicional testa se $n1$ é maior que $n3$, o que é verdadeiro, pois temos 3 e 2, respectivamente, resultando em Verdadeiro, e o programa prossegue no **então**.

```
algoritmo "o maior"
Var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
        Fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
        Fimse
    Fimse
Fimalgoritmo
```

Memoria:o maior			
n1	n2	n3	Tela
3	1	2	3 1 2

Figura 82 – O resultado Verdadeiro também na segunda condicional mantém o programa no bloco do **então**

Assim, o programa mostra na tela que o maior é o conteúdo de n1, o número 3.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
        Fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
        Fimse
    Fimse
Fimalgoritmo
    
```

Memoria:o maior			
n1	n2	n3	Tela
3	1	2	3 1 2
			O maior e 3

Figura 83 – Tela mostrando que n1 é a variável de maior valor

O programa se fecha com o resultado mostrando que o maior número digitado é 3.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
        ┌───┐
        │   │
se n1>n2 entao
    ┌───┐
    │   │
se n1>n3 entao
    escreva("o maior e ",n1)
Senão
    escreva("o maior e ",n3)
Fimse
Senão
se n2>n3 entao
    escreva("o maior e ", n2)
Senão
    escreva("o maior e ", n3)
Fimse
Fimse
Fimalgoritmo
    
```

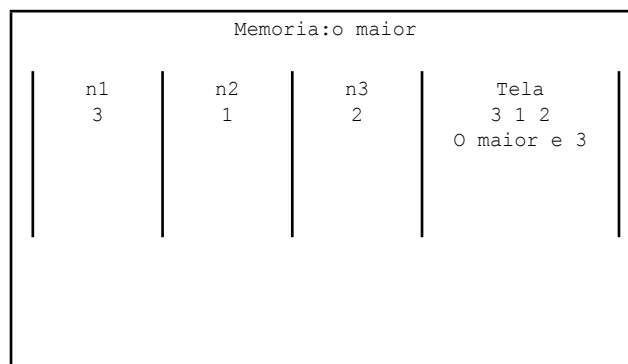


Figura 84 – Programa encerrado

O último teste é para testar a possibilidade de entrarmos com os valores 3, 2, e 1 nas variáveis n1, n2, e n3.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
Inicio
    leia(n1,n2,n3)
se n1>n2 entao
se n1>n3 entao
    escreva("o maior e ",n1)
Senão
    escreva("o maior e ",n3)
Fimse
Fimse
    
```

```

Senão
  se n2>n3 entao
    escreva("o maior e ", n2)
  Senão
    escreva("o maior e ", n3)
  Fimse
Fimse
Fimalgoritmo

```

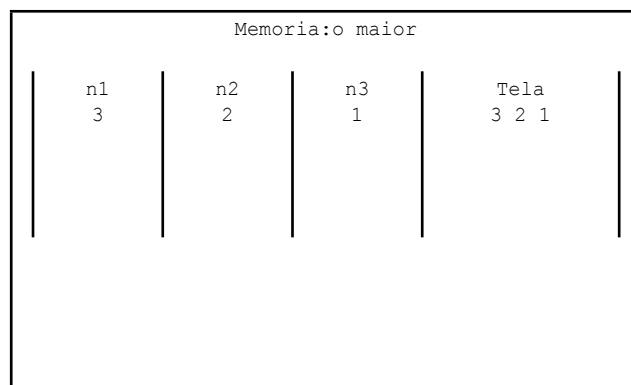


Figura 85 – Determinação de valores para as variáveis

A primeira condicional resulta em Verdadeiro, pois $n1$ é maior que $n2$, $3 > 2$, passando para o bloco do **então**.

```

algoritmo "o maior"
Var
  n1,n2,n3:inteiro
inicio
  leia(n1,n2,n3)
  ┌───┴───>
  se n1>n2 entao
    se n1>n3 entao
      escreva("o maior e ",n1)
    Senão
      escreva("o maior e ",n3)
    Fimse
  Senão
    se n2>n3 entao
      escreva("o maior e ", n2)
    Senão
      escreva("o maior e ", n3)
    Fimse
  Fimse
Fimalgoritmo

```

Memoria:o maior			
n1	n2	n3	Tela
3	2	1	3 2 1

Figura 86 – Como o resultado é Verdadeiro para a primeira condicional, o programa segue no bloco do **então**

O segundo teste é feito verificando se n1 é maior que n3; nesse caso, é verdadeiro também, pois 3 > 1.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
        Fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
        Fimse
    Fimse
Fimalgoritmo
    
```

Memoria:o maior			
n1	n2	n3	Tela
3	2	1	3 2 1

Figura 87 – Como o resultado é Verdadeiro também para o segundo teste, o programa continua no bloco do **então**

O programa escreve então que o maior número é aquele contido na variável n1.

```

algoritmo "o maior"
Var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
        Fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
        Fimse
    Fimse
Fimalgoritmo
    
```

Memoria:o maior			
n1	n2	n3	Tela
3	2	1	3 2 1
			O maior e 3

Figura 88 – Tela demonstrando que a variável n1 é a que possui maior valor

O que acontece nesse teste e no anterior é que n1 é maior que n2 e n3; assim, essa variável é que deverá apresentar-se como a maior entrada.

```
algoritmo "o maior"
Var
    n1,n2,n3:inteiro
inicio
    leia(n1,n2,n3)
    se n1>n2 entao
        se n1>n3 entao
            escreva("o maior e ",n1)
        Senão
            escreva("o maior e ",n3)
        Fimse
    Senão
        se n2>n3 entao
            escreva("o maior e ", n2)
        Senão
            escreva("o maior e ", n3)
        Fimse
    Fimse
Fimalgoritmo
```

Memoria:o maior			
n1	n2	n3	Tela
3	2	1	3 2 1
			O maior e 3

Figura 89 – Tela demonstrando que o valor 3 deverá ser a maior entrada

Assim, foram feitos os testes com todas as combinações possíveis para a entrada de dados. A conclusão é que, em cada uma das possibilidades, o programa tomou um caminho diferente, daí a importância de realizar cada um dos testes, pois pode ocorrer um erro na programação de um dos caminhos, e esse erro estar justamente num caminho que não foi testado.

3.2.4 Escolha-caso

Por enquanto, vimos somente um teste em que uma operação condicional apresenta apenas dois resultados possíveis. Se fosse necessário escolher um resultado dentre vários possíveis, teríamos de encadear várias condicionais.

Existe uma estrutura que evita o encadeamento de condicionais. Essa estrutura é a chamada **escolha-caso**. A estrutura de decisão **escolha-caso** é utilizada quando é necessário testar uma única expressão que produz um resultado dentre vários possíveis, ou, então, testar o valor de uma variável em que está armazenada uma determinada informação. Compara-se o resultado, obtido opção a opção, testando-se com os valores individuais fornecidos de cada cláusula.

Sintaxe:

```
escolha (variável)
caso (opção 1)
    Bloco de Instruções
caso (opção 2)
    Bloco de Instruções
—
—
—
caso (opção n)
    Bloco de Instruções
fimescolha
```

Ou

```
escolha (variável)
caso (opção 1)
    Bloco de Instruções
caso (opção 2)
    Bloco de Instruções
.
.
outrocaso:
    Bloco de Instruções
fimescolha
```

Na estrutura em formato de fluxograma, temos:

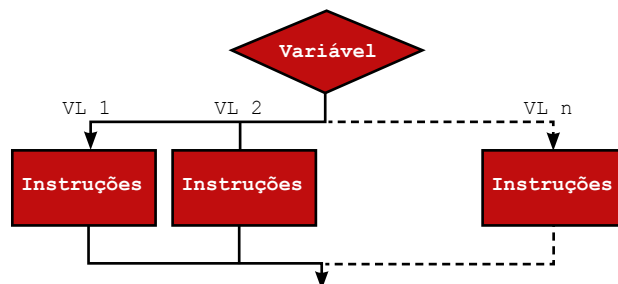


Figura 90 – Escolha-caso

```
escolha (variável)
caso (opção 1), (opção 1.1), (opção 1.2):
    Bloco de Instruções
caso (opção 2), (opção 2.1):
    Bloco de Instruções
.
.
outrocaso:
    Bloco de Instruções
fimescolha
```

Exemplo de aplicação

Fazer uma calculadora simples, que leia dois números e a operação.

```
algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
inicio
    escreva("Digite: numero op numero ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        senao
            escreva("divisao por zero")
        fimse
    outrocaso
        escreva("operacao invalida")
    fimescolha
fimalgoritmo
```

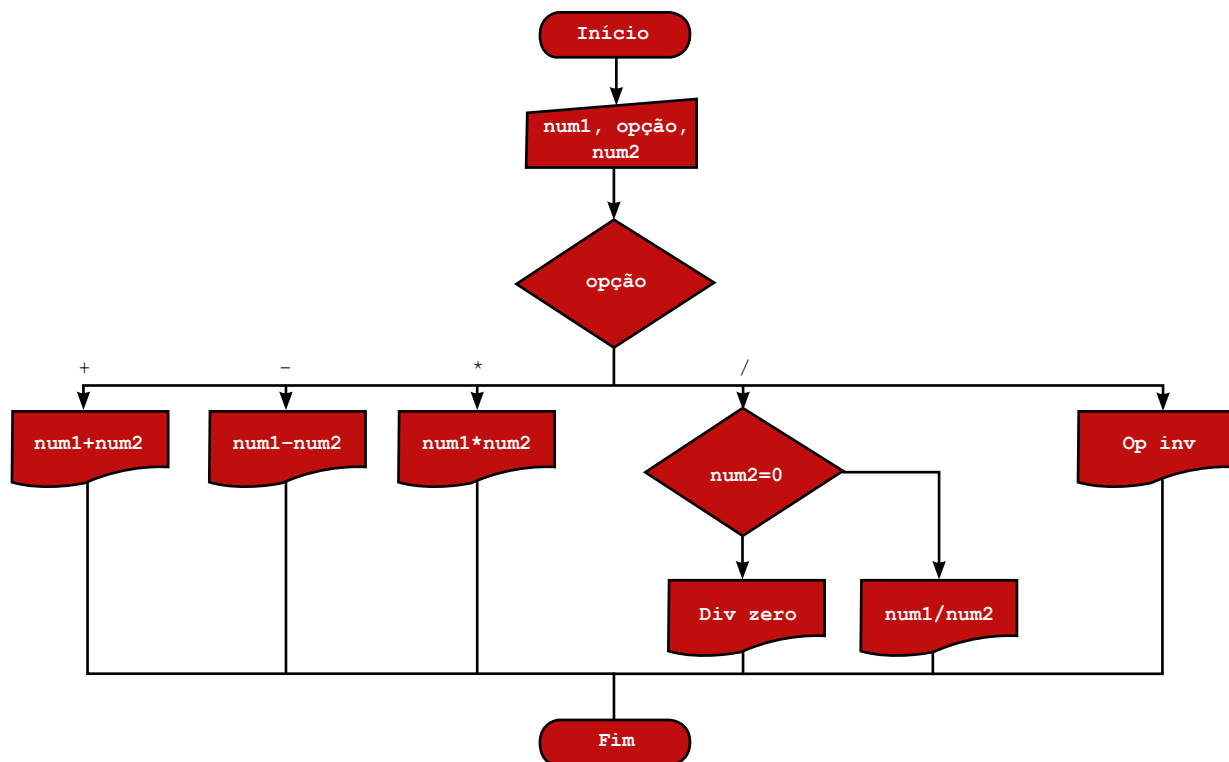



Figura 91 – Calculadora usando escolha-caso

Como sempre, é necessário fazer o teste de mesa.

O computador começa com a memória limpa, em que será executado o programa calc.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
Outrocaso

```

```
    escreva("operacao invalida")
Fimescolha
Fimalgoritmo
```

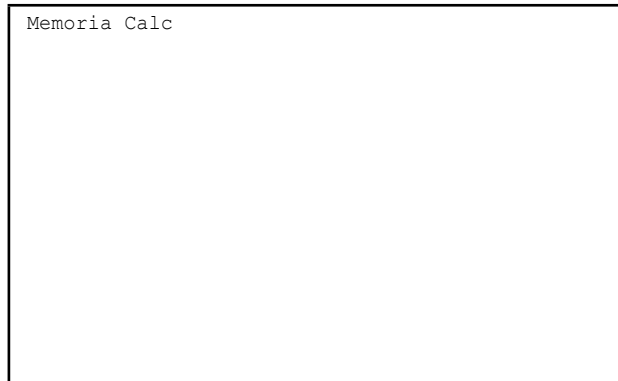


Figura 92 – Memória iniciada para o programa calc

O passo seguinte é criar as variáveis que armazenarão as informações.

```
algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
```

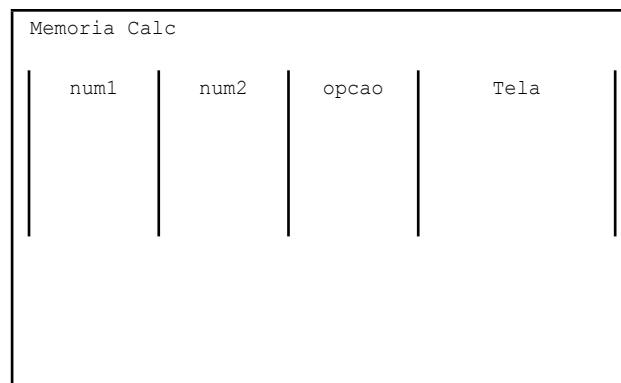


Figura 93 – Criação das variáveis

Iniciando a execução, o primeiro comando é a saída **digite: num op num**, apenas para dizer ao usuário as entradas.

```
algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
```

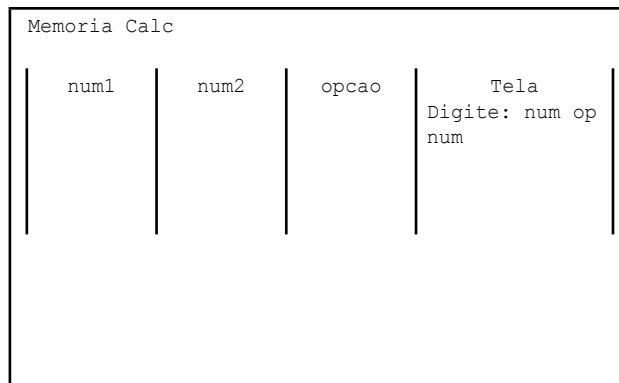


Figura 94 – Execução do primeiro comando

Para testar a soma, será dada a entrada de 2 + 3, sendo o número 2 atribuído à variável num1, o caractere + atribuído à variável opção e o número 3 atribuído à variável num2.

```
algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
```

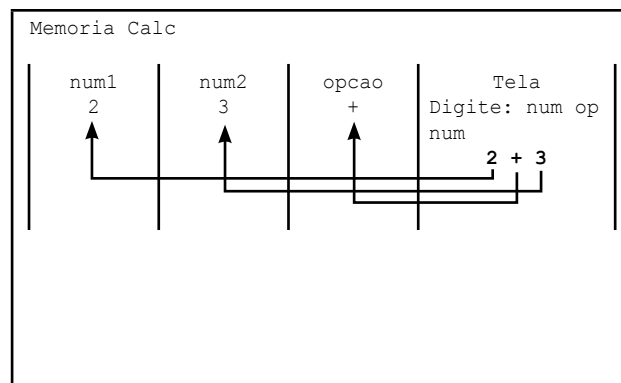


Figura 95 – Atribuição dos dados às variáveis

O comando *escolha* irá comparar cada uma das alternativas sequencialmente até encontrar o correspondente ao conteúdo da variável opção. Nessa situação, encontrará imediatamente no primeiro caso.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
2	3	+	Digite: num op num 2 + 3

Figura 96 – Comparação das alternativas

Uma vez encontrado o bloco correspondente, o programa executa a instrução de saída, a soma entre num1 e num2.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva (num1+num2)
    caso "-"
        escreva (num1-num2)
    caso "*"
        escreva (num1*num2)
    caso "/"
        se num2<>0 entao
            escreva (num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
2	3	+	Digite: num op
			num
			2 + 3
			5

Figura 97 – Execução da instrução de saída

Uma vez encerrado o bloco, o programa pula para o final do *escolha* e, se houvesse outros comandos, seguiria executando sequencialmente; como nesse caso é **finalgoritmo**, encerra o processamento.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo

```

Memoria Calc			
num1	num2	opcao	Tela
2	3	+	Digite: num op num 2 + 3 5

Figura 98 – Término do processamento

Para testar a subtração, será dada a entrada de 2 - 3, sendo o número 2 atribuído à variável num1, o caractere - atribuído à variável opção e o número 3 atribuído à variável num2.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
Outrocaso
    escreva("operacao invalida")
Fimescolha
Fimalgoritmo
    
```

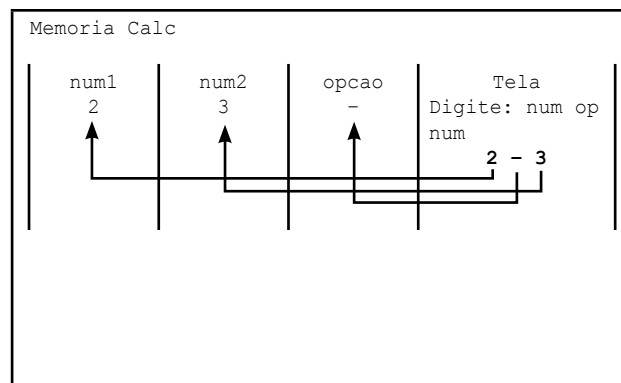



Figura 99 – Distribuição dos dados pelas variáveis

O comando *escolha* irá comparar cada uma das alternativas sequencialmente até encontrar o correspondente ao conteúdo da variável opção. O *escolha* comparará com o primeiro caso, +, e não corresponderá; depois, irá para o segundo caso e encontrará a correspondência.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
2	3	-	Digite: num op num 2 - 3

Figura 100 – Comparação das alternativas até encontrar correspondência

Uma vez encontrado o bloco correspondente, o programa executa a instrução de saída, a subtração entre num1 e num2.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
2	3	-	Digite: num op num 2 - 3 - 1

Figura 101 – Execução da instrução de saída no bloco correspondente

Uma vez encerrado o bloco, o programa pula para o final do *escolha* e, caso houvesse outros comandos, seguiria executando; como, no caso, é **finalgoritmo**, encerra o processamento.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo

```

Memoria Calc				
num1	num2	opcao	Tela	
2	3	-	Digite: num op	
			num	
			2 - 3	
			- 1	

Figura 102 – Finalização do programa

Para testar a multiplicação, será dada a entrada de $2 * 3$, sendo o número 2 atribuído à variável num1, o caractere * atribuído à variável opção e o número 3 atribuído à variável num2.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
Outrocaso
    escreva("operacao invalida")
Fimescolha
Fimalgoritmo
    
```

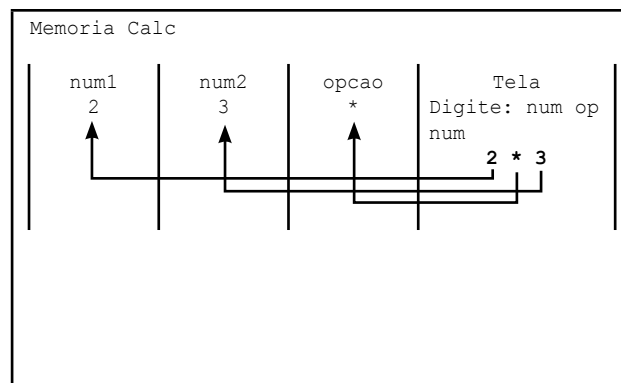


Figura 103 – Atribuindo os dados a cada variável

O comando *escolha* irá comparar cada uma das alternativas sequencialmente até encontrar o correspondente ao conteúdo da variável opção. O *escolha* comparará com o primeiro caso, +, e não corresponderá; comparará com o segundo caso, -, que também não corresponderá; e encontrará a correspondência apenas no terceiro caso.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
2	3	*	Digite: num op num 2 * 3

Figura 104 – Comparação das alternativas disponíveis

Uma vez encontrado o bloco correspondente, o programa executa a instrução de saída, a multiplicação entre num1 e num2.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
Outrocaso
    escreva("operacao invalida")
Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
2	3	*	Digite: num op num 2 * 3 6

Figura 105 – Execução da instrução de saída

Encerrado o bloco, o programa pula para o final do *escolha* e, caso houvesse outros comandos, seguiria executando; como, no caso, é **finalgoritmo**, encerra o processamento.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva (num1+num2)
    caso "-"
        escreva (num1-num2)
    caso "*"
        escreva (num1*num2)
    caso "/"
        se num2<>0 entao
            escreva (num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo

```

Memoria Calc				
num1	num2	opcao	Tela	
2	3	*	Digite: num op	
			num	
			2 * 3	
			6	

Figura 106 – Término do programa

Na divisão existem duas situações possíveis: a primeira é com o divisor diferente de zero, pois, no caso de divisão por zero, o sistema deverá indicar erro. Para a divisão será dada a entrada de 8 / 2, sendo o número 8 atribuído à variável num1, o caractere / atribuído à variável opção e o número 2 atribuído à variável num2.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

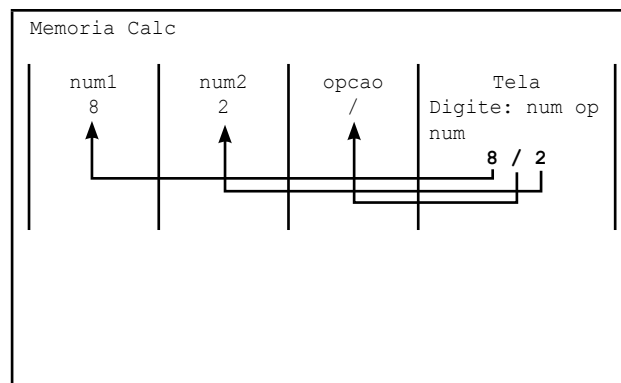



Figura 107 – Distribuição dos dados pelas variáveis

O comando *escolha* irá comparar cada uma das alternativas sequencialmente até encontrar o correspondente ao conteúdo da variável opção. O *escolha* comparará com o primeiro caso, +, e não corresponderá; e comparará com o segundo caso, -, até encontrar o bloco com /.

```

algoritmo "calc"
Var
    num1, num2: real
    opcao:caracter
Inicio
    escreva ("Digite: num op num ")
    leia (num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva (num1+num2)
    caso "-"
        escreva (num1-num2)
    caso "*"
        escreva (num1*num2)
    caso "/"
        se num2<>0 entao
            escreva (num1/num2)
        Senão
            escreva ("divisao por zero")
    Fimse
Outrocaso
    escreva ("operacao invalida")
Fimescolha
Fimalgoritmo
    
```

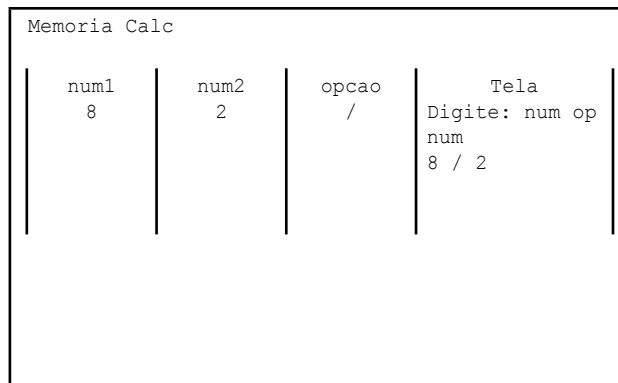


Figura 108 – Comparação das alternativas até encontrar correspondência

Uma vez encontrado o bloco da divisão, o programa, inicialmente, verifica se o divisor, ou seja, a variável num2, é diferente de zero. Como no caso a operação resultará em Verdadeiro, pois num2 contém o valor 2, será executado o bloco do **então**, nesse caso, a instrução de saída, a divisão entre num1 e num2.

```

algoritmo "calc"
Var
    num1, num2: real
    opcao:caracter
Inicio
    escreva ("Digite: num op num ")
    leia (num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva (num1+num2)
    caso "-"
        escreva (num1-num2)
    caso "*"
        escreva (num1*num2)
    caso "/"
        se num2<>0 entao
            escreva (num1/num2)
        Senão
            escreva ("divisao por zero")
        Fimse
    Outrocaso
        escreva ("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
8	2	/	Digite: num op num 8 / 2 4

Figura 109 – Executando a instrução de saída

Uma vez encerrado o bloco do **então**, o programa pula para o **fimse**, prosseguindo para o final do *escolha*, e encerra o processamento.

```

algoritmo "calc"
Var
    num1, num2: real
    opcao:caracter
Inicio
    escreva ("Digite: num op num ")
    leia (num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva (num1+num2)
    caso "-"
        escreva (num1-num2)
    caso "*"
        escreva (num1*num2)
    caso "/"
        se num2<>0 entao
            escreva (num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
8	2	/	Digite: num op num 8 / 2 4

Figura 110 – Programa encerrado

A segunda possibilidade, na divisão, é o divisor igual a zero, caso em que o sistema deverá indicar erro. Para a divisão, será dada a entrada de 8 / 0, sendo o número 8 atribuído à variável num1, o caractere / atribuído à variável opção e o número 0 atribuído à variável num2.

```
algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
Outrocaso
    escreva("operacao invalida")
Fimescolha
Fimalgoritmo
```

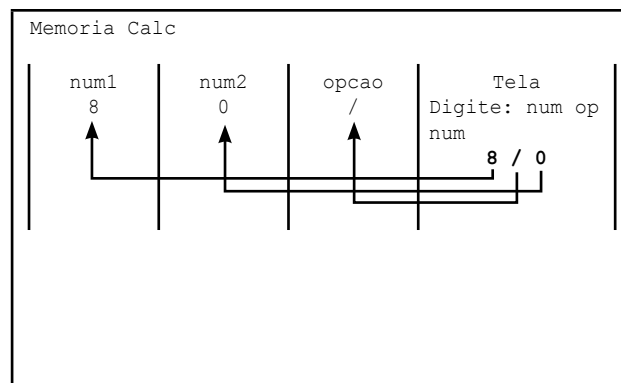


Figura 111 – Determinação de dados para as variáveis

O comando *escolha* irá comparar cada uma das alternativas sequencialmente até encontrar o correspondente ao conteúdo da variável opção. O *escolha* comparará com o primeiro caso, +, e não corresponderá; e comparará com o segundo caso, -, até encontrar o bloco com /.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
8	0	/	Digite: num op num 8 / 0

Figura 112 – Comparação buscando correspondência

Uma vez encontrado o bloco da divisão, o programa, inicialmente, verifica se o divisor, ou seja, a variável num2, é diferente de zero. Como no caso a operação resultará em Falso, pois num2 contém o valor 0, será executado o bloco do **então**, nesse caso, a instrução de saída, a mensagem de erro.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
8	0	/	Digite: num op num 8 / 0 Divisão por zero

Figura 113 – Exibição da mensagem de erro

Encerrado o bloco do **então**, o programa prossegue para o final do *escolha* e finaliza o processamento.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva (num1+num2)
    caso "-"
        escreva (num1-num2)
    caso "*"
        escreva (num1*num2)
    caso "/"
        se num2<>0 entao
            escreva (num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
8	0	/	Digite: num op num 8 / 0 4

Figura 114 – Finalização do programa

Ainda falta um teste a ser realizado. O usuário pode digitar uma operação inválida, que não é nem soma, nem subtração, nem multiplicação, nem divisão. Vamos testar com %.

```
algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
    Fimse
Outrocaso
    escreva("operacao invalida")
Fimescolha
Fimalgoritmo
```

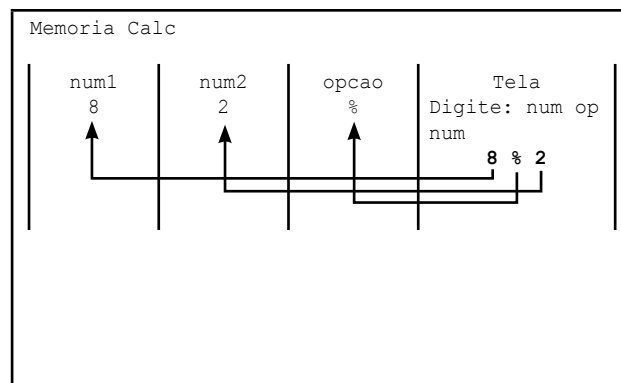



Figura 115 – Teste com %

O comando *escolha* irá comparar cada uma das alternativas sequencialmente até encontrar o correspondente ao conteúdo da variável opção. O *escolha* comparará com o primeiro caso, +, e não corresponderá; comparará com o segundo caso, -, e assim por diante; porém, como não encontrará o correspondente, o programa irá executar o bloco **outrocaso**, que é a saída da mensagem **operação inválida**.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva(num1+num2)
    caso "-"
        escreva(num1-num2)
    caso "*"
        escreva(num1*num2)
    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo
    
```

Memoria Calc			
num1	num2	opcao	Tela
8	2	%	Digite: num op num 8 % 2 Operação invalida

Figura 116 – Exibição da mensagem "operação inválida", por não haver correspondente entre as alternativas

Encerrado o bloco do **então**, o programa fecha o bloco do *escolha*, finalizando o processamento.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
Inicio
    escreva("Digite: num op num ")
    leia(num1, opção, num2)
    escolha (opcao)
    caso "+"
        escreva (num1+num2)
    caso "-"
        escreva (num1-num2)
    caso "*"
        escreva (num1*num2)
    caso "/"
        se num2<>0 entao
            escreva (num1/num2)
        Senão
            escreva("divisao por zero")
        Fimse
    Outrocaso
        escreva("operacao invalida")
    Fimescolha
Fimalgoritmo

```

Memoria Calc			
num1	num2	opcao	Tela
8	2	%	Digite: num op
			num
			8 % 2
			4

Figura 117 – Processamento encerrado



Saiba mais

No fim da década de 1970, surgiu uma nova linguagem de programação voltada para o ensino, principalmente, de crianças e adolescentes. Essa linguagem se chama Logo. Versões desse programa ainda podem ser encontradas gratuitamente na internet. Para aqueles que tiverem a curiosidade, o *site* xLogo oferece gratuitamente o manual e o programa:

<<http://xlogo.tuxfamily.org/pt/>>.

3.3 Laboratório

Na linguagem C, o comando correspondente ao **se** é o *if*.

Sintaxe:

```
if (expr) {
    bloco de comandos 1
}
```

A sintaxe corresponde à estrutura:

```
se <condição> então
    bloco de comandos 1
fimse
```

Deve ficar claro que, na linguagem C, se a condição for verdadeira, o programa irá executar o próximo comando ou bloco.

Assim:

```
if (true)
{
    bloco de comandos 1
}
```



Observação

Na linguagem C, o inteiro 0 (zero), nas condicionais, é tratado como *false*. Normalmente, o 1 (um) é considerado *true*, mas, na realidade, a condicional considera verdadeiro qualquer valor diferente de zero.

Exemplo de aplicação

Converta o pseudocódigo do programa desenvolvido anteriormente, que mostrava na tela caso o número informado fosse negativo para a linguagem C++.

<pre>algoritmo "negativo" var numero:inteiro inicio escreva("Entre com um número") leia(numero) se numero<0 entao escreva("O numero negativo ") fimse finalgoritmo</pre>	<pre>#include <stdio.h> void main(){ ..int numero; ..printf("Entre com um numero: "); scanf("%d",&numero); ..if (numero<0){ ...printf("\nO numero negativo \n"); } }</pre>
---	--

Ao executarmos o programa com a entrada 5, temos a seguinte saída:

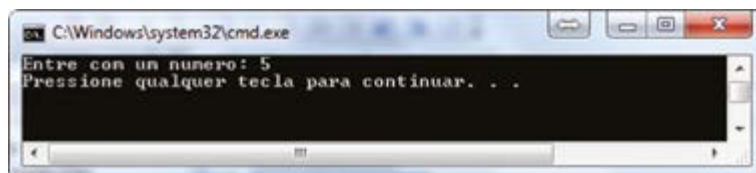


Figura 118 – Saída do programa caso a entrada seja positiva

Executando com -5 na entrada, temos a seguinte saída:

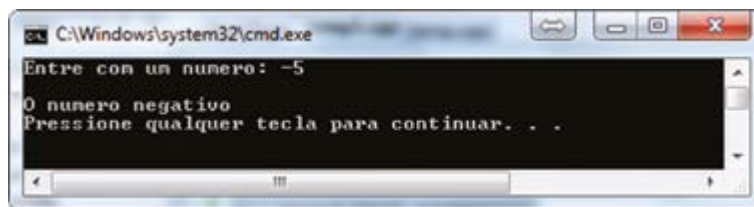


Figura 119 – Saída do programa com uma entrada negativa



Observação

Um erro comum é colocar o símbolo de fim de comando (;) após a condicional, anulando a estrutura do *if*.

```
if (true) ;  
{  
    bloco de comandos 1  
}
```

if encerrado

No caso da estrutura **se então senão**, a sintaxe será:

```
if ( expr ) {  
    bloco de comandos 1  
}  
else {  
    bloco de comandos 2  
}
```

Se o resultado *expr* for 1 (verdadeiro), o bloco de comandos 1 será executado. A inclusão do complemento *e/se* requisita a execução do bloco de comandos 2, ou seja, caso o resultado *expr* seja o valor 0 (Falso). Cada bloco de comandos deve ser delimitado por uma chave aberta e outra fechada. Se dentro de um bloco tivermos apenas um comando a ser executado, as chaves poderão ser omitidas (na verdade, deixaremos de ter um bloco).

A indentação (recuo de linha) dos comandos é fundamental para uma maior clareza do código. O estilo de indentação varia a gosto do programador. Além da forma ilustrada anteriormente, outro estilo bastante utilizado por programadores C é:

```
if ( expr )
{
    bloco de comandos 1
}
else {
    bloco de comandos 2
}
```

A estrutura corresponde, em pseudocódigo, a:

```
se <condição> entao
    bloco de comandos 1
senao
    bloco de comandos 2
fimse
```

Vejamos mais um exemplo.

Exemplo de aplicação

Converta o pseudocódigo do programa que mostra caso um número digitado seja positivo ou negativo para a linguagem C++.

<pre>algoritmo "negativo" var numero:inteiro inicio escreva("Entre com um número") leia(numero) se numero<0 entao escreva(" negativo ") senão escreva(" positivo ") fimse finalgoritmo</pre>	<pre>#include <stdio.h> void main() { ..int numero; printf("Entre com um numero: "); scanf("%d",&numero); if (numero<0) { printf("\n0 numero negativo \n"); } else { printf("\n0 numero positivo \n"); } }</pre>
---	---

Na saída para o valor 5 como entrada, teremos:



Figura 120 – Saída do programa com *else*, e entrada positiva

Como saída para o valor -5, teremos:

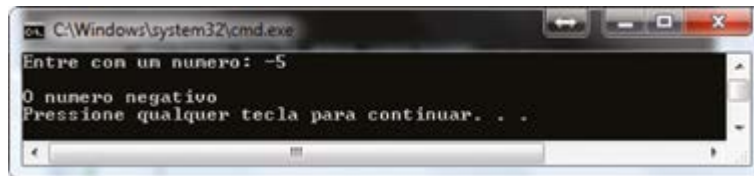


Figura 121 – Saída do programa com *else*, e entrada negativa

Como os blocos do **então** e do **senão**, nesse programa, possuem apenas um comando, o programa pode ser escrito da seguinte maneira:

```
#include <stdio.h>
void main(){
    int numero;
    printf("Entre com um numero: ");
    scanf("%d",&numero);
    if (numero<0)
        printf("\nO numero negativo \n");
    else
        printf("\nO numero positivo \n");
}
```

Podemos aninhar comandos *if*. Um exemplo simples é ilustrado a seguir.

```
if ( expr ) {
    if ( expr2 ) {
        .
        .
        .
    }
    else
    {
        .
    }
}
```

```

        .
        .
    }
}
else {
    if ( expr3 ) {
        .
        .
        .
    }
    else
    {
        .
        .
        .
    }
}
}

```

Assim, podemos tomar como exemplo o raciocínio desenvolvido nos pseudocódigos fazendo-se a simples tradução para a linguagem C.

Exemplo de aplicação

Converta o pseudocódigo do programa que mostra o maior número dentre os três digitados para a linguagem C++.

<pre> algoritmo "o maior" var n1,n2,n3:inteiro inicio leia(n1,n2,n3) se n1>n2 entao se n1>n3 entao escreva("o maior e ",n1) senao escreva("o maior e ",n3) fimse senao se n2>n3 entao escreva("o maior e ", n2) senao escreva("o maior e ", n3) fimse fimse fimse fimalgoritmo </pre>	<pre> #include <stdio.h> void main() { int n1,n2,n3; scanf("%d %d %d",&n1,&n2,&n3); if (n1>n2) if (n1>n3) printf("\no maior e %d\n", n1); else printf("\no maior e %d\n", n3); else if (n2>n3) printf("\no maior e %d\n", n2); else printf("\no maior e %d\n", n3); } </pre>
--	---

Como foi feito nos testes de mesa, agora testaremos na prática as diversas entradas.

- Entradas 1, 2, 3:

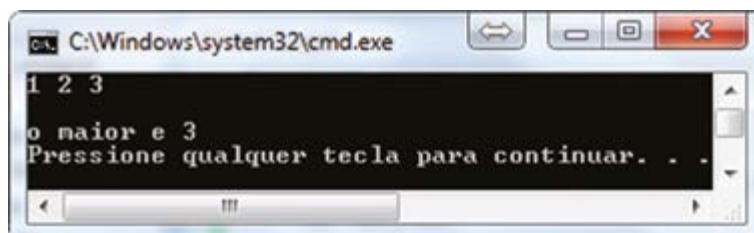


Figura 122 – Saída do programa o maior número com entradas 1, 2, 3

- Entradas 1, 3, 2:

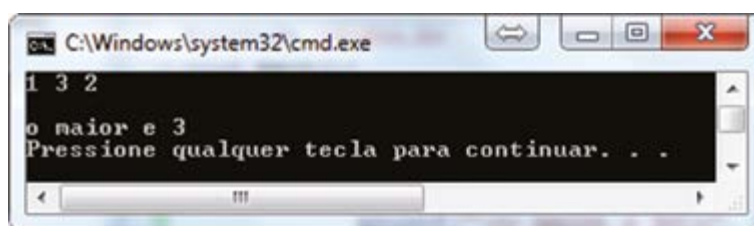


Figura 123 – Saída do programa o maior número com entradas 1, 3, 2

- Entradas 2, 1, 3:



Figura 124 – Saída do programa o maior número com entradas 2, 1, 3

- Entradas 2, 3, 1:

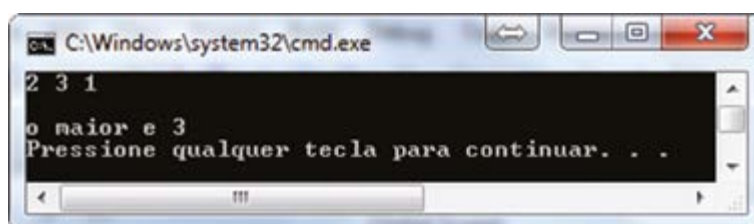


Figura 125 – Saída do programa o maior número com entradas 2, 3, 1

- Entradas 3, 1, 2:

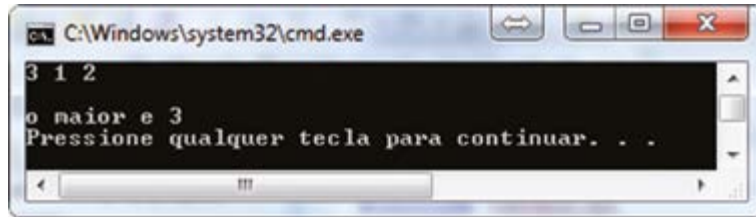


Figura 126 – Saída do programa o maior número com entradas 3, 1, 2

- Entradas 3, 2, 1:

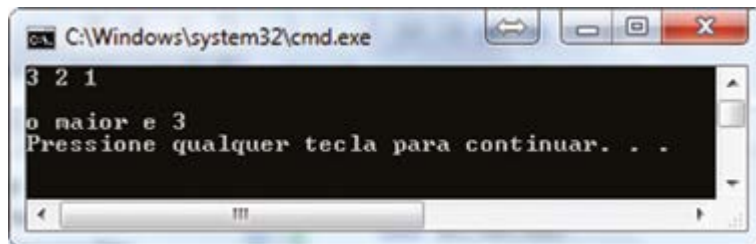


Figura 127 – Saída do programa o maior número com entradas 3, 2, 1

3.3.1 Escolha-caso na linguagem C++

O comando (*switch*) para selecionar um dentre um conjunto de possíveis casos corresponde ao *escolha* do pseudocódigo. Sua forma geral é:

```
switch ( expr )
{
case op1:
... /* comandos executados se expr == op1 */
    break;
case op2:
... /* comandos executados se expr == op2 */
    break;
case op3:
... /* comandos executados se expr == op3 */
    break;
default:
... /* executados se expr for diferente de todos */
    break;
}
```

Opcao1 deve ser um número inteiro ou uma constante caractere. Se *expressao* tiver o mesmo valor de *opcao1*, o bloco de comandos que se segue ao caso *opcao1* será executado, até que seja encontrado o comando *break*. Caso o comando *break* seja omitido, a execução dos comandos do bloco do *case* continuará com os comandos do *case* seguinte. O caso *default* (nenhum dos outros) pode aparecer em qualquer posição, mas, para facilitar a leitura, é colocado por último.

Vamos tomar como exemplo o raciocínio desenvolvido nos pseudocódigos da construção de uma calculadora fazendo a simples tradução para a linguagem C.

Exemplo de aplicação

Fazer uma calculadora simples, que leia dois números e a operação.

```

algoritmo "calc"
Var
    num1,num2: real
    opcao:caracter
inicio
    escreva("Digite: numero op nume-
ro ")
    leia(num1, opção, num2)
    escolha (opcao)

    caso "+"
        escreva(num1+num2)

    caso "-"
        escreva(num1-num2)

    caso "*"
        escreva(num1*num2)

    caso "/"
        se num2<>0 entao
            escreva(num1/num2)
        senao
            escreva("divisao por zero")
        fimse
    outrocaso
        escreva("operacao invalida")

    fimescolha
fimalgoritmo
    
```

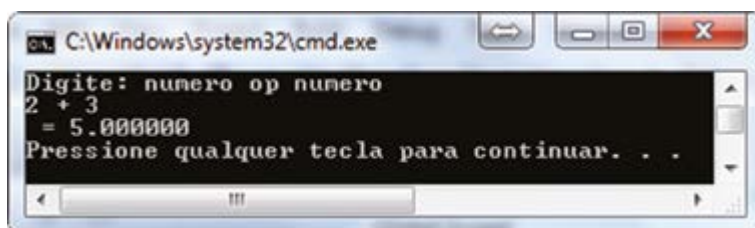
```

#include <stdio.h>
void main ()
{
    float num1, num2;
    char op;

    printf("Digite: numero op nume-
ro\n");
    scanf ("%f %c %f", &num1, &op,
    &num2);
    switch (op)
    {
        case '+':
            printf(" = %f\n", num1+num2);
            break;
        case '-':
            printf(" = %f\n", num1-num2);
            break;
        case '*':
            printf(" = %f\n", num1*num2);
            break;
        case '/':
            if (num2 != 0)
                printf("%10.5f\n",num1/num2);
            else
                printf("divisao por zero\n");
            break;
        default:
            printf("Operador invalido!\n");
            break;
    }
}
    
```

Executando o programa para as diversas entradas.

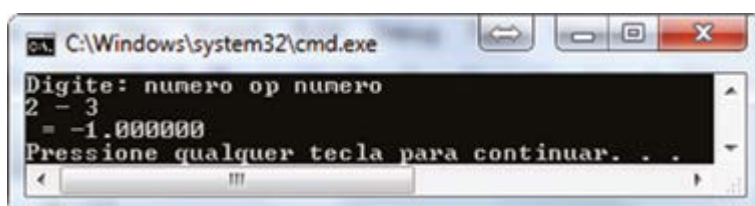
- Soma:



```
C:\Windows\system32\cmd.exe
Digite: numero op numero
2 + 3
= 5.000000
Pressione qualquer tecla para continuar. . .
```

Figura 128 – Calculando 2 + 3

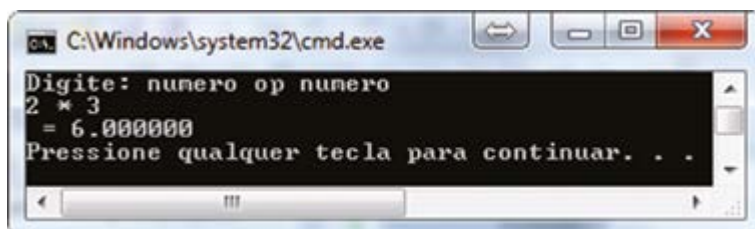
- Subtração:



```
C:\Windows\system32\cmd.exe
Digite: numero op numero
2 - 3
= -1.000000
Pressione qualquer tecla para continuar. . .
```

Figura 129 – Calculando 2 - 3

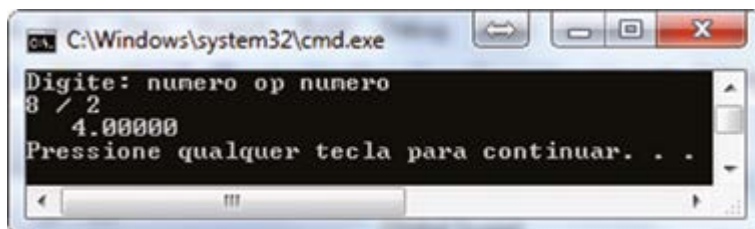
- Multiplicação:



```
C:\Windows\system32\cmd.exe
Digite: numero op numero
2 * 3
= 6.000000
Pressione qualquer tecla para continuar. . .
```

Figura 130 – Calculando 2 * 3

- Divisão:



```
C:\Windows\system32\cmd.exe
Digite: numero op numero
8 / 2
= 4.000000
Pressione qualquer tecla para continuar. . .
```

Figura 131 – Calculando 8/2

- Divisão por zero:

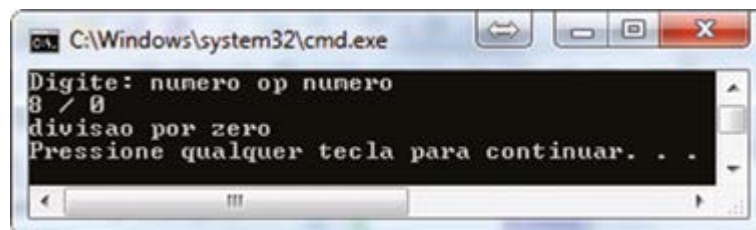


Figura 132 – Calculando 8/0

- Operação inválida:

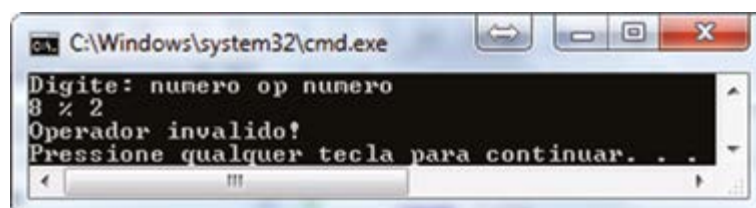


Figura 133 – Calculando 8 % 2

Na linguagem C, temos mais dois casos de condicionais.

O primeiro caso é o chamado *if-else-if*, que é um complemento do *if-else*.

O funcionamento é idêntico ao do *if-else*, mas a estrutura é visualmente próxima à do *switch-case*.

```
if (condição1) {bloco de comandos 1};  
else if (condição 2) bloco {bloco de comandos 2};  
else if (condição 3) bloco {bloco de comandos 3};  
.  
.  
else {bloco de default};
```

O segundo caso é o operador *?*. Trata-se de uma condicional que devolve um valor e envolve três expressões:

```
condição?expressão 1:expressão 2;
```

O operador funciona da seguinte forma: se a condição resultar em Verdadeiro, retornará o resultado da expressão 1; se a condição for falsa, retornará o resultado da expressão 2.



Lembrete

Vimos a possibilidade de o programa seguir por caminhos diferentes de processamento, conforme decisões tomadas durante a execução. Com isso, não segue mais um caminho sequencial, abrindo a possibilidade de execuções alternativas que podem variar conforme a situação e a necessidade, considerando a lógica desenvolvida.



Saiba mais

Em 2008, o Massachusetts Institute of Technology (MIT), o mesmo que desenvolveu a linguagem Logo, lançou o Scratch. Esse programa hoje é o padrão para o ensino da programação de computadores para crianças a partir dos oito anos. Com o Scratch, é possível desenvolver jogos, música, histórias, uma infinidade de programas que podem ser compartilhados na rede social dos usuários Scratch. É recomendável o conhecimento desse programa, que facilita muito a aprendizagem da programação de computadores.

O vídeo oficial da apresentação do Scratch pode ser visto no seguinte endereço:

<<http://scratch.mit.edu/about/>>.



Resumo

Na Unidade III vimos as tomadas de decisão, como um programa pode seguir por um caminho ou por outro. Aprendemos que esse encaminhamento é feito conforme o resultado lógico de uma expressão.

O primeiro comando visto foi o **se então**, que pode também vir acompanhado de um **senão**. Abordamos que, conforme o resultado da expressão lógica que acompanha o **se**, o programa segue um fluxo diferente. Caso o resultado seja verdadeiro, o programa executa o bloco do **então**; caso seja falso, se tiver o **senão**, executará o bloco do **senão**, e, se não tiver, pulará o bloco do **então** e prosseguirá o programa.

Vimos que as condicionais podem ser encadeadas de forma que criem estruturas complexas, as quais podem resultar em várias possibilidades de

execução. Vimos também um segundo modelo de condicional: aquele em que a decisão vai além das duas opções de Falso e Verdadeiro, podendo escolher o bloco a ser executado dentre muitos resultados. Esse tipo de funcional é o **escolha-caso**.

Por fim, estudamos a tradução das tomadas de decisão para a linguagem C, em que **se** (condição) é traduzido para *if* (condição); *senão*, para *else*; **escolha**, para *switch*; **caso**, para *case*; e **outrocaso**, para *default*. Lembrando que, na tradução, o *case* exige um *break* para encerrar o bloco em cada um dos casos.



Exercícios

Questão 1. Para se tomar uma decisão, uma série de condições deve ser avaliada para se decidir se uma ação deverá ou não ser realizada. As estruturas condicionais são utilizadas quando é preciso tomar decisões, ou quando são necessárias comparações. Com base nessas informações, escolha a alternativa que contém a afirmação considerada correta:

- A) Nestas estruturas condicionais uma operação lógica (<condição>) é avaliada; se o resultado desta avaliação for verdadeiro (V), então um determinado conjunto de instruções é executado. Caso contrário, ou seja, quando o resultado da avaliação for falso (F), um comando diferente é executado.
- B) O programa, ao deparar-se com a condição *se*, entende que encontrará uma expressão cujo resultado é um valor numérico. Caso esta expressão resulte em verdadeiro, será executado o bloco de comandos entre o *então* e o *senão*, passando a execução para o primeiro comando após o *fimse*. Caso a condição resulte em falso, será executado o bloco entre o *senão* e o *fimse*.
- C) Ao encontrar o *se*, o programa irá avaliar se a operação seguinte irá resultar em verdadeiro. Caso a operação seja diferente do valor lógico, resulta em falso. Sendo assim, o fluxo executará as instruções contidas no bloco do *senão*, e em seguida as contidas no bloco *se*.
- D) Quando for necessário escolher um resultado apenas, temos que encadear várias condicionais; no entanto, existe uma estrutura que evita o encadeamento de condicionais: a *escolha-caso*.
- E) A estrutura de decisão *escolha-caso* é utilizada principalmente quando se precisa testar várias expressões que produzem um resultado entre várias possibilidades, ou então testar o valor de uma variável, em que está armazenado um determinado conteúdo. Compara-se o resultado obtido opção a opção, testando-se com os valores fornecidos individuais de cada cláusula.

Resposta correta: alternativa A.

Análise das alternativas

A) Alternativa correta.

Justificativa: na estrutura condicional uma operação lógica (<condição>) é avaliada; se o resultado desta avaliação for verdadeiro (V), então um determinado conjunto de instruções é executado, e quando o resultado da avaliação for falso (F), um comando diferente é executado.

B) Alternativa incorreta.

Justificativa: o programa, ao deparar-se com a condição *se*, entende que encontrará uma expressão cujo resultado é lógico. Se a expressão resulta em verdadeiro será executado o bloco de comandos entre o *então* e o *senão*, passando a execução para o primeiro comando após o *fimse*. Caso a condição resulte em falso, será executado o bloco entre o *senão* e o *fimse*.

C) Alternativa incorreta.

Justificativa: ao encontrar o *se*, o programa irá avaliar se a operação seguinte irá resultar em verdadeiro. Caso a operação seja diferente do valor lógico, resulta em falso. Sendo assim, o fluxo executará as instruções contidas no bloco do *senão*.

D) Alternativa incorreta.

Justificativa: quando é necessário escolher um resultado entre vários possíveis, temos que encadear várias condicionais; no entanto, existe uma estrutura que evita o encadeamento de condicionais. Esta estrutura é a chamada *escolha-caso*.

E) Alternativa incorreta.

Justificativa: a estrutura de decisão *escolha-caso* é utilizada principalmente quando se precisa testar uma única expressão que produz um resultado entre várias possibilidades, ou então testar o valor de uma variável em que está armazenado um determinado conteúdo.

Questão 2. Existem várias estruturas que auxiliam a tomada de decisões em lógica de programação. Analise as afirmativas abaixo.

I – Na linguagem computacional temos três tipos de algoritmos mais utilizados: descrição narrativa, fluxograma e pseudocódigo ou português.

II – A descrição narrativa consiste em entender o problema proposto e escrever sua solução através de uma linguagem genérica, ou seja, a língua portuguesa.

III – No fluxograma, os passos para a resolução de um problema são feitos através de vários símbolos padrão.

IV – O foco do pseudocódigo é a lógica e a rigidez semântica.

É correto apenas o que se afirma em:

- A) I.
B) IV.
C) I e III.
D) II e III.
E) II e IV.

Resolução desta questão na plataforma.

This image shows a full page of blank white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page, typical of notebook paper or a document template. There are no margins, text, or other markings present.