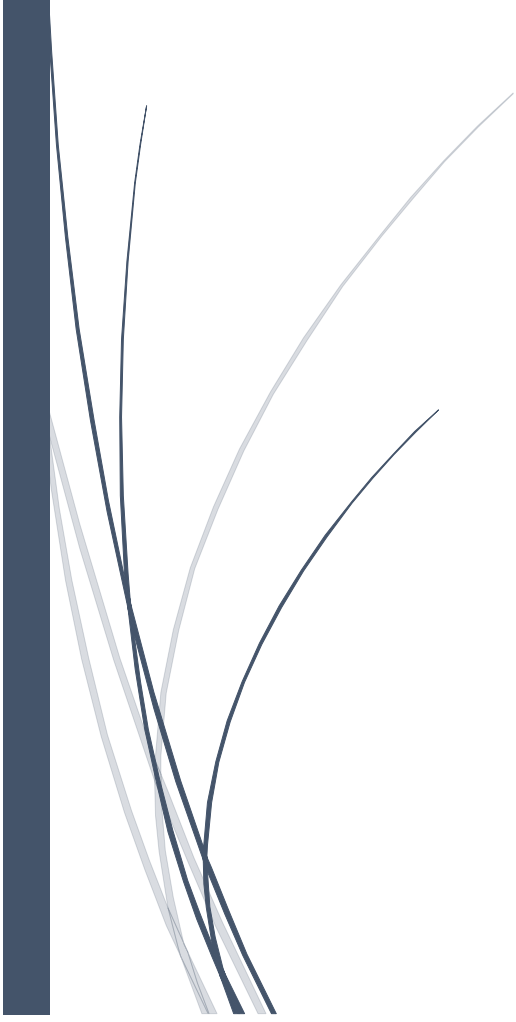
2013

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 devem ser definidas antes do main.
- A primeira função a ser codificada deve ser o início.
- O código fica guardado em ficheiros do tipo .cpp;

Estrutura Início

Início:


```
using namespace std;

int main()

{
```

.

Estrutura Fim

Fim:


```
}
```

Variáveis

Equivalência entre TIPOS de variáveis

TIPO	Portugol	C++
Inteiro	Inteiro	Long
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.

- Long

  1 – Definir e atribuir variável no mesmo passo:

  long variavel =valor;

  exemplo: long xpto=1;

2 – Definir e atribuir variável em passos separados:

long variavel;

Variável=valor;

Nota 1: deve ser definido com letra minúscula.

Nota 2: *valor* é um número inteiro.

- 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: Deve ser definido *double* (letra minúscula).

Nota 2: *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 plicas.

- Boolean

  1 – Definir e atribuir variável no mesmo passo:

  bool variavel =false;

  2 – Definir e atribuir variável em passos separados:

  bool variavel;

  variavel=false;

  Nota 1deve ser definido por bool (letra minuscula).

  Nota 2: 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	Long
Char	Char

Tabela 2 - Tipo de variáveis para leitura

Se for a primeira vez a ler do teclado

É necessário é fazer o import da biblioteca iostream, antes de qualquer função:

```
#include <iostream>
```

Se a variável não estiver definida em memória

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

Passo 2: Definir a variável:

```
TIPO variavel;
```

Passo 3: cin >> variavel;

Se a variável já estiver definida em memória

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

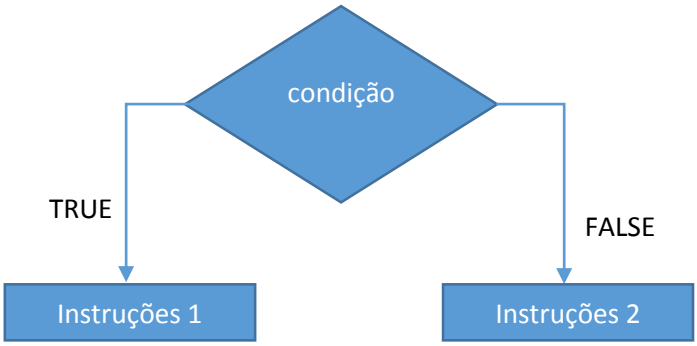
Output – Escrever

Para escrever no ecrã:

cout << expressao

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

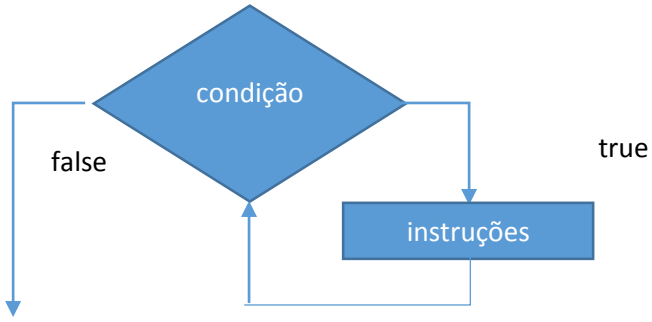
Condição "if"

```
if (n%2==0) {
        cout << "Par";
}
```

Condição "if else"

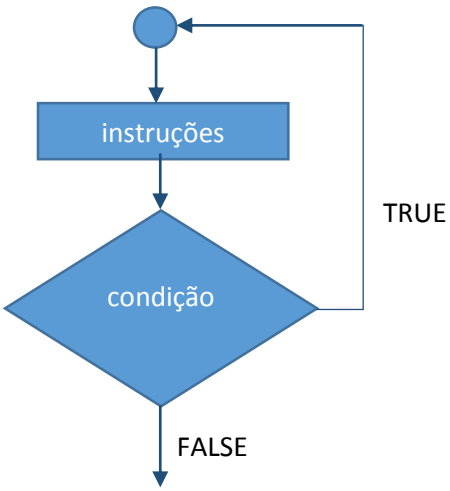
```
if (n % 2 == 0) {
        cout << "Par";
} else {
        cout << "Impar";
}
```

Condição *"while"*




```
While (condição){
        Instruções
}
```

Condição *"do while"*



Instruções

}while(*condição*);

Exemplos práticos

Condição "while"

```
while(i<=10){
        System.out.print(i);
        i++;
}
```

Condição "do while"

```
do{
        i=scanner.nextInt();
}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 TIPOx dos parametros pode ser identificado:

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

Definir função *Exemplo* sem parâmetros de entrada

```
TIPO NOME ()  {
```

Definir função *Exemplo* com parâmetros de entrada

```
TIPO NOME (TIPO_PARAMETRO 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


```
variavel = NOME(PARAMETRO);
```

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	a % b
Potenciação	a ^ b	pow(base,expoente);
Concatenação de texto	,	<<

Tabela 3 - Equivalência de operadores aritméticos

nota: para usar o pow em C++, deve se incluir a biblioteca math.h antes de qualquer função

Lógicos

Nome	Portugol	C++
Disjunção	a E b	a && b
Conjunção	a OU b	a || b
Conjunção Exclusiva	a XOU b	a ^ b
Negação	NAO a	! a

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

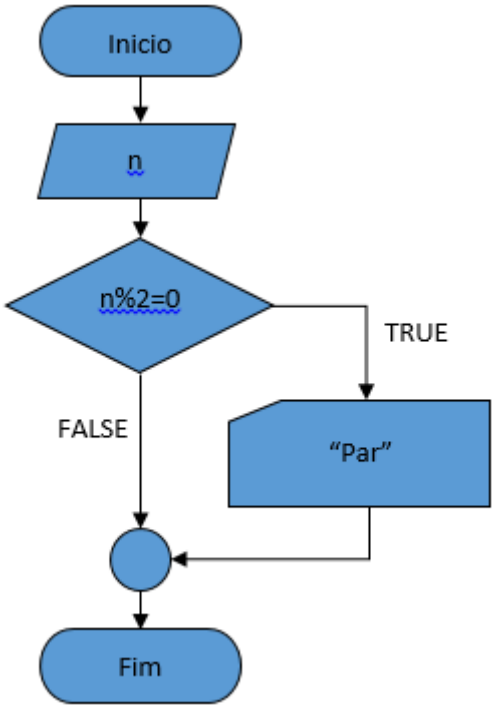
ANEXO

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.

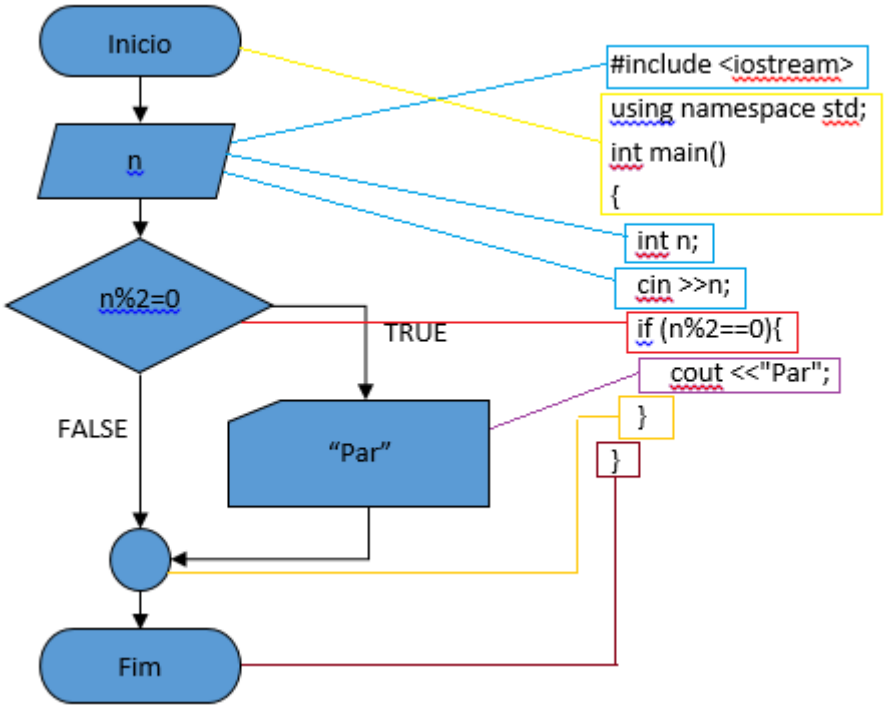
Fluxograma



Código

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   cin >>n;
   if (n%2==0){
       cout <<"Par";
   }
}
```

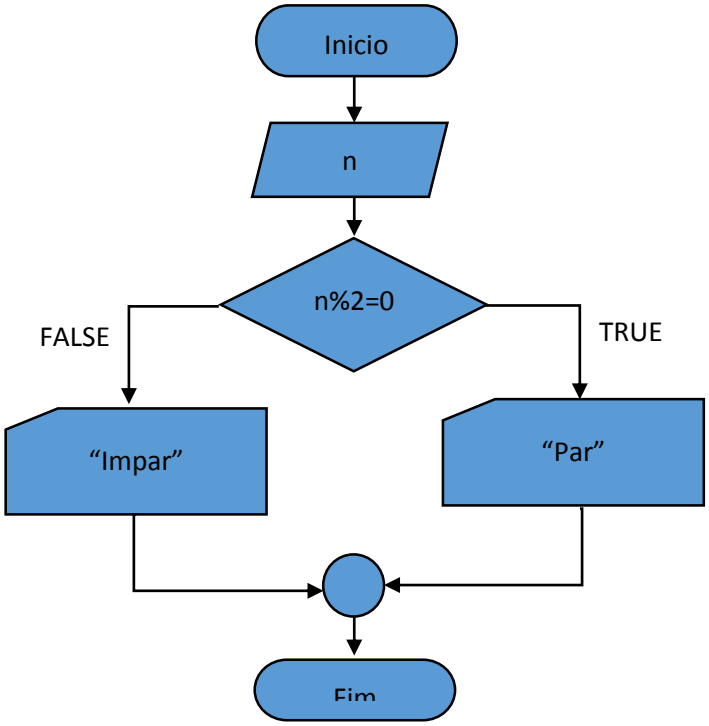
Esquema detalhado



Algoritmo com o uso da condição “if else”

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

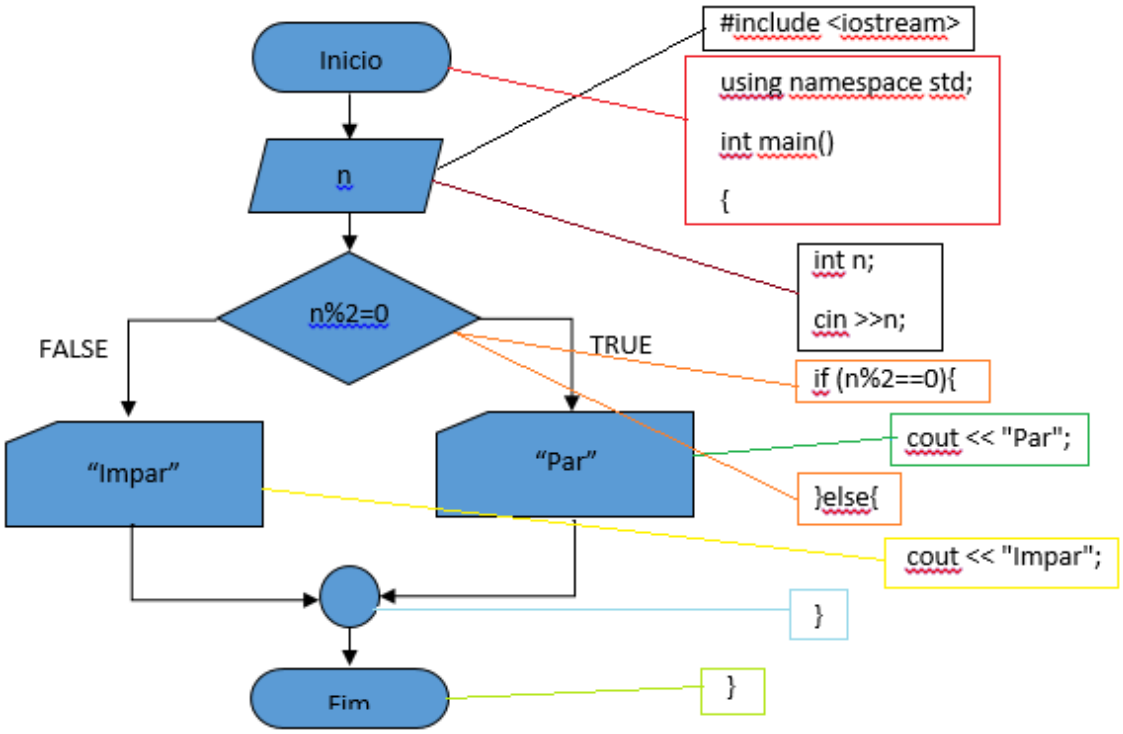
Fluxograma



Código:

```
#include <iostream>

using namespace std;

int main()

{

	int n;

	cin >>n;

	if (n%2==0){

		cout << "Par";

	}else{

		cout << "Impar";

	}

	return 0;

}
```

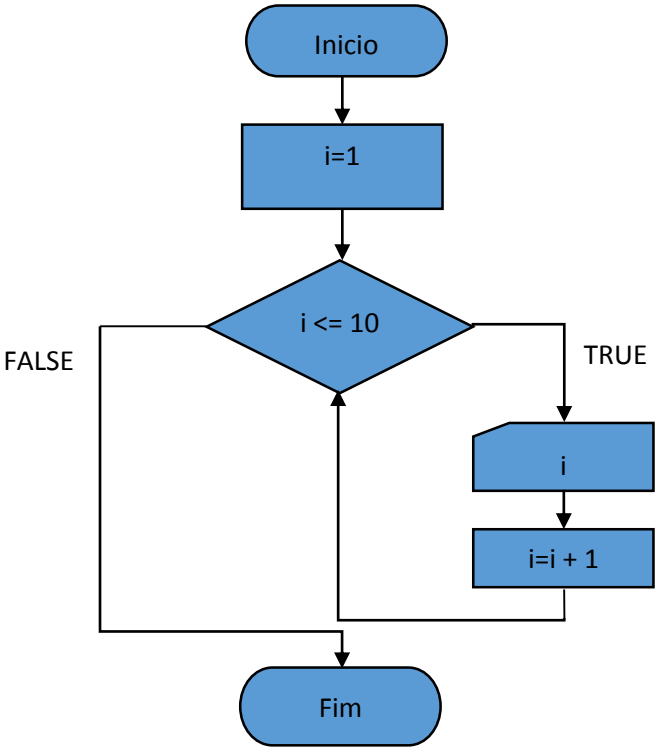
Esquema detalhado



Algoritmo com o uso da condição “while”

Problema: Escrever um número de 1 a 10.

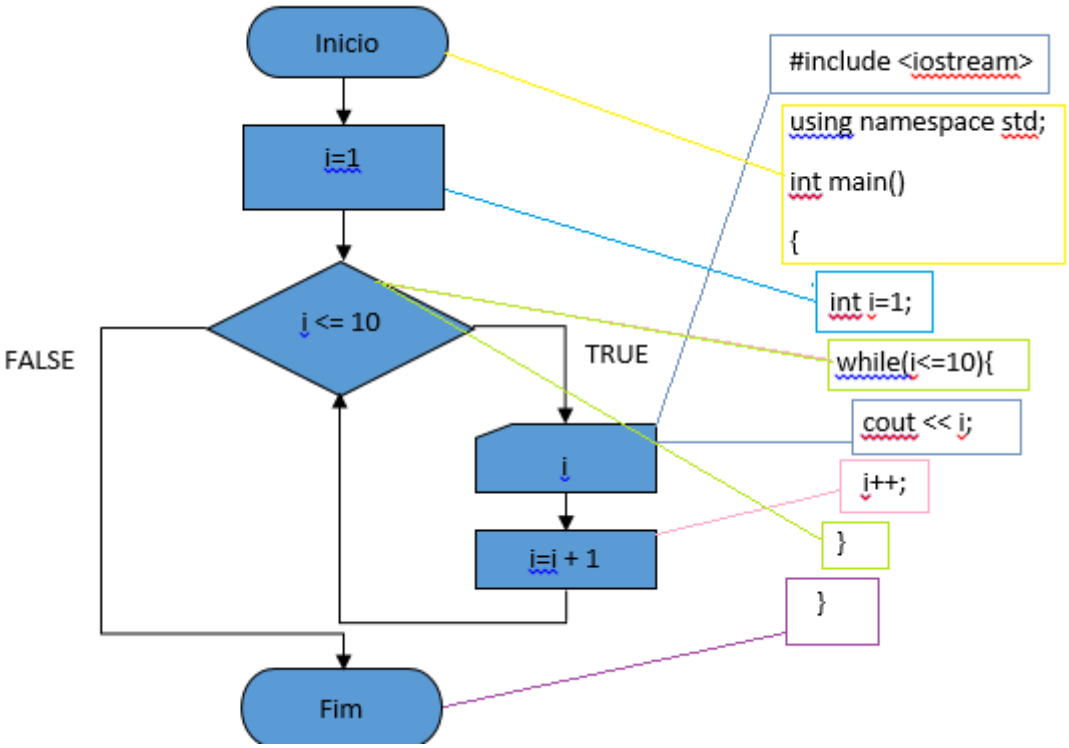
Fluxograma



Código

```
#include <iostream>

using namespace std;

int main()

{

    int i=1;

     while(i<=10){

        cout << i;

        i++;

     }

  }
```

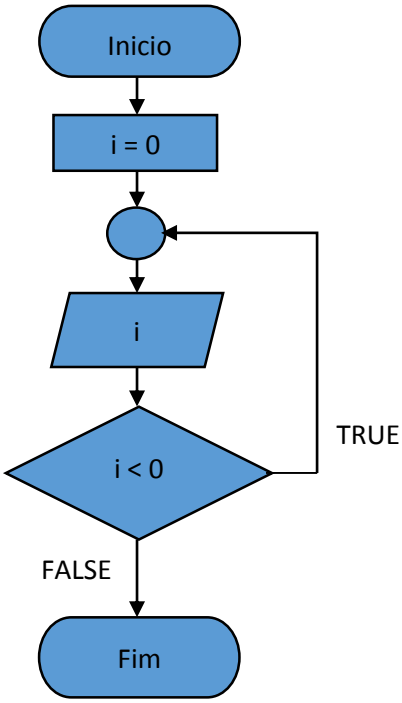
Esquema detalhado



Algoritmo com o uso da condição " do while"

Problema: Pedir um número positivo.

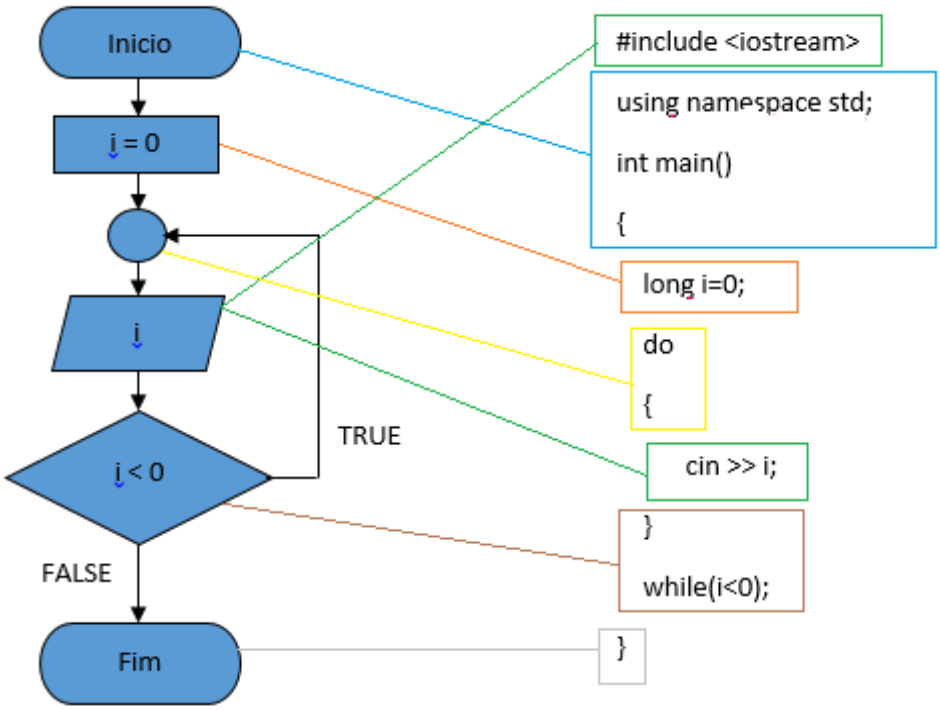
Fluxograma



Código

```
#include <iostream>

using namespace std;

int main()

{

   long i=0;

   do

   {

      cin >> i;

   }

   while(i<0);

}
```

Esquema detalhado

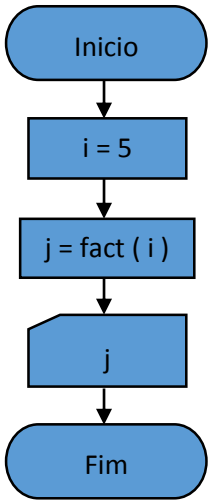


Algoritmo com o uso de uma função

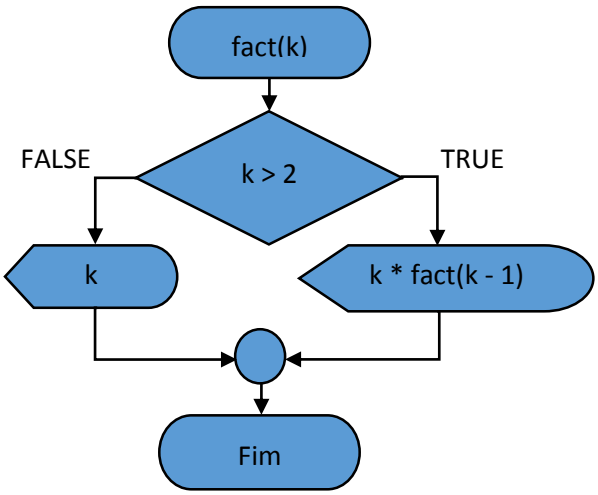
Problema: Factorial de um número.

Fluxogramas

Código principal



Função fact(k)



Código

```
#include <iostream>
using namespace std;
long fact(long k)
{
   if (k>2) {
      return k*fact(k-1);
   }   else{
      return k;
   }
}

int main()
{
   long i=5;
   long j;
   j=fact(i);
   cout << j;
}
```

Esquema detalhado