2013

Portugol

Equivalências de estruturas entre Portugol e Visual Basic

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 Visual Basic:

- Usa a mudança de linha para terminar uma linha de código.
- Normalmente faz uso de eventos para executar o código mas neste caso foi usado o modo de consola
- As funções podem ser definidas antes ou depois do Main.
- O código dever ser guardado num ficheiro com o mesmo nome do módulo (Module) e com extensão *.vb*.
- A linguagem permite fazer a inclusão de bibliotecas através da instrução Imports. As bibliotecas devem ser incluídas imediatamente antes da definição do módulo (ver *estrutura início*).
- A primeira função a ser codificada deve ser o início.

Estrutura Início

Início:


```
Module Programa

    Sub Main()
       Resto do programa


End Module
```

Nota: O nome do módulo (*Programa*), é um nome que identifica o algoritmo que está a ser resolvido. Todo o código dever ser guardado no ficheiro Programa.vb.

Estrutura Fim

Fim:


```
    Console.ReadLine()
End Sub
```

Nota: É usado "`Console.ReadLine()`" para pausar o programa antes de fechar a consola.

Nota 2: Para terminar uma função, usa-se:

```
End Function
```

Variáveis

Equivalência entre TIPOS de variáveis

TIPO	Portugol	Visual Basic
Inteiro	Inteiro	Long
Real	Real	Double
Texto	Texto	String
Caracter	Caracter	Char
Lógico	Logico	Boolean

Tabela 1 - Tipos de variáveis

Definição e atribuição de variáveis

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: Dim variável As TIPO = expressao

Se a variável estiver definida em memória
variavel = expressao

Alguns exemplos de definição e atribuição de variáveis

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

- Long

  1 – Definir e atribuir variável no mesmo passo:

```
Dim i As Long = valor
```

  2 – Definir e atribuir variável em passos separados:

```
Dim variavel As Long
variavel = valor
```

  Nota 1: Pode ser definido como *Long* ou *long.*

  Nota 2: *valor* é um número inteiro.

- Double

  1 – Definir e atribuir variável no mesmo passo:

```
Dim variavel As Double = valor
```

  2 – Definir e atribuir variável em passos separados:

```
Dim variavel As Double
variavel = valor
```

  Nota 1: Pode ser definido como *Double* ou *double*

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

- String

  1 – Definir e atribuir variável no mesmo passo:

```
Dim variavel As String = "valor"
```

  2 – Definir e atribuir variável em passos separados:

```
Dim variavel As String
variavel = "valor"
```

  Nota 1: Pode ser definido como *String* ou *string*

  Nota 2: têm de ser usadas aspas.

- char

1 – Definir e atribuir variável no mesmo passo:

```
Dim variavel As Char = "X"
```

2 – Definir e atribuir variável em passos separados:

```
Dim variavel As Char
variavel = "X"
```

Nota 1: Pode ser definido como *Char* ou *char*

Nota 2: *X* é um caracter e deve estar dentro de aspas.

- Boolean

1 – Definir e atribuir variável no mesmo passo:

```
Dim variavel As Boolean = False
```

2 – Definir e atribuir variável em passos separados:

```
Dim variavel As Boolean
variavel = False
```

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	Visual Basic
Real	Double
Texto	String
Lógico	Boolean
INT	Long
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:

Dim variável As TIPO

Passo 3: `variavel = Console.ReadLine()`

Nota: É usado ".ReadLine()" para deixar uma linha depois da leitura do valor. Caso isto não seja necessário, usa-se ".Read()".

Se a variável já estiver definida em memória

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

Output – Escrever

Para escrever no ecrã:

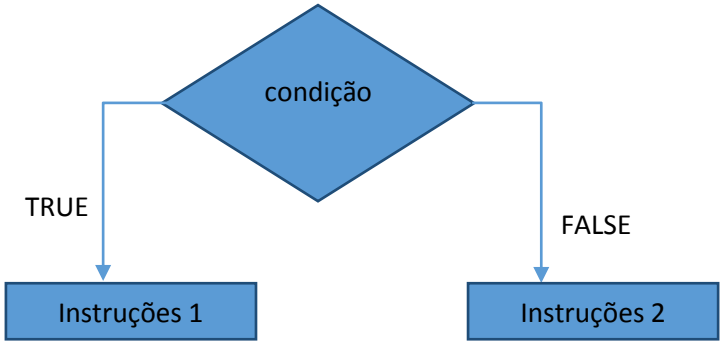
```
Console.WriteLine(expressao)
```

Nota 1: Caso seja necessário devolver texto também, é necessário converter as variáveis para string (.ToString)

Ex: `Console.WriteLine(expressao.ToString + "")`

Estruturas de Decisão

Condição *"if"* e *"if else"*



Para TRUE, escrever:

```
If (condição) Then
        Instruções 1
```

Para FALSE:

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

Senão, escrever:

```
Else
    Instruções 2
```

Exemplos práticos

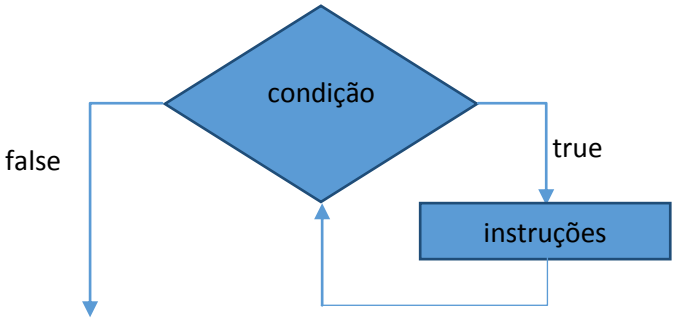
Condição "if"

```
If (n Mod 2 = 0) Then
    Console.Write("Par")
End If
```

Condição "if else"

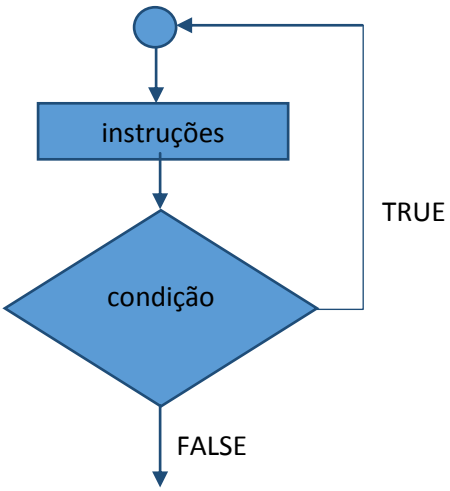
```
If (n Mod 2 = 0) Then
    Console.Write("Par")
Else
    Console.Write("Impar")
End If
```

Condição *"while"*




```
While (condição)
        Instruções
End While
```

Condição *"do while"*



Instruções

```
Loop While (condição)
```

Exemplos práticos

Condição "while"

```
While (i <= 10)
    Console.WriteLine(i)
    i = i + 1
End While
```

Condição "do while"

```
Do
    i = Console.ReadLine()
Loop While (i < 0)
```

Estrutura Conector

Conector

Se for uma condição "*do while*" escrever:

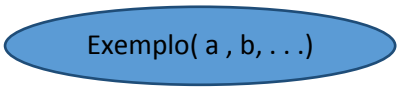
```
Do
```

Senão, escrever:

```
End If
```

Funções

Definir funções



Nota: Depois da função ser executada pelo menos uma vez (ver *Algumas notas sobre Java*), o tipo de retorno das função RETURN_TIPO e o TIPOx dos parametros pode ser identificado:

```
Function exemplo(a, b, …)
```

Definir função *Exemplo* sem parâmetros de entrada

```
Function NOME()
```

Definir função *Exemplo* com parâmetros de entrada

```
Function NOME(PARAMETRO)
```

NOME – Nome dado à função.

PARAMETRO – Variável utilizada pela função para auxiliar o cálculo.

Chamada de funções

NOME(PARAMETRO);

Exemplos do uso de funções

```
Module Funcao

    Sub Main()
        Dim i As Long
        i = Console.ReadLine()
        Dim j As Double
        j = factorial(i)
        Console.WriteLine(i)
        Console.ReadLine()
    End Sub

    Function factorial(k)
        If k > 2 Then
            Return k * factorial(k - 1)
        Else
            Return k
        End If
    End Function

End Module
```

Estrutura de retorno

Return


```
Return expressao
```

Operadores

Aritméticos

Nome	Portugol	Visual Basic
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	a ^ b
Concatenação de texto	a, b	a + b

Tabela 3 - Equivalência de operadores aritméticos

Lógicos

Nome	Portugol	Visual Basic
Disjunção	a E b	a And b
Conjunção	a OU b	a Or b
Conjunção Exclusiva	a XOU b	a Xor b
Negação	NAO b	Not b

Tabela 4 - Equivalência de operadores lógicos

Relacionais

Nome	Portugol	Visual Basic
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

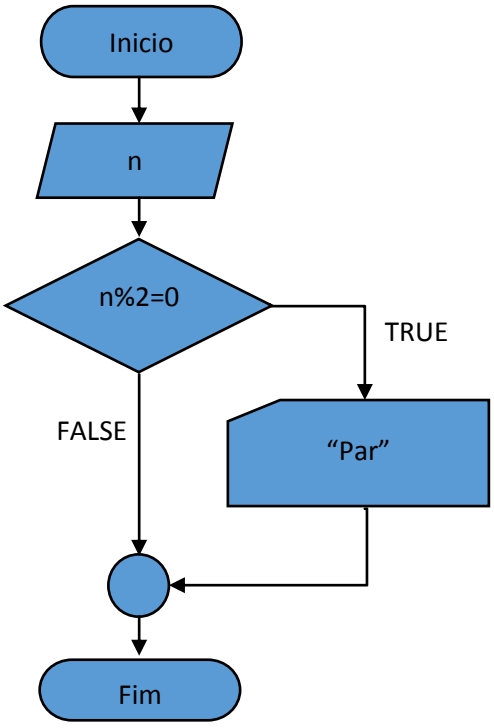
ANEXO

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.

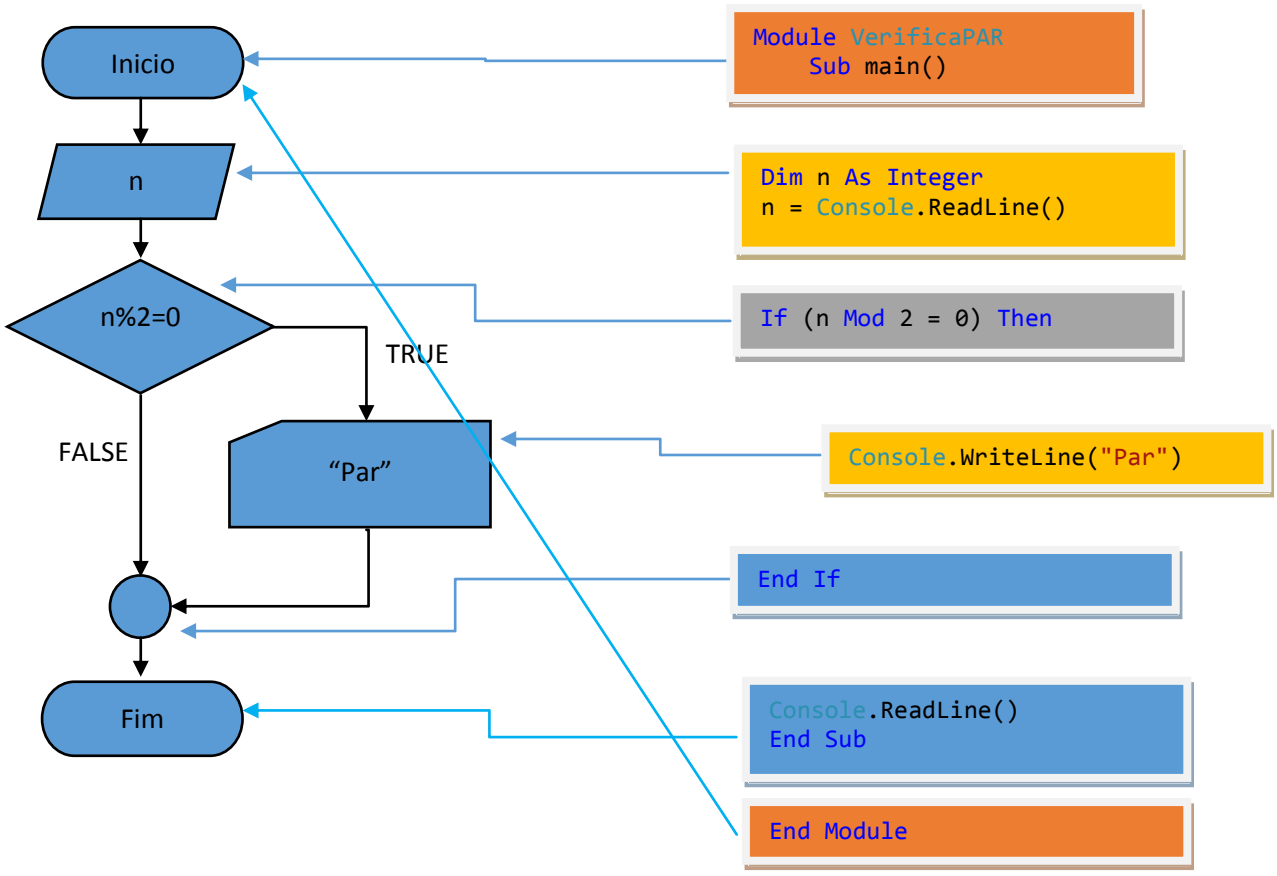
Fluxograma



Código

```
Module VerificaPAR
    Sub main()
        Dim n As Integer
        n = Console.ReadLine()
        If (n Mod 2 = 0) Then
            Console.WriteLine("Par")
        End If
        Console.ReadLine()
    End Sub
End Module
```

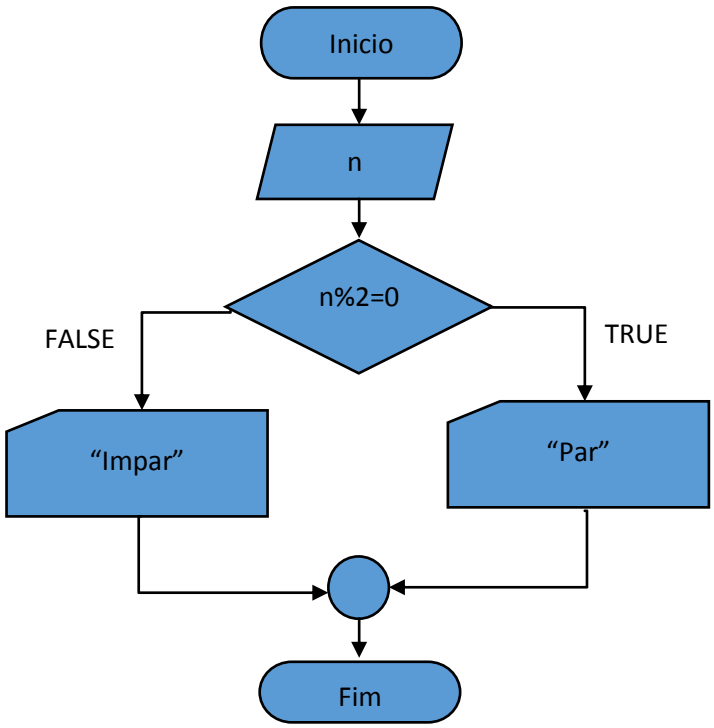
Esquema detalhado



Algoritmo com o uso da condição “if else”

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

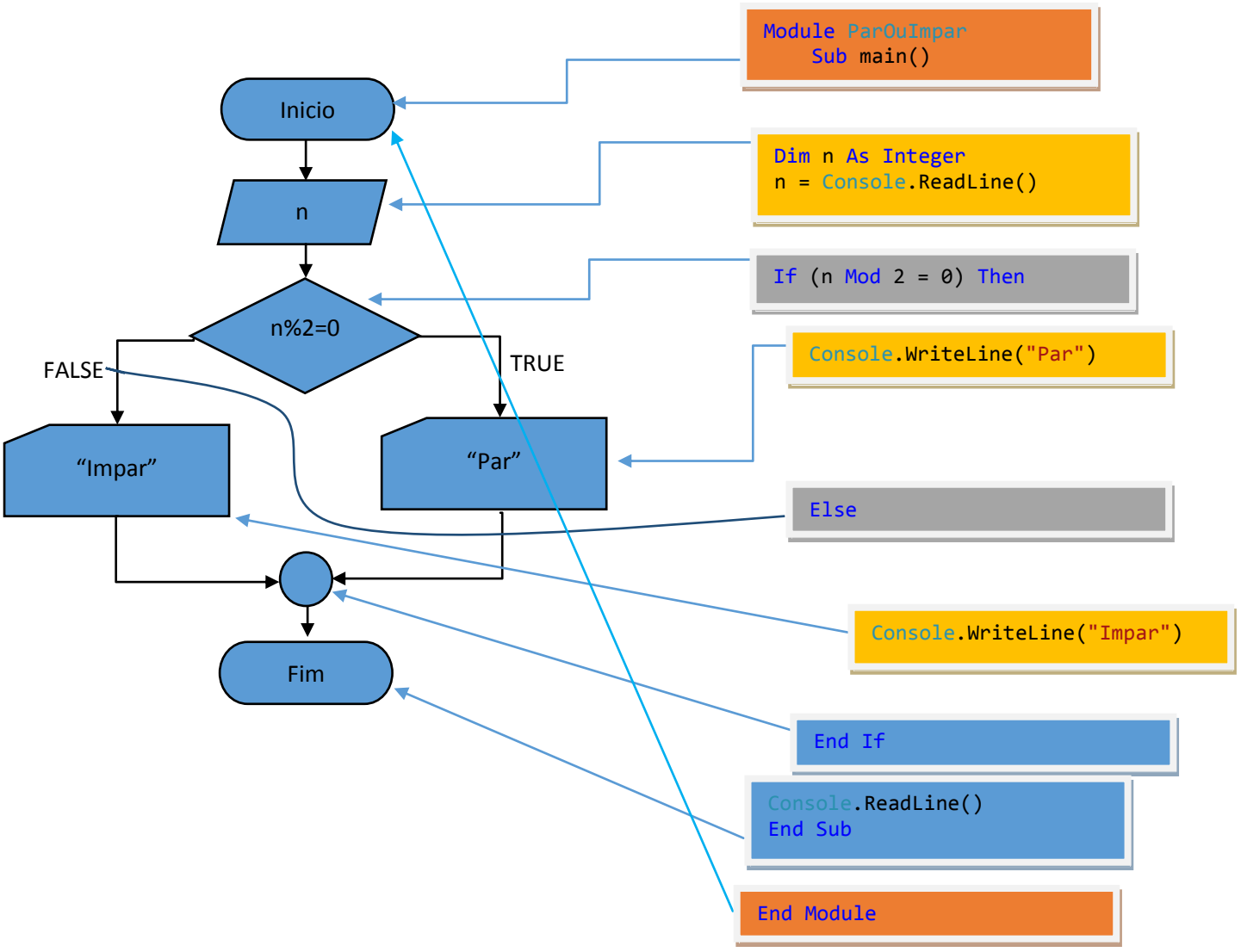
Fluxograma



Código:

```
Module ParOuImpar
    Sub main()
        Dim n As Integer
        n = Console.ReadLine()
        If (n Mod 2 = 0) Then
            Console.WriteLine("Par")
        Else
            Console.WriteLine("Impar")
        End If
        Console.ReadLine()
    End Sub
End Module
```

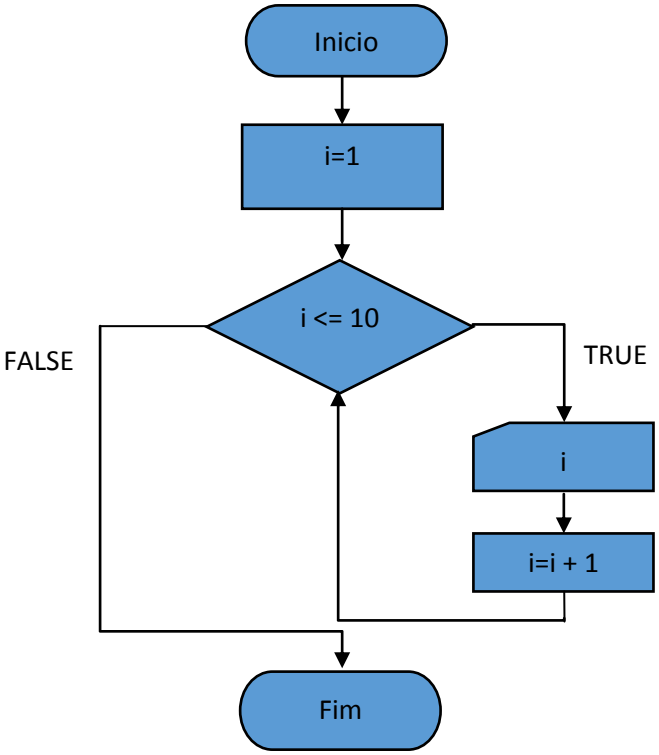
Esquema detalhado



Algoritmo com o uso da condição “while”

Problema: Escrever um número de 1 a 10.

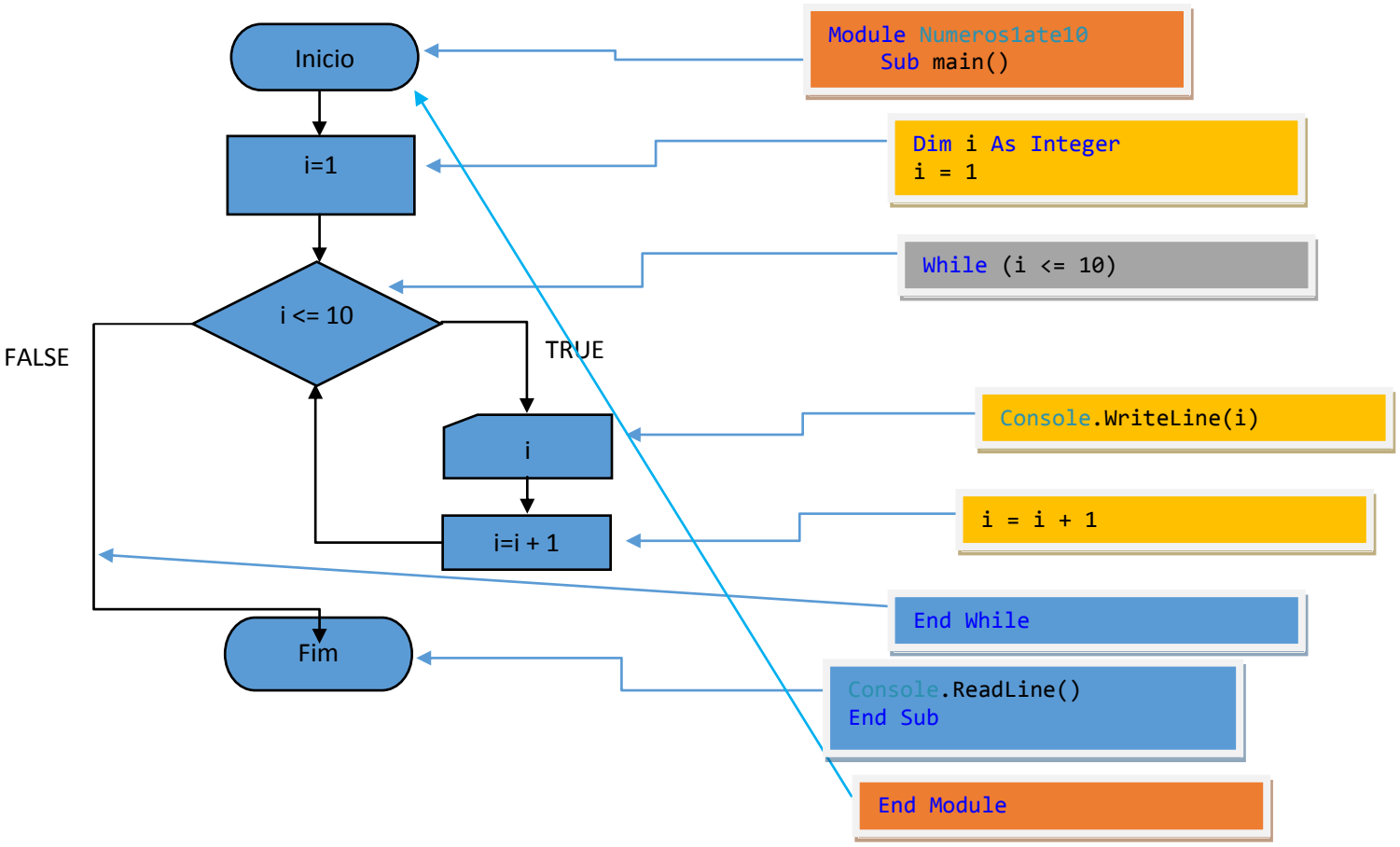
Fluxograma



Código

```
Module Numeros1ate10
    Sub main()
        Dim i As Integer
        i = 1
        While (i <= 10)
            Console.WriteLine(i)
            i = i + 1
        End While
        Console.ReadLine()
    End Sub
End Module
```

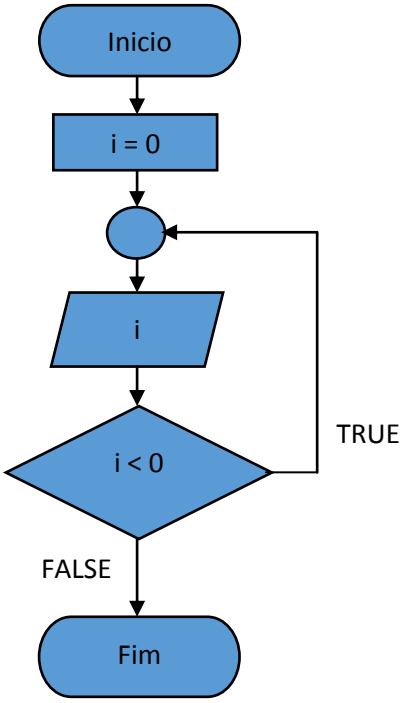
Esquema detalhado



Algoritmo com o uso da condição " do while"

Problema: Pedir um número positivo.

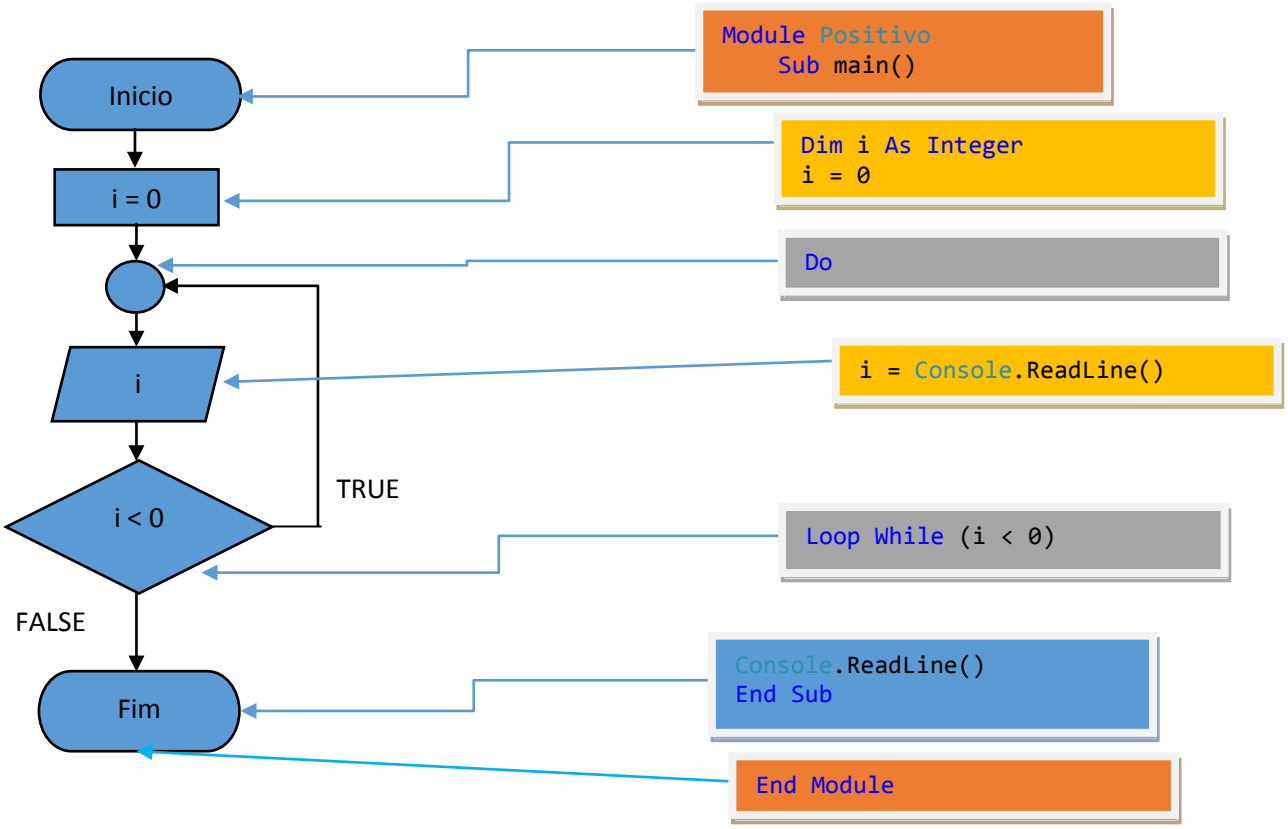
Fluxograma



Código

```
Module Positivo
    Sub main()
        Dim i As Integer
        i = 0
        Do
            i = Console.ReadLine()
        Loop While (i < 0)
        Console.ReadLine()
    End Sub
End Module
```

Esquema detalhado

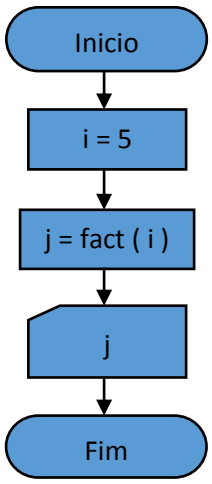


Algoritmo com o uso de uma função

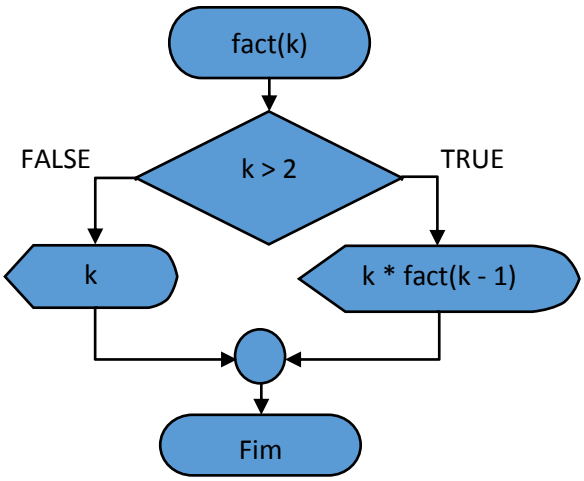
Problema: Factorial de um número.

Fluxogramas

Código principal



Função fact(k)



Código

```
Module Funcao

    Sub Main()
        Dim i As Integer
        i = 5
        Dim j As Integer
        j = factorial(i)
        Console.WriteLine(j)
        Console.ReadLine()
    End Sub

    Function factorial(k)
        If k > 2 Then
            Return k * factorial(k - 1)
        Else
            Return k
        End If
    End Function

End Module
```

Esquema detalhado