2013

Portugol

Equivalências de estruturas entre Portugol e C#

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 C#:

- É case sensitive.
- Usa o ponto e virgula (;) para terminar uma linha de código.
- As funções podem ser definidas antes ou depois no main.
- A primeira função a ser codificada deve ser o início.
- As classes em C# estão sempre dentro de namespaces;
- O código fica guardado em ficheiros do tipo .cs;

Estrutura Início

Início:


```
using System;

namespace nome{

        class nome{

                public static void Main(string[] args){

                Resto do programa

}}
```

Nota: O nome da classe (*nome*), é um nome que identifica o algoritmo que está a ser resolvido, e não necessita de estar associado ao ficheiro .cs. O nome do namespace também não necessita de estar associado ao projecto.

Estrutura Fim

Fim:

}

Variáveis

Equivalência entre TIPOS de variáveis

TIPO	Portugol	C#
Inteiro	Inteiro	int
Real	Real	double
Texto	Texto	String
Caracter	Caracter	char
Lógico	Logico	bool

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: TIPO variavel = 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.

- **Int**

  1 – Definir e atribuir variável no mesmo passo:

  int variavel =valor;

  2 – Definir e atribuir variável em passos separados:

  int variavel;

  variável = valor;

  Nota 1: *valor* é um número inteiro.

  Nota 2: Podem ser introduzidas várias variáveis através do uso da vírgula (int a,b;).

- **Double**

  1 – Definir e atribuir variável no mesmo passo:

  double variavel =valor;

  2 – Definir e atribuir variável em passos separados:

  double variavel;

  variavel=valor;

  Nota 1: *valor* é um número decimal. Ex: 5.3.

- **String**

  1 – Definir e atribuir variável no mesmo passo:

  string variavel="valor";

  2 – Definir e atribuir variável em passos separados:

  string variavel;
  variavel=”valor”;

  Nota 1: tem que ser definido com letra minúscula.
  Nota 2: têm de ser usadas aspas.

- **Char**

  1 – Definir e atribuir variável no mesmo passo:

  char variavel =’X’;

2 – Definir e atribuir variável em passos separados:

char variavel;

variavel='X';

Nota 1: tem que ser definido com letra minúscula

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

- **Bool**

1 – Definir e atribuir variável no mesmo passo:

bool variavel =false;

2 – Definir e atribuir variável em passos separados:

bool variavel;

variavel=false;

Nota 1: Este tipo de dados pode assumir o valor *true* ou *false.*

Estruturas input/output

Input – Ler

Tipo	C#
Real	double
Texto	string
Lógico	bool
INT	int
Char	char

Tabela 2 - Tipo de variáveis para leitura

Se a variável já estiver definida em memória

Passo 1: Em C# a leitura de inputs necessita de ser transformada no tipo da variável que vai receber esse input, isso e feito através do método Parse(), que tem como parâmetro o comando de leitura:

```
variavel =tipo.Parse(Console.ReadLine());
```

Nota 1: tipo é substituido pelo tipo da variavel ou seja por int, double, char, string etc.

Se a variável não estiver definida em memória

Passo 1: Definir a variável:

```
TIPO variavel;
```

Passo 2: Especificar que o input é desse tipo e associar o input à variável:

```
variavel= tipo.Parse(Console.ReadLine());
```

Output – Escrever

Para escrever no ecrã:

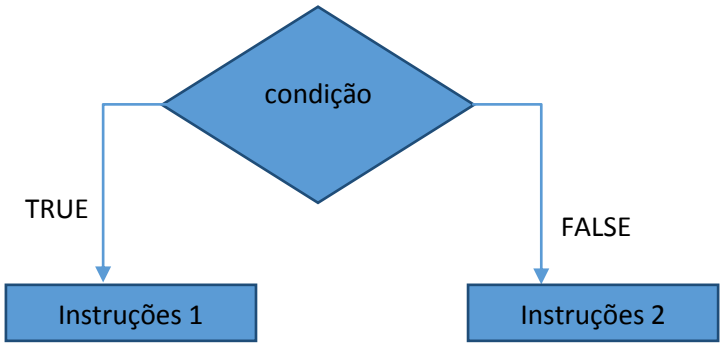
Console.Write(expressão + “”);

Para escrever no ecrã e passar de linha:

Console.WriteLine(expressão + “”);

Estruturas de Decisão

Condição *“if”* e *“if else”*



Para TRUE, escrever:

If (*condição*) {

Instruções 1

Para FALSE:

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

Senão, escrever:

} else {

Instruções 2

Exemplos práticos

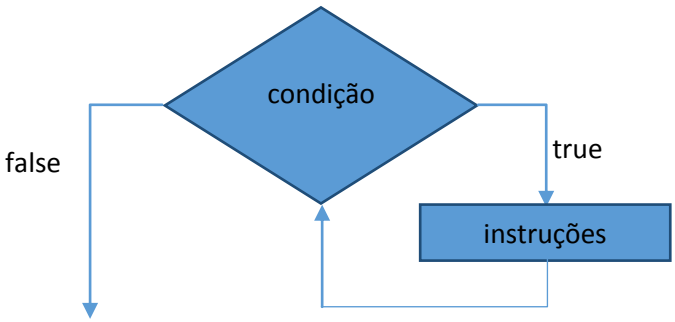
Condição "if"

```
if (n%2==0) {
        Console.WriteLine("Par");
}
```

Condição "if else"

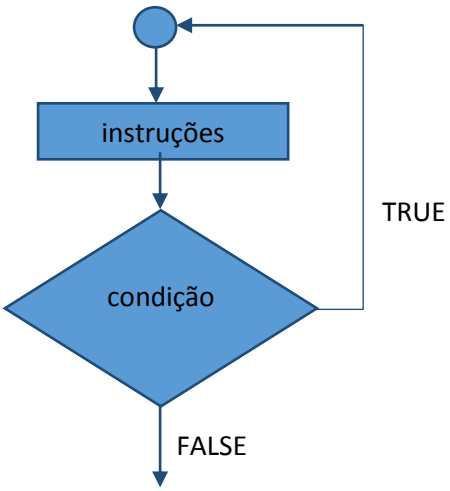
```
if (n % 2 == 0) {
        Console.WriteLine ("Par");
} else {
        Console.WriteLine ("Impar");
}
```

Condição *"while"*




```
while (condição){
        Instruções
}
```

Condição *"do while"*



Instruções

```
do{

}while(condição);
```

Exemplos práticos

Condição "while"

```
while(i<=10){
        Console.Write(i);
        i++;
}
```

Condição "do while"

```
do{
        i=int.Parse(Console.Read());
}while(i<0);
```

Estrutura Conector

Conector

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

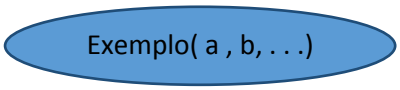
do {

Senão, escrever:

}

Funções

Definir funções



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

```
public static RETURN_TIPO exemplo( TIPO1 a , TIPO2 b, . . .. )
{
```

Definir função *Exemplo* sem parâmetros de entrada

```
public static TIPO NOME ()  {
```

Definir função *Exemplo* com parâmetros de entrada

```
public static TIPO NOME (PARAMETRO)  {
```

TIPO – Executa a função e calcula o tipo de retorno.
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

```
public class Funcao {

  public static void main(String[] args) {
   int i=5;
   int j;
   j=fact(i);
   Console.WriteLine(j);
  }

  public static int fact(int k){
    if (k>2) {
      return k*fact(k-1);
    }else{
      return k;
    }
  }
}
```

Estrutura de retorno

Return


```
return expressao;
```

Operadores

Aritméticos

Nome	Portugol	C#
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
Potenciação		Math.pow(base,expoente);
Concatenação de texto		+

Tabela 3 - Equivalência de operadores aritméticos

Lógicos

Nome	Portugol	C#
Disjunção	a && b	a && b
Conjunção	a || b	a || b
Conjunção Exclusiva	a ^ b	a ^ b
Negação		a !b

Tabela 4 - Equivalência de operadores lógicos

Relacionais

Nome	Portugol	C#
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

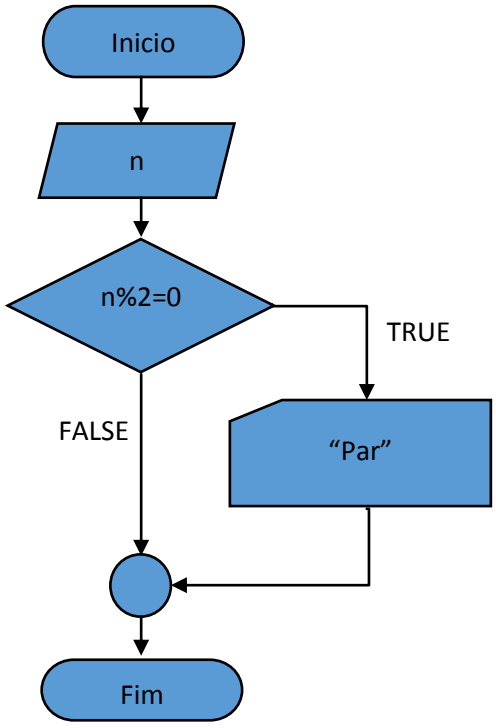
ANEXO

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.

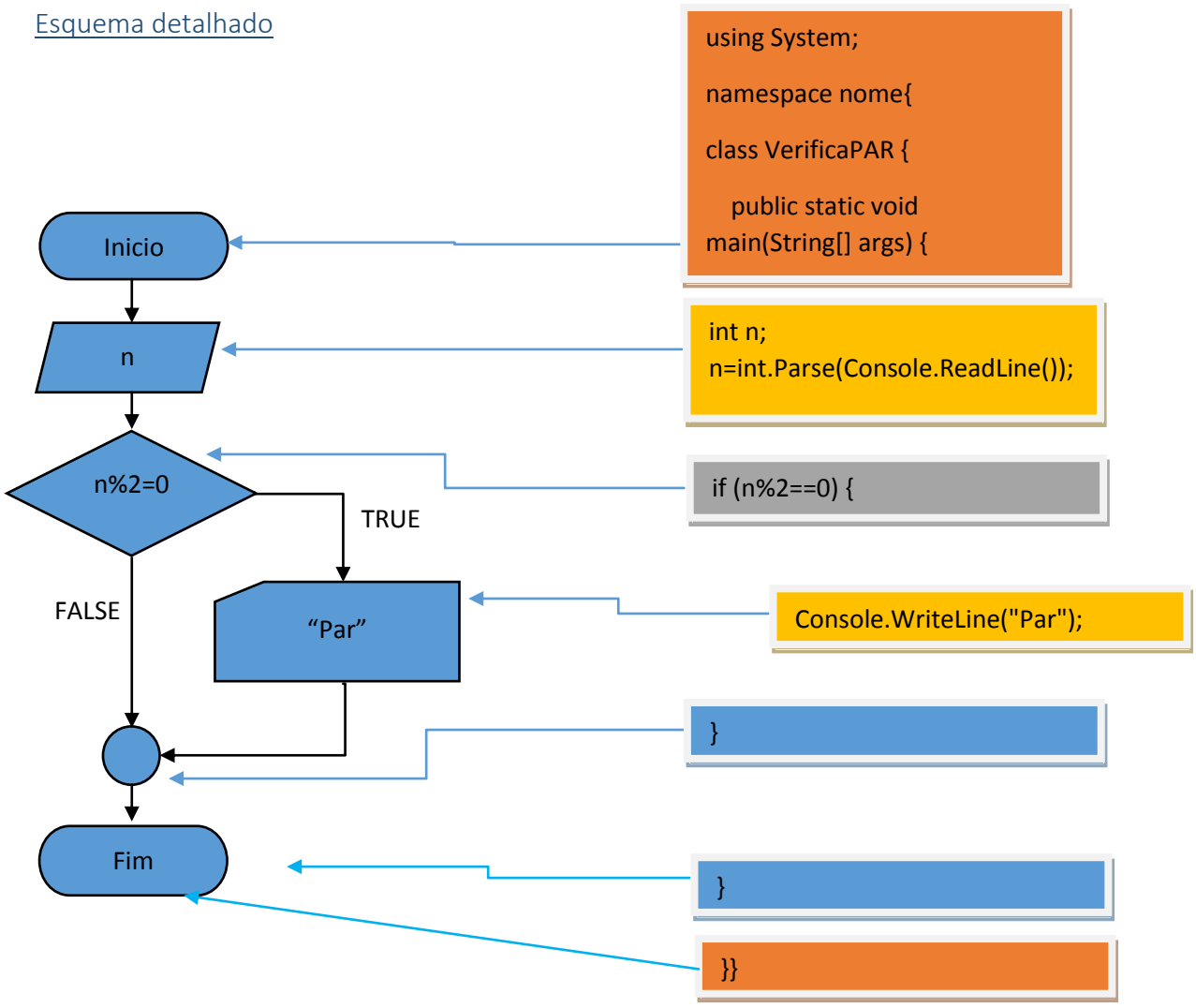
Fluxograma



Código

```
using System;

namespace nome{

class VerificaPAR {

   public static void main(String[] args) {
      int n;
      n=int.Parse(Console.ReadLine());
      if (n%2==0) {
         Console.WriteLine("Par");
      }
   }}}
```

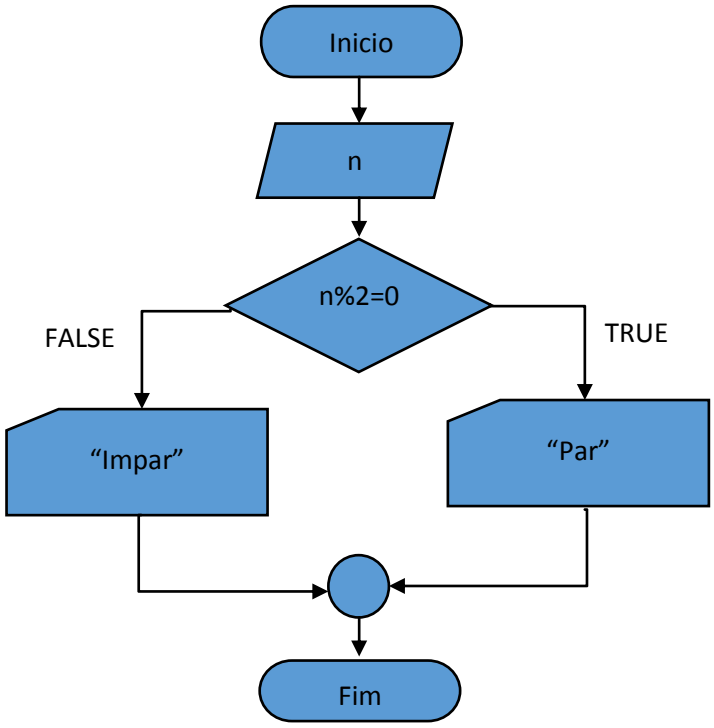
Esquema detalhado



Algoritmo com o uso da condição “if else”

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

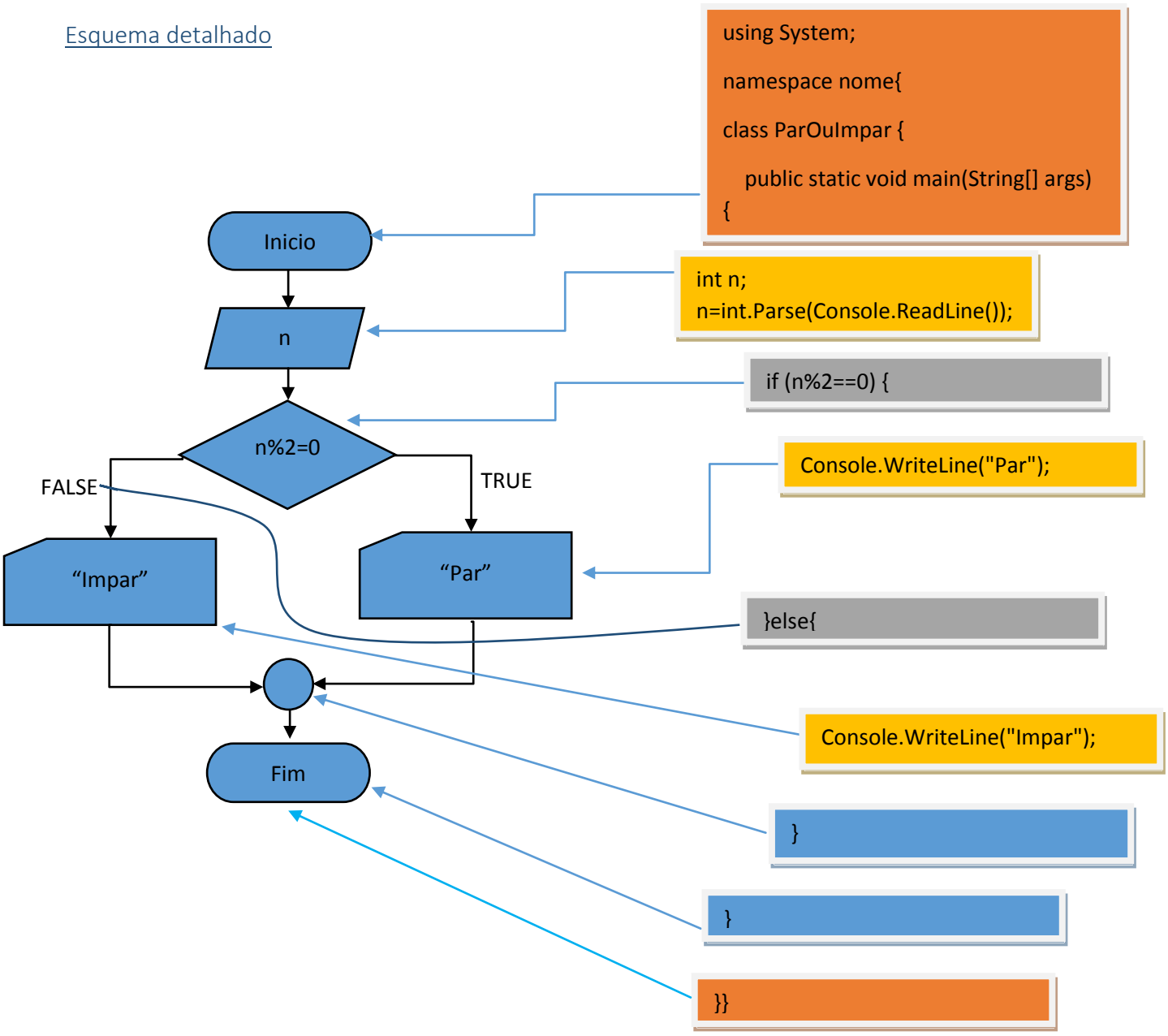
Fluxograma



Código:

```
using System;

namespace nome{


class ParOuImpar {

   public static void main(String[] args) {

      int n;
      n = int.Parse(Console.ReadLine());
      if (n % 2 == 0) {
         Console.WriteLine("Par");
      } else {
         Console.WriteLine("Impar");
      }

   }

}}
```

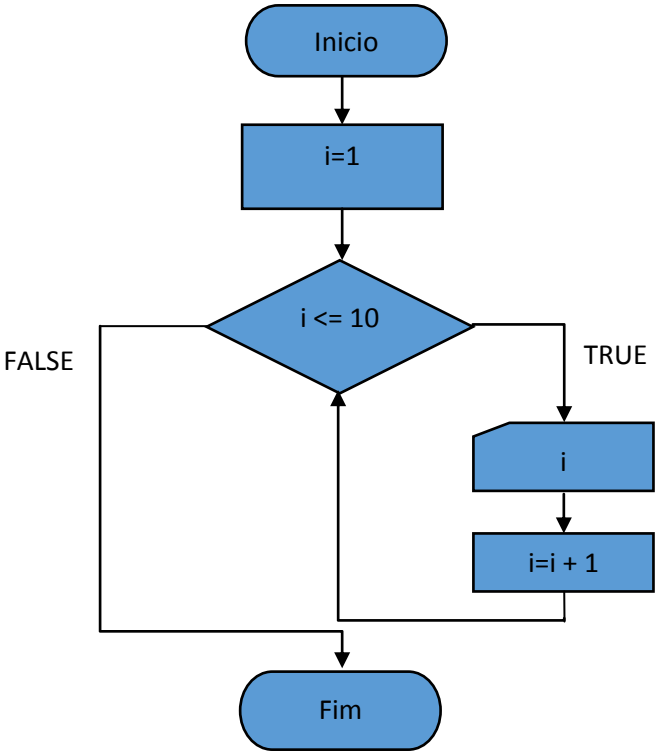
Esquema detalhado



Algoritmo com o uso da condição “while”

Problema: Escrever um número de 1 a 10.

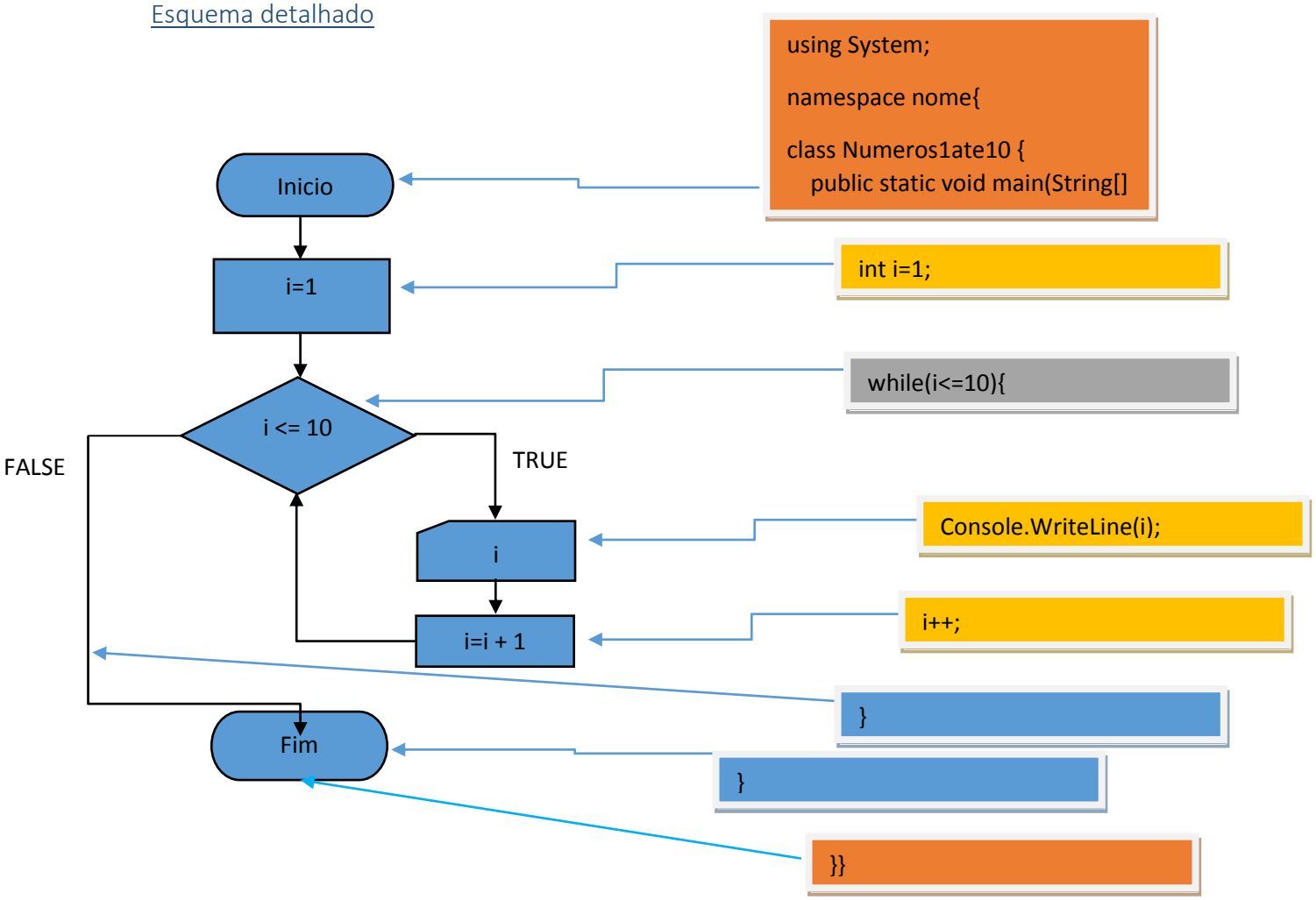
Fluxograma



Código

```
using System;

namespace nome{

class Numeros1ate10 {

   public static void main(String[] args) {

      int i=1;
      while(i<=10){
         Console.WriteLine(i);
         i++;
      }

   }

}}
```

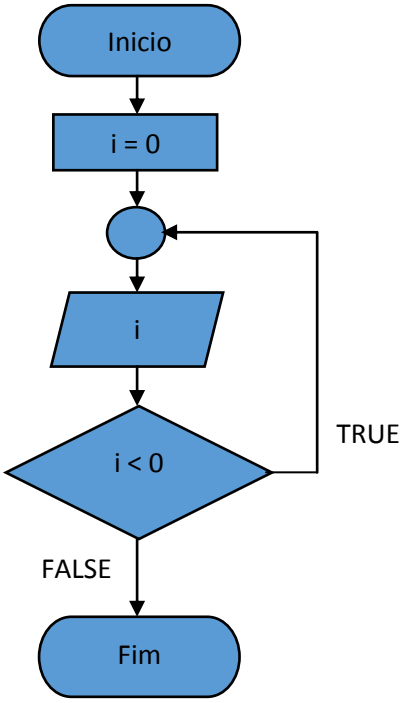
Esquema detalhado



Algoritmo com o uso da condição " do while"

Problema: Pedir um número positivo.

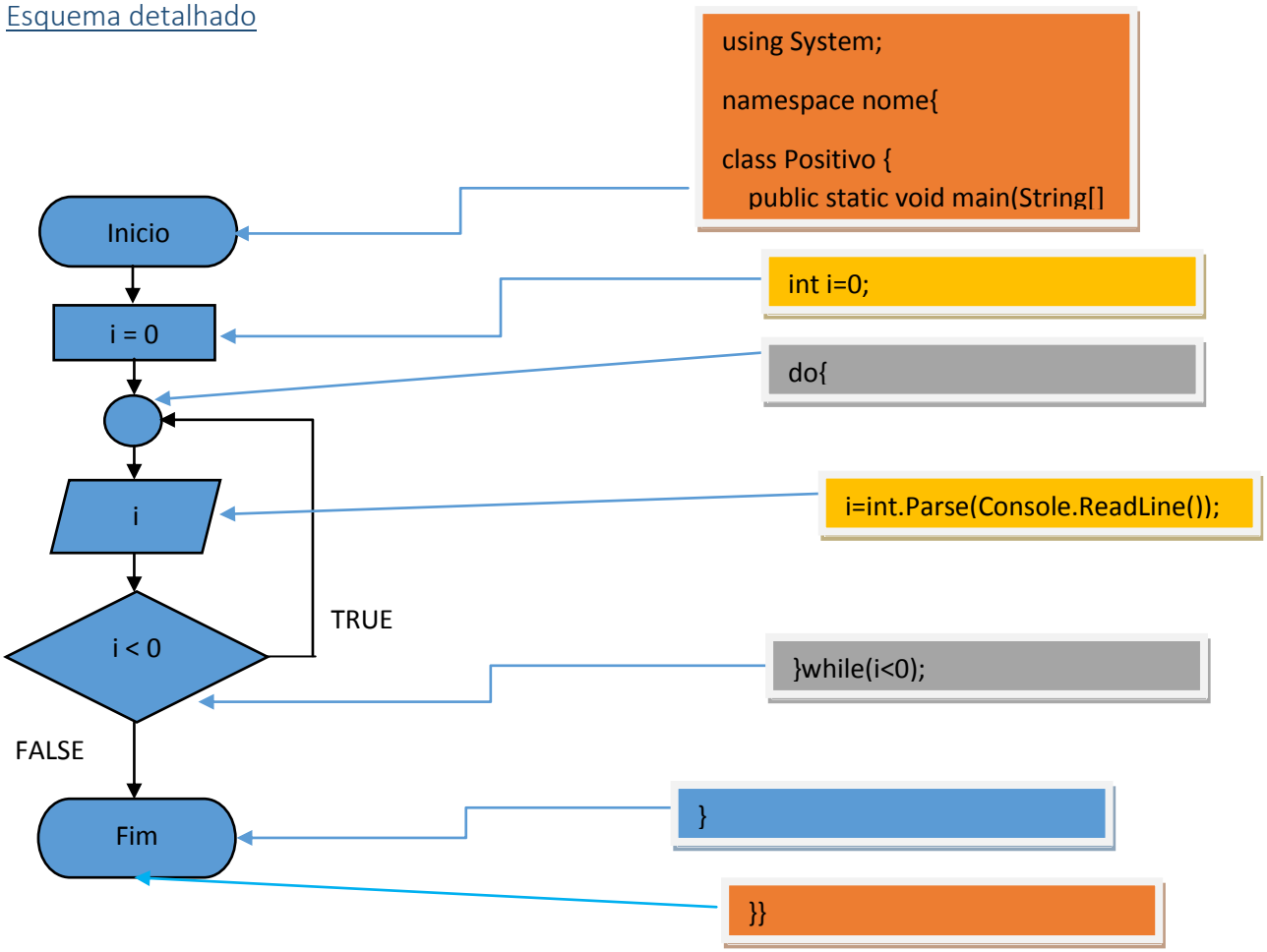
Fluxograma



Código

```
using System;

namespace nome{

class Positivo {

   public static void main(String[] args) {

     int i=0;
     do{
        i=int.Parse(Console.ReadLine());
     }while(i<0);

   }

}}
```

Esquema detalhado

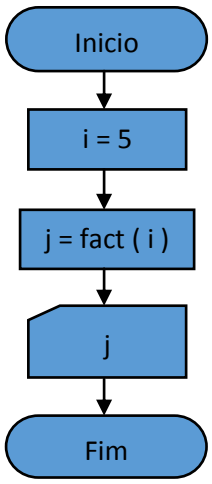


Algoritmo com o uso de uma função

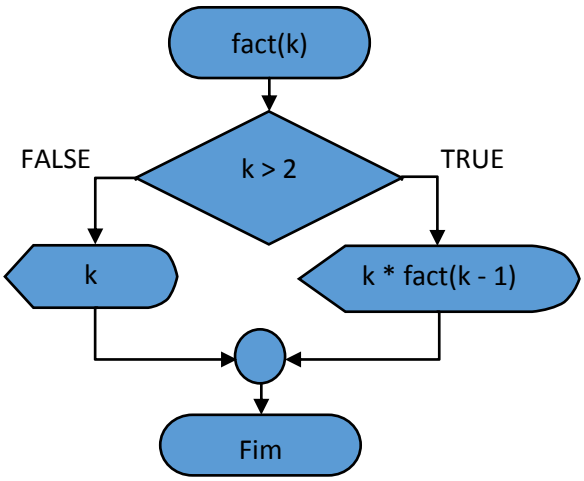
Problema: Factorial de um número.

Fluxogramas

Código principal



Função fact(k)



Código

```
using System;

namespace nome{

class Funcao {

   public static void main(String[] args) {

     int i=5;
     int j;
     j=fact(i);
     Console.WriteLine(j);

   }

   public static int fact(int k){

      if (k>2) {
         return k*fact(k-1);
      }else{
         return k;
      }
   }
}
}
```

Esquema detalhado