2013

Portugol

Equivalências de estruturas entre Portugol e Pascal

Decode Team
INSTITUTO POLITÉCNICO DE TOMAR

Índice

Nota Geral:

Devido à especificação da linguagem, a tradução só é possível depois de ser executado o fluxograma.

Algumas notas sobre Pascal

- O pascal não é case sensitive.
- O pascal utiliza o ponto-e-vírgula para indicar o final da linha de código.
- O fim do programa é sinalizado com um "End." Sendo neste caso um ponto final e não um ponto-e-vírgula.
- As variáveis tem de ser declaradas antes do "Main". (Ou por outras palavras, antes do Begin em que elas iram ser utilizadas).
- As funções e Processos, têm de ser criados antes do código "Main".
- Qualquer tipo de texto tem se ser escrito dentro de pelicas (Exemplo -> 'Ola Mundo').
- Para fazer comentários em pascal é utilizado //
    - Exemplo: //comentário

Estrutura Início

Início:


```
Begin  //inicio
        <Código>
```

Nota: O programa vai ter de ser guardado com a extensão ".pas" para que possa ser executado.

Estrutura Fim

Fim:

End. //Fim

Variáveis

Equivalência entre TIPOS de variáveis

TIPO	Portugol	Java
Inteiro	Inteiro	Integer
Real	Real	Real
Texto	Texto	String
Caracter	Caracter	Char
Lógico	Logico	Boolean

Tabela 1 - Tipos de variáveis

Definição e atribuição de variáveis

variavel <- expressao

Se a variável não estiver definida em memória

Passo 1: Avaliar a expressão (VALOR).

Passo 2: Calcular Tipo do VALOR.

Passo 3: Declarar a variável: Var <variavel> : <Tipo>;

Se a variável estiver definida em memória
<variável> := <atribuição>;

Alguns exemplos de definição e atribuição de variáveis

Existem duas formas de definir variáveis e proceder à sua atribuição.

- Integer

  1 – Definir e atribuir variável

  Var variável : Integer;

  Variável := valor;

  Nota 1: Pode ser definido como *Integer* ou *integer.*

  Nota 2: *valor* é um número inteiro.

- Real

  1 – Definir e atribuir variável

  Var variável : Real;

  Variável := valor;

  Nota 1: Pode ser definido como *Real* ou real.

  Nota 2: *valor* é um número decimal. Ex: 5.3.

- String

  1 – Definir e atribuir variável

  Var variável : String;

  Variável := 'Valor';

  Nota 1: Pode ser definido como *String* ou string.
  Nota 2: têm de ser usadas pelicas.

- char

  1 – Definir e atribuir variável

  Var variável : Char;

  Variável := 'Valor';

  Nota 1: Pode ser definido como Char ou char.
  Nota 2: têm de ser usadas pelicas.

- Boolean

  1 – Definir e atribuir variável

  Var variável : *Boolean*;

  Variável := true;

  Nota 1: Pode ser definido como *Boolean* ou *boolean.*

  Nota 2: Este tipo de dados pode assumir o valor *true* ou *false.*

Estruturas input/output

Input – Ler

Tipo	Java
Real	Real
Texto	Line
Lógico	Boolean
INT	Integer
Char	Char

Tabela 2 - Tipo de variáveis para leitura

Se a variável não estiver definida em memória

Passo 1: Identificar o tipo (TIPO) de dados que foi lido.

Passo 2: Definir a variável:

Var <variavel> : <Tipo>;

Passo 3: Read(variavel) ; //Ler e continuar na mesma linha(ao escrever no ecrã)

Readln(variavel) ; // Ler e mudar de linha (ao escrever no ecrã)

Se a variável já estiver definida em memória

Passo 1: Realizar apenas o ***Passo 3*** do ponto anterior.

Output – Escrever

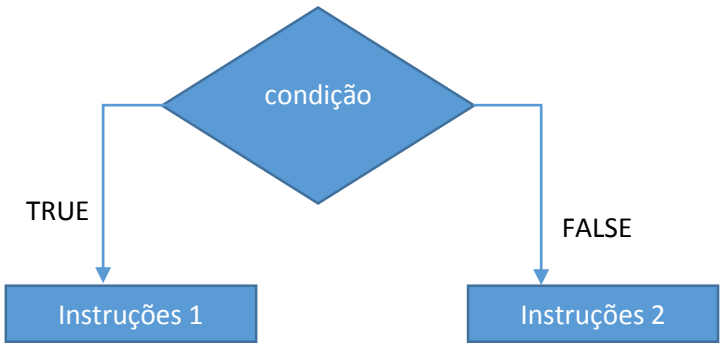
Para escrever no ecrã:

Write(expressão); //escrever e continuar na mesma linha(ao escrever no ecrã)

Writeln(expressão); //escrever e mudar de linha(ao escrever no ecrã)

Estruturas de Decisão

Condição *"if"* e *"if else"*



Para TRUE, escrever:

If (*condição*) then
Begin
Instruções 1

Para FALSE:

Se Instruções 2 for igual a (conector) não fazer nada.

Senão, escrever:

End
Else
Begin
Instruções 2

Exemplos práticos

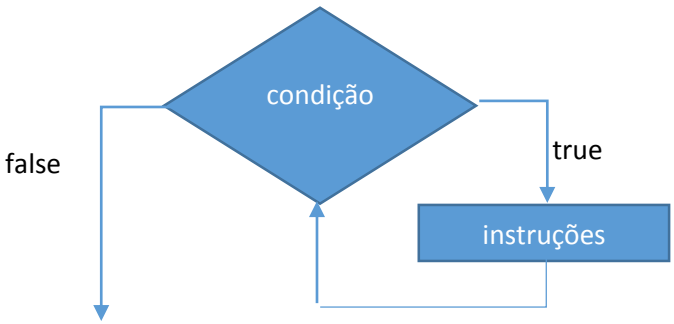
Condição "if"

```
If (n mod 2 = 0) then
Begin
        Writeln('Par');
End;
```

Condição "if else"

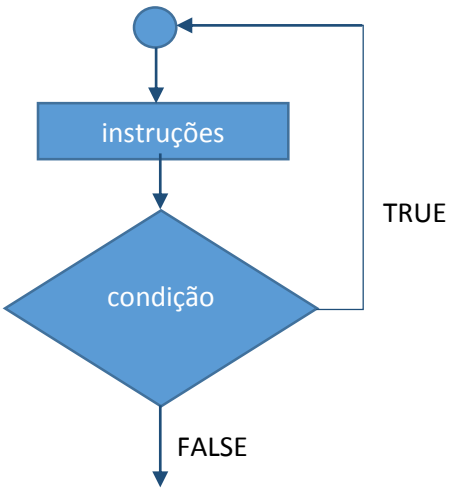
```
If (5 mod 2 = 0) then
Begin
        Writeln('Par');
End
else
Begin
        Writeln('Impar');
End;
```

Condição *"while"*




```
While (condição) do
Begin
        Instruções
End;
```

Condição *"do while"*



Instruções

```
End;
Until(condição);
```

Exemplos práticos

Condição "while"

```
While (i <= 10) do
Begin
        Writeln(i);
        I := i + 1;
 End;
```

Condição "do while"

```
repeat
Begin
        Read(i);
 End;
 Until(i < 0);
```

Estrutura Conector

Conector

Se for uma condição "*repeat*" escrever:

Repeat
Begin

Senão, escrever:

End;

Funções

Definir funções

Nota: Nas funções existe duas maneiras de obter retorno de um valor, uma delas é atribuir o valor a retornar ao nome da função, a outra maneira é criar uma variável local (tem de ser a primeira variável local declarada) e atribuir o valor a retornar.

```
Function NOME (a : TIPO; B : TIPO; …) : RETURN_TIPO;
Begin
```

Definir função *Exemplo* sem parâmetros de entrada

```
Function NOME : RETURN_TIPO;
Begin
```

Definir função *Exemplo* com parâmetros de entrada

```
Function NOME (PARAMETRO: TIPO) : RETURN_TIPO;
Begin
```

TIPO – Tipo de dados do parâmetro.
Consultar *tabela 1* no ponto *Equivalência entre TIPOS de variáveis*.

RETURN_TIPO – Tipo de dados de retorno da função.
Consultar *tabela 1* no ponto *Equivalência entre TIPOS de variáveis*.

NOME – Nome dado à função.

PARAMETRO – Variável utilizada pela função para auxiliar o cálculo.

Chamada de funções


```
NOME(PARAMETRO);
```

Exemplos do uso de funções

```
Function fact(k : Integer) : Integer;
Begin
        if(k > 2) then
        Begin
                fact := k * fact(k-1);
        End
        Else
        Begin
                fact := k;

        End;
End;
Var i, j : Integer;
 Begin
        i := 5;
        j := fact(i);
        writeln(j);
 End.
```

Estrutura de retorno

Return

Nota: Nas funções existe duas maneiras de obter retorno de um valor, uma delas é atribuir o valor a retornar ao nome da função, a outra maneira é criar uma variável local (tem de ser a primeira variável local declarada) e atribuir o valor a retornar.

Operadores

Aritméticos

Nome	Portugol	Java
Adição	a + b	a + b
Subtração	a – b	a - b
Divisão	a / b	a / b
Multiplicação	a * b	a * b
Resto da divisão inteira	a % b	a mod b
Potenciação	a ^ b	exp(ln(base)*expoente)

Tabela 3 - Equivalência de operadores aritméticos

Lógicos

Nome	Portugol	Java
Disjunção	a E b	a and b
Conjunção	a OU b	a or b
Conjunção Exclusiva	a XOU b	a xor b
Negação	NAO a	

Tabela 4 - Equivalência de operadores lógicos

Relacionais

Nome	Portugol	Java
Igual	a == b	a = b
Diferente	a != b	a <> b
Maior	a > b	a > b
Maior ou igual	a >= b	a >= b
Menor	a < b	a < b
Menor ou igual	a <= b	a <= b

Tabela 5 - Equivalência de operadores relacionais

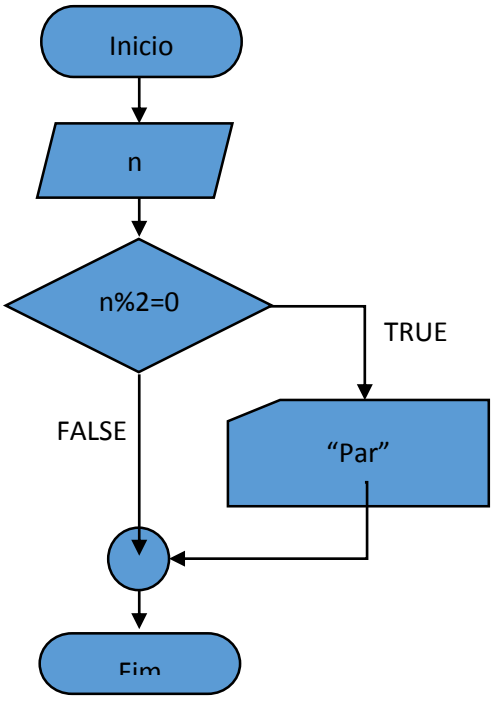
ANEXO

Para uma compreensão mais abrangente do uso das estruturas, ficam alguns exemplos mais extensivos, com o uso de várias estruturas em algoritmos completos.

Algoritmo com o uso da condição "if"

Problema: Verificar se um número introduzido pelo utilizador é par.

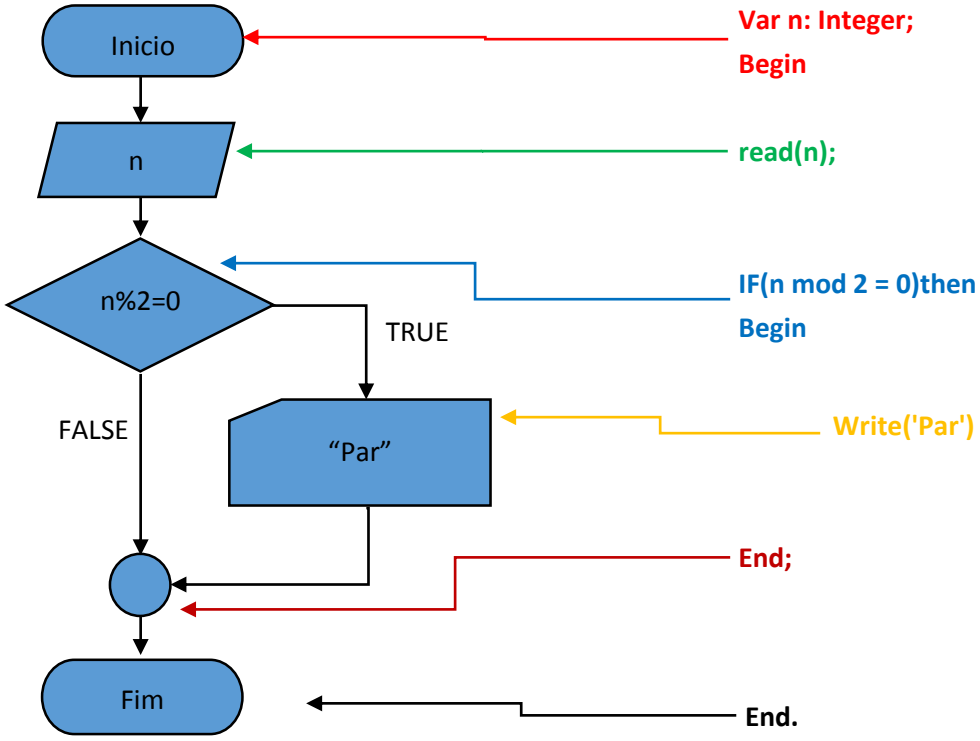
Fluxograma



Código

```
Var n: Integer;
 Begin
        read(n);
        IF(n mod 2 = 0)then
        Begin
                Write('Par')
        End;
 End.
```

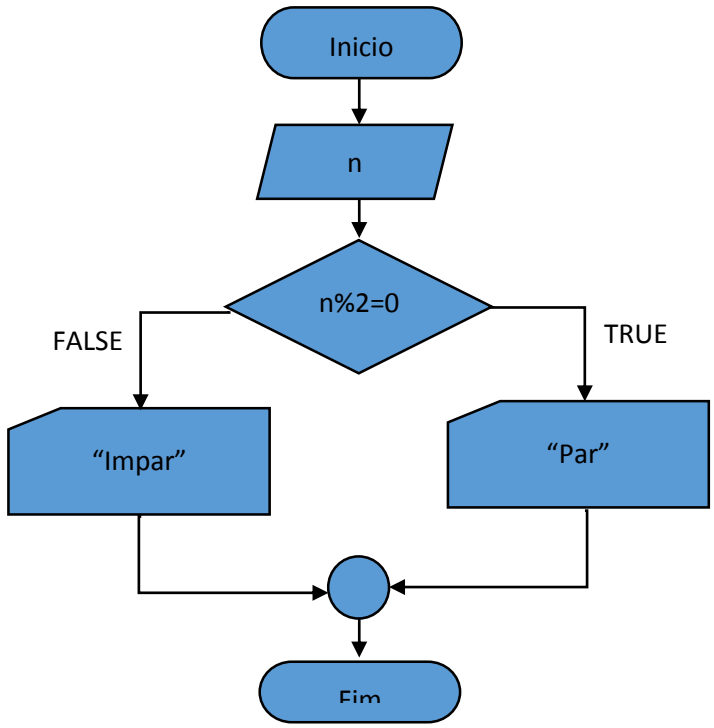
Esquema detalhado



Algoritmo com o uso da condição "if else"

Problema: Verificar se um número introduzido pelo utilizador é par ou ímpar.

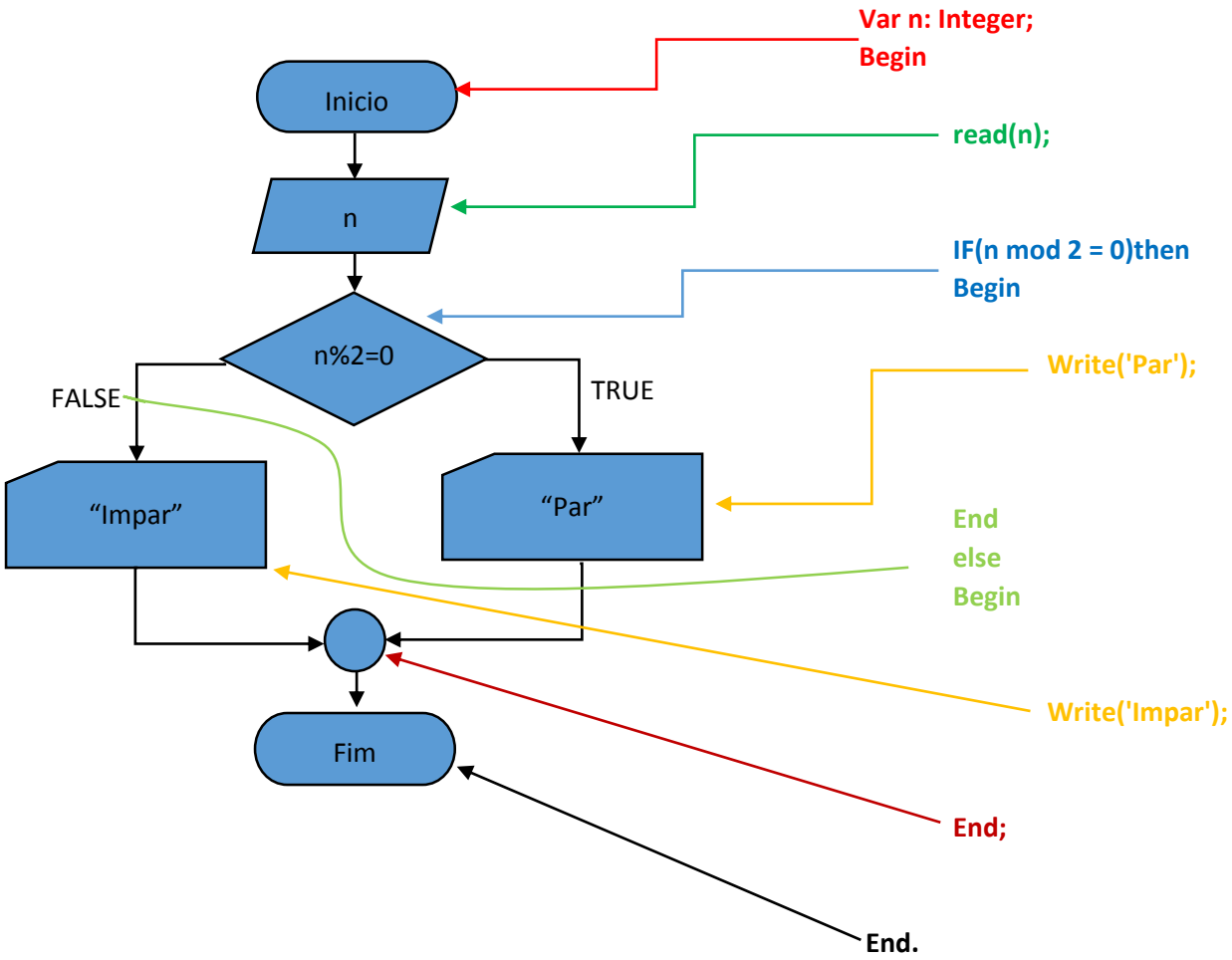
Fluxograma



Código:

```
Var n: Integer;
 Begin
        read(n);
        IF(n mod 2 = 0)then
        Begin
                Write('Par');
        End
        else
        Begin
                Write('Impar');
        End;
 End.
```

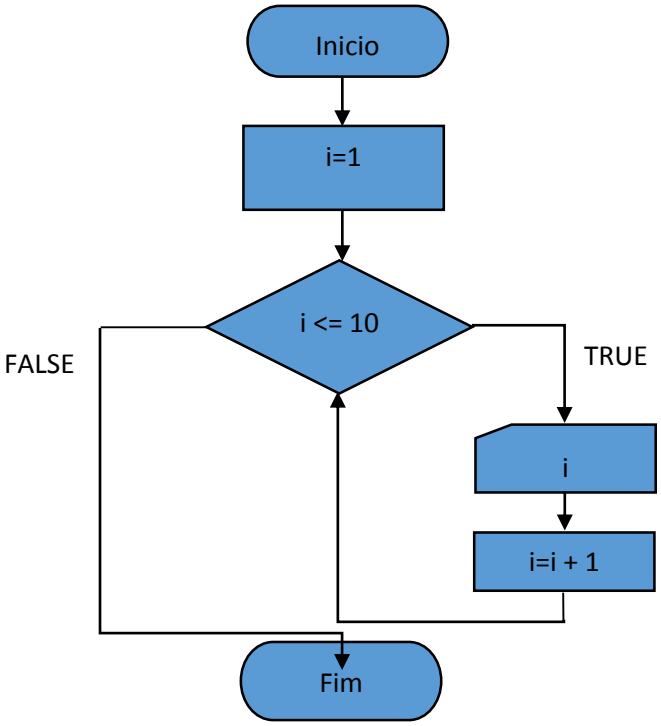
Esquema detalhado



Algoritmo com o uso da condição “while”

Problema: Escrever um número de 1 a 10.

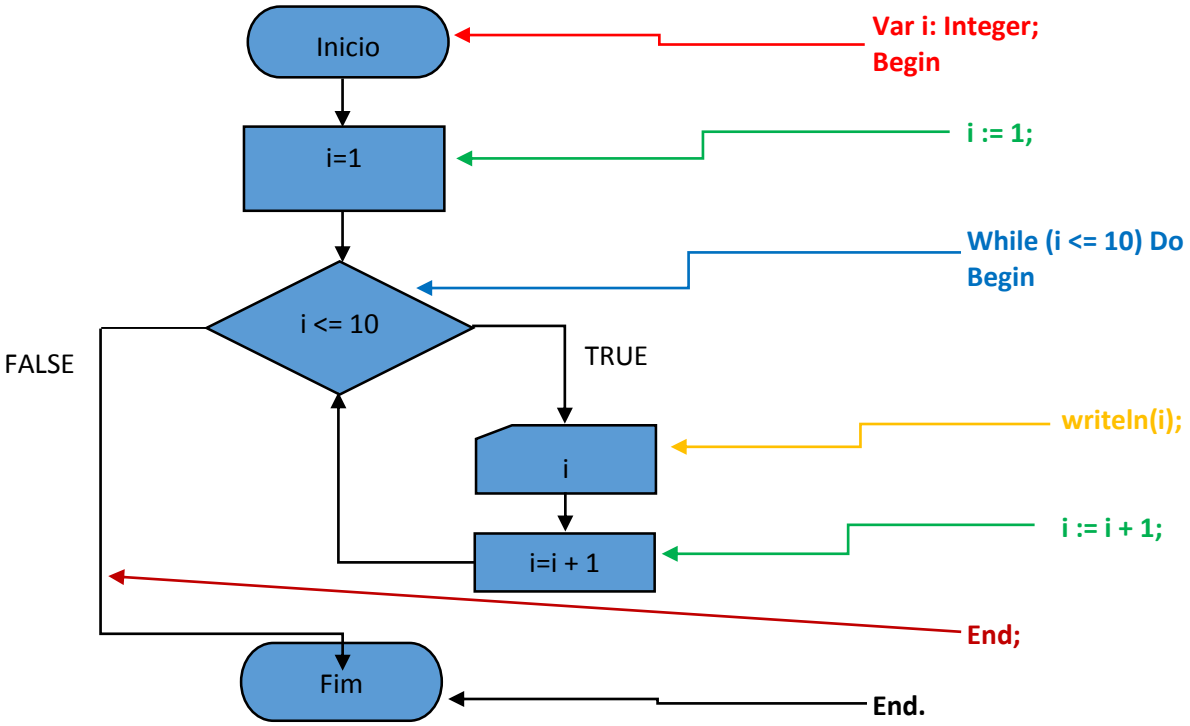
Fluxograma



Código

```
Var i: Integer;
 Begin
        i := 1;
        While (i <= 10) Do
        Begin
                writeln(i);
                i := i + 1;
        End;
 End.
```

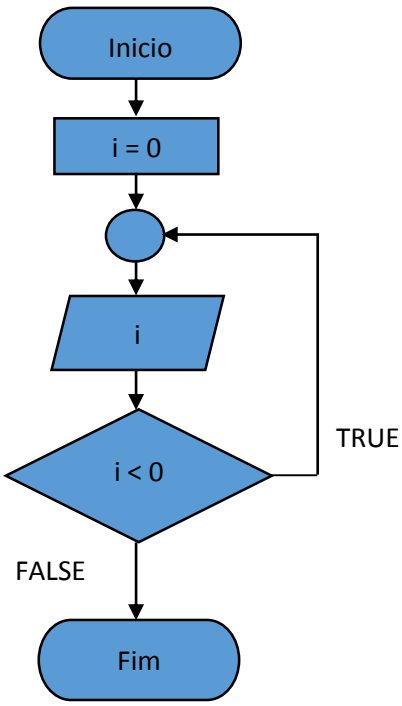
Esquema detalhado



Algoritmo com o uso da condição " do while"

Problema: Pedir um número positivo.

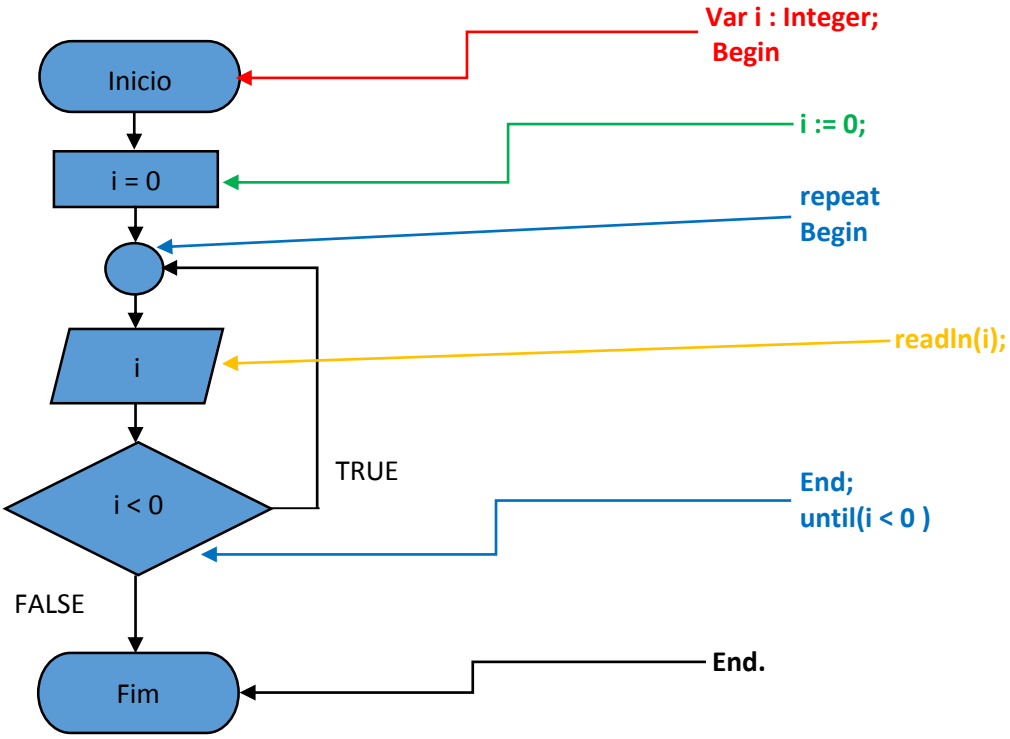
Fluxograma



Código

```
Var i : Integer;
 Begin
        i := 0;
        repeat
        Begin
                readln(i);
        End;
        until(i < 0 )
 End.
```

Esquema detalhado

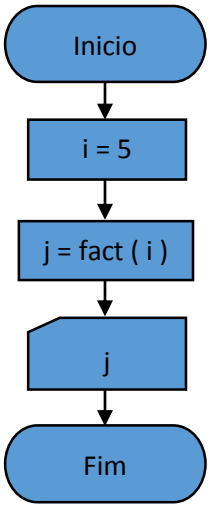


Algoritmo com o uso de uma função

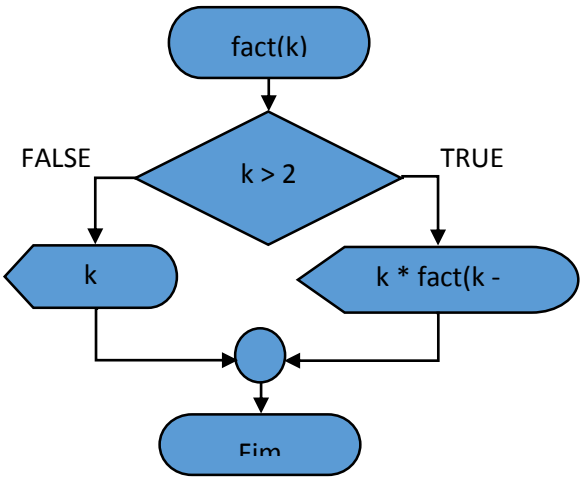
Problema: Factorial de um número.

Fluxogramas

Código principal



Função fact(k)



Código

```
Function fact(k : Integer) : Integer;
Begin
        if(k > 2) then
        Begin
                fact := k * fact(k-1);
        End
        Else
        Begin
                fact := k;

        End;
End;
Var i, j : Integer;
 Begin
        i := 5;
        j := fact(i);
        writeln(j);
 End.
```

Esquema detalhado