

TECNOLOGIAS DE PROGRAMAÇÃO DE SISTEMAS DE
INFORMAÇÃO

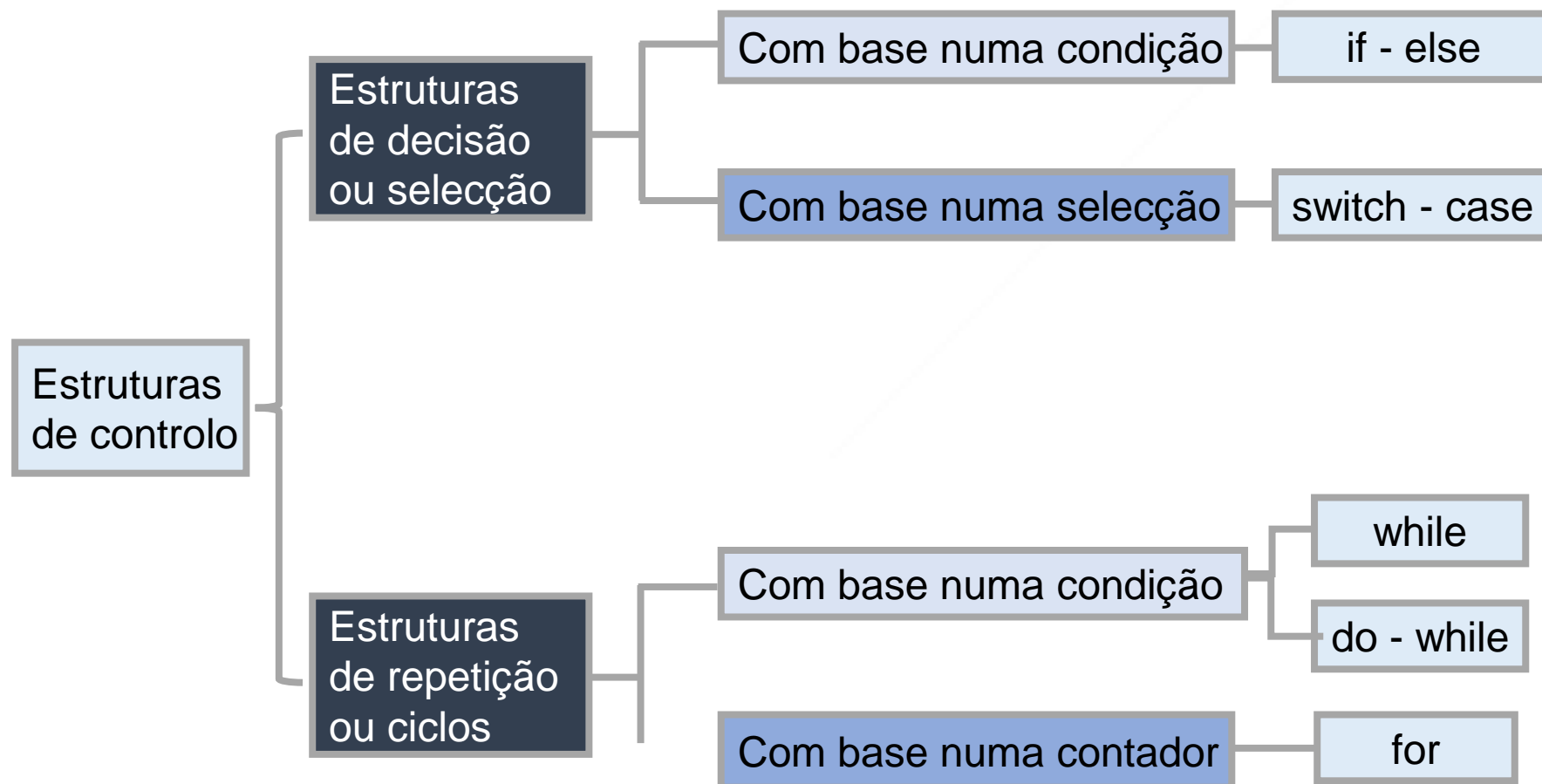
ESTRUTURAS DE CONTROLE

PROGRAMAÇÃO ORIENTADA A OBJECTOS | Prof. Doutora Frederica Gonçalves

Cofinanciado por:



UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS



UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Condição **if-else** :
 - Estrutura de decisão com base numa condição if _ else :

```
if (condição){  
    instrução 1;  
    Instrução 2;  
    ...;  
}  
else{  
    instrução 1;  
    Instrução 2;  
    ...;  
};
```

```
bloco_de_instruções =  
    instrução 1;  
    Instrução 2;  
    ...;
```

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Condição **if-else** :
 - Depois da palavra **if** surge, dentro de parênteses curvos, uma **condição** ou uma **expressão booleana**;
 - Se essa condição for verdadeira, será executado um bloco de instruções
 - caso contrário:

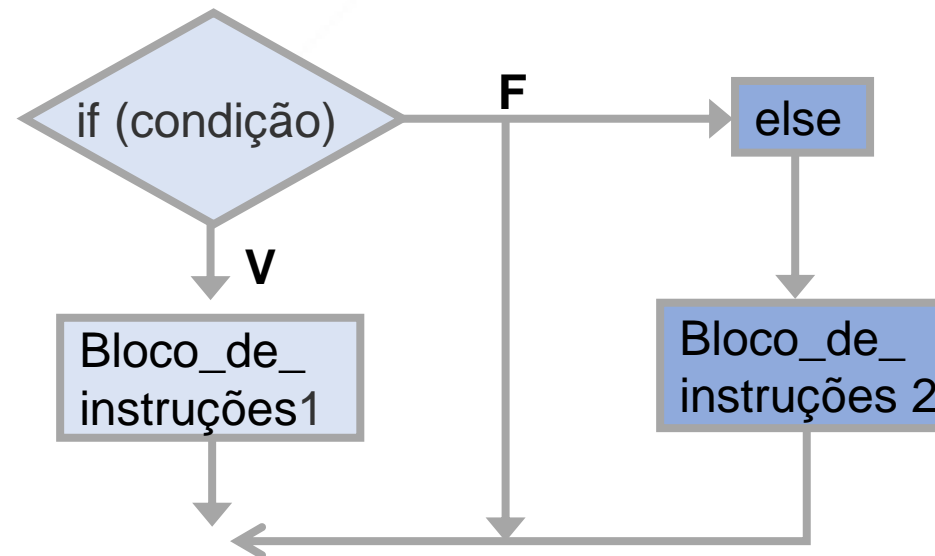
Se não existir cláusula else, não acontece nada e passa-se à instrução que se segue no final da estrutura if;

Se existir uma cláusula else, será executado o bloco de instruções que se segue

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Condição **if-else**: representação esquemática

```
if (condição)
    bloco_de_instruções1;
else
    bloco_de_instruções2;
```



UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Condição **if-else**:

```
...  
    int nota;  
        cout << "Introduza a nota do aluno: ";  
        cin >> nota;  
  
    if (nota < 10)  
        cout << "Reprovado ";  
    else  
        cout << "Aprovado ";  
...
```

As estruturas de decisão **if - else** podem assumir uma forma mais extensa, incluindo diversos if e else encaixados.

```
if (condição)  
    < bloco_de_instruções >;  
else if (condição)  
    < bloco_de_instruções >;  
else if (condição)  
    < bloco_de_instruções >;  
.....  
else  
    < bloco_de_instruções >;
```

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Condição **if-else**:

```
...  
    int nota;  
    cout << "Digite a nota: "; cin >> nota;  
    if (nota < 0)  
        cout << "Nota nao valida ";  
    else if (nota < 10)  
        cout << "Reprovado ";  
    else if (nota < 14)  
        cout << "Suficiente ";  
    else if (nota < 18)  
        cout << "Bom ";  
    else if (nota <= 20)  
        cout << "Muito Bom ";  
    else  
        cout << "Nota nao valida ";  
...
```

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Operador condicional**

- O C++ fornece um operador especial que frequentemente pode ser utilizado no lugar da instrução **if – else**, o **operador condicional**.

```
if (nota < 10)
    cout << "Reprovado ";
else
    cout << "Aprovado ";
```

```
(nota < 10) ?
    cout << "Reprovado ":
    cout << "Aprovado ";
```


UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Operador relacionais ou comparativos**

Operador	Significado
==	Igual a (x==0 significa x tem um valor igual a 0)
!=	Diferente de (x !=0 significa x diferente de 0)
<	Menor que (x < 0 significa x menor que 0)
>	Maior que (x > 0 significa x maior que 0)
<=	Menor ou igual (x <= 0 significa x menor <= ou igual que 0)
>=	Maior ou igual (x >= 0 significa x maior ou igual que 0)

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Operador lógicos**

Operador	Significado
&&	And – Conjunção Lógica
 	Or – Disjunção Lógica
!	Not – Negação Lógica

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Operadores relacionais** ou **comparativos**

$x < 2$ (x é menor que dois)

- Em c++, o valor lógico de uma expressão booleana pode ser:
 - Um dos habituais valores lógicos: **True** (verdadeiro) ou **false** (falso);
 - Um valor do tipo inteiro, sendo que neste caso é:
 - zero é equivalente a **false**;
 - Qualquer outro valor inteiro diferente de zero que é equivalente a **true**.

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Operadores lógicos e respectivas tabelas

P	Q	P&&Q
V	V	V
V	F	F
F	V	F
F	F	F

P	Q	P Q
V	V	V
V	F	V
F	V	V
F	F	F

P	!P
V	F
F	V

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS


- **Operadores lógicos e respetivas tabelas : Exemplo**

<code>(n>=3) && (n<=9)</code>	→	Assume o valor de verdadeiro se n for maior ou igual a três e n menor ou igual que nove;
<code>(n>=3) (n<=9)</code>	→	assume o valor de verdadeiro se n for maior ou igual a cinco e n menor ou igual que dez;
<code>!(n==0)</code>	→	Negação de n igual a zero, ou seja, a expressão é verdadeira se n for diferente de zero;

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Ordem de prioridades dos operadores aritméticos, relacionais e lógicos em ++:**

Maior
prioridade



Menor
prioridade

operador	acção
+, -, ++, --, !	positivo, sinal negativo, incremento, decremento e Not ou negação lógica;
*, /, %	Multiplicação, divisão, resto da divisão;
+, -	Adição, subtracção;
<, <=, >, >=	Menor que, menor ou igual, maior que, maior ou igual;
==, !=	Igual a, diferente de;
&&	E lógico
 	Ou lógico
=, +=, -=, *=, etc	Atribuição simples, etc

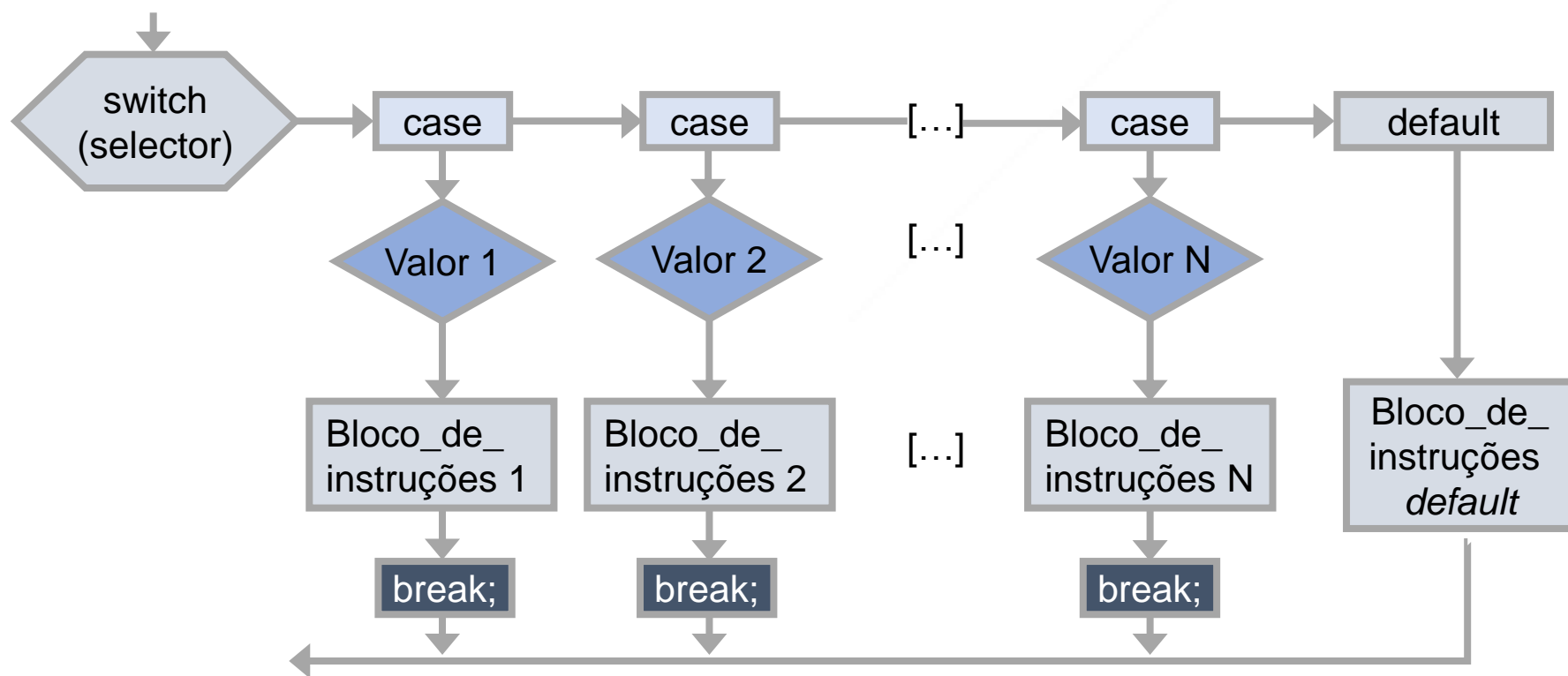
UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Estrutura de selecção **switch – case**
 - Estrutura de controlo que consiste numa selecção entre vários casos possíveis

```
switch (selector) {  
    case <valor1>:  
        <bloco_de_instruções1>;  
        break;  
    case <valor2>:  
        <bloco_de_instruções2>;  
        break;  
    ...  
    [ default : <bloco_de_instruções>; ]  
}
```

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Representação estrutura de selecção switch – case



UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Estrutura de selecção switch – case

```
...  
    int n;  
    cout <<"Introd. um numero: ";  
    cin >> n;  
  
    switch (n){  
        case 0: cout <<"numero = 0\n";  
        break;  
    default: cout << "numero != 0\n";  
    }  
...
```

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Ciclo **while**

- O formato do ciclo **while** é o seguinte:

```
while (condição)
    <bloco _de _instruções>;
```

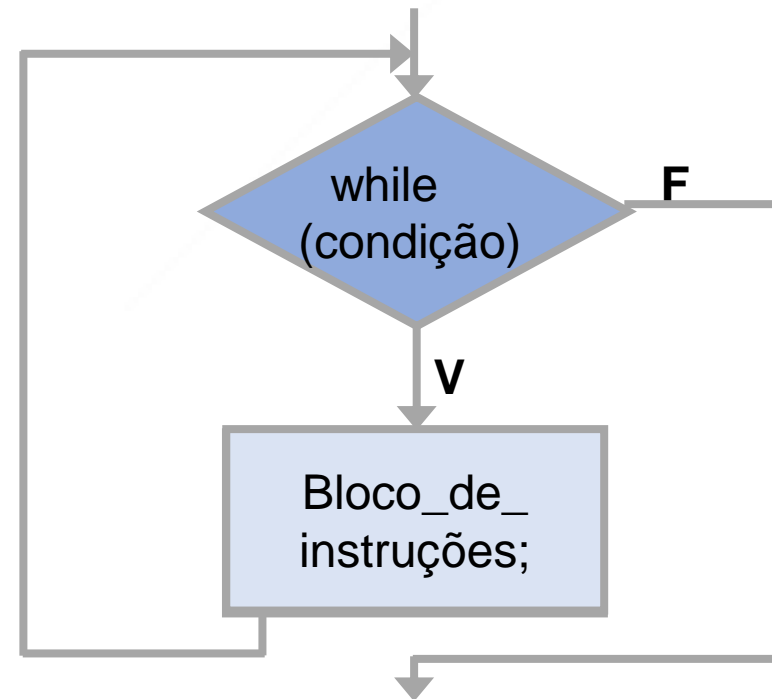
- A condição controla a execução do ciclo. Se, e enquanto a condição for verdadeira, o bloco de instruções é executado. Quando a condição deixa de ser verdadeira o ciclo termina.

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Representação esquemática ciclo **while**

```
while (condição){  
    bloco_de_instruções;  
}
```

```
...  
char c = 'A';  
while(c<= 'D')  
    cout <<c++;  
...
```



UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Ciclo **do while**

- O formato do ciclo **do - while** é o seguinte:

```
do
    < bloco_ de_ instruções>
while
    (condição);
```

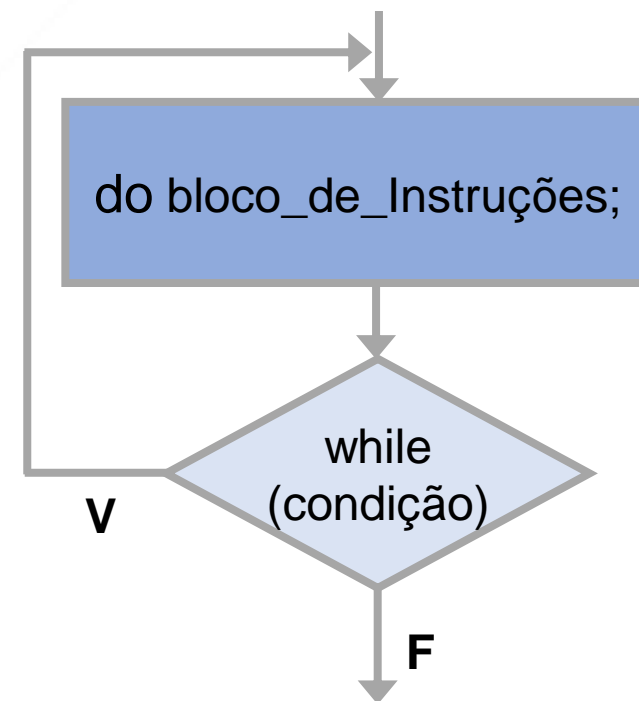
- É uma variante dos ciclos em que as instruções nela contidas executam sempre pelo menos uma vez, visto que a condição que controla o bloco surge apenas no fim do bloco de instruções.

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Representação esquemática ciclo **do while**

```
do  
    bloco_de_instruções  
while  
    (condição);
```

```
...  
int n;  
do {  
    cout << "Introduza um numero: ";  
    cin >> n;  
}  
while ( n < 1 || n > 10 );  
    cout << " Numero introduzido" << n << "\n";  
...
```



UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- Ciclo **for**

- O formato do ciclo **for** é o seguinte:

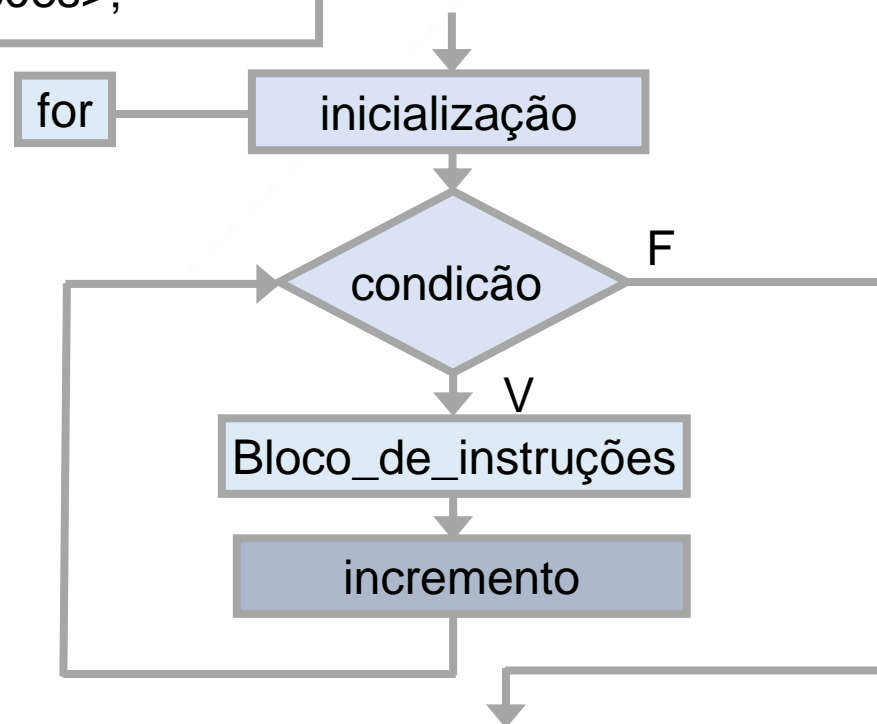
```
for (inicialização; condição; incremento)  
    <bloco _de _instruções>;
```

- Inicialização de uma variável contador;
- Condição de controlo do ciclo;
- Actualização da variável contador;

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

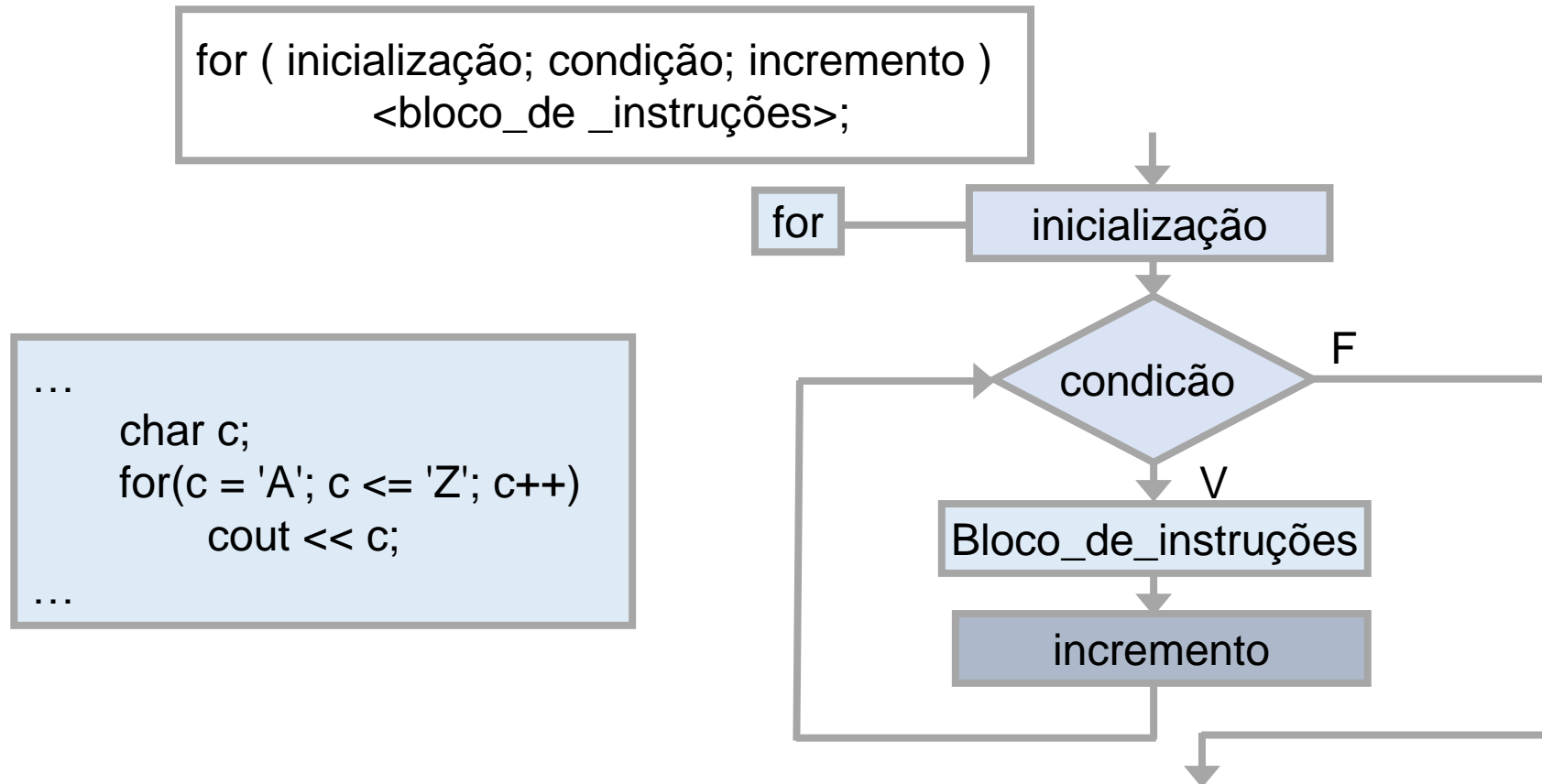
- Representação esquemática do Ciclo **for**

```
for ( inicialização; condição; incremento )  
    <bloco_de _instruções>;
```



ESTRUTURAS DE CONTROLO EM C++ (26)

- Representação esquemática do ciclo **for**





CTeSP

CURSOS TÉCNICOS
SUPERIORES PROFISSIONAIS