

Estruturas Condicionais

Algoritmos e Estruturas de Dados I

Prof. Lucas Astore

Prof. Cristiano Rodrigues

Revisão

Revisão

- Tipos de variáveis:

inteiro	0 , 1 , 50, ..
---------	----------------

float	3.5 , 2.45 ..
-------	---------------

double	234.434, 122.1
--------	----------------

caractere[]	"Olá mundo"
-------------	-------------

caractere	'a' , 'r' ..
-----------	--------------

boolean	true, false
---------	-------------

```
#include <stdio.h>
```

→ diretivas

```
int main() {
```

→ cabeçalho da função

```
    int a, b;
```

```
    double val;
```

```
    float val2;
```

```
    char c;
```

→ declarações

```
    printf("Hello"
```

```
    " World\n");
```

```
    scanf("%d %d %lf %f %c",&a,  
&b,&val,&val2, &c);
```

```
    printf("%d %d %.2lf %.2f %c",a,  
b,val,val2,c);
```

```
    return 0;
```

```
}
```

→ instruções

Revisão

- Entrada e Saída de dados

```
scanf("formatação", arg1, arg2, ..., argn);
```

especificadores de formato

endereços de memória

```
...  
int idade;  
char sexo;  
...  
scanf("%d %c", &idade, &sexo);  
...
```

```
printf("formatação", arg1, arg2, ..., argn);
```

%d %f %s

especificadores
de formato
(iniciam com %)

caracteres de
controle
(iniciam com \)

caracteres
comuns
(demais)

valores

substituído pelo
argumento
correspondente

produz efeito
especial

exibido
literalmente

Revisão

- Entrada e Saída de dados

Especificador	Tipo
%d	int
%.xf	float (opcional x casas decimais)
%.xf	double (opcional x casas decimais)
%s	char[]
%b	bool
%c	char

Operadores Matemáticos

Símbolo	Exemplo	Operação
+	2 + 3 X + 3	Soma
-	3 - 5	Subtração
*	2 * 6	Multiplicação
/	3.0 / 2	Divisão
%	3 % 2	Resto da divisão

Operadores Atribuição

Símbolo	Exemplo	Equivale
+= -= *= /=	x += y x *= 9	x = x + y x = x * 9
%= ++ --	 x++ y--	 x = x + 1 y = y - 1

Operadores Relacionais (Binários)

Símbolo	Exemplo	Operação
<code>==</code>	<code>x == y</code> <code>x == true</code> <code>nome == "Pedro"</code>	Igualdade
<code>!=</code>	<code>presente != true</code>	Diferença
<code><</code>	<code>idade < 17</code>	Menor
<code><=</code>	<code>valorCompra <= 100</code>	Menor igual
<code>></code>	<code>altura > 1.70</code>	Maior
<code>>=</code>	<code>idade >= 18</code>	Maior igual

Operadores Lógicos

Símbolo	Exemplo	Equivale
<code>!</code>	<code>!achou</code> <code>!(true) = false</code> <code>!(false) = true</code>	Negação
<code>&&</code>	<code>Op1 && Op2</code>	AND
<code> </code>	<code>Op1 Op2</code>	OR

E lógico (AND)

Op1	Op2	Op1 && Op2
V	V	V
V	F	F
F	V	F
F	F	F

Operadores Lógicos

Símbolo	Exemplo	Equivale
!	!achou !(true) = false !(false) = true	Negação
&&	Op1 && Op2	AND
	Op1 Op2	OR

OU lógico (OR)

Op1	Op2	Op1 Op2
V	V	V
V	F	V
F	V	V
F	F	F

Estrutura sequencial

Estrutura sequencial

Os comandos são separados por ponto e vírgula e executados de forma sequencial, ou seja, na ordem em que eles aparecem

Forma geral:

```
<comando 1>;  
<comando 2>;  
<comando 3>;  
...  
<comando n>;
```

Estrutura sequencial

Exemplo 1

Ler os valores dos catetos de um triângulo retângulo e mostrar a hipotenusa

Algoritmo

```
real a, b, c;  
escrever:  "Entrar com 1o cateto:";  
ler b;  
escrever:  "Entrar com 2o cateto:";  
ler c;  
a = raiz(pow(b,2) + pow(c,2));  
imprimir "Hipotenusa: " + a;
```

Fim Algoritmo

Estrutura Sequencial

- Um bloco de comandos é delimitado por '{' (indica início do bloco) e por '}' (indica final do bloco)
- Os comandos pertencentes a esse bloco estarão dentro dessa delimitação. Sugere-se que essa **hierarquia** seja estabelecida **visualmente por um recuo na escrita (indentação)**.
- O fluxo de execução dentro do bloco será sequencial, **caso não exista uma estrutura de controle** ou chamadas de **funções** que o altere.
- Dizer que o fluxo é sequencial é afirmar que os comandos serão executados linha a linha, ou seja, termina a execução de uma linha para iniciar a próxima, sem retroceder para linhas já executadas

Estrutura condicional

Estrutura condicional

Em nosso dia a dia, quase sempre, temos que tomar decisões

Se fizer sol, *então*, ...

Se idade maior que 18, *então*, ...

Se eu ganhar na mega sena, *então*, ...

Se o meu time ganhar, *então*, ...

Se eu passar em cálculo, *então*, ...

A programação é totalmente relacionada à tomada de decisões

Comando Se

```
se (expressão) então  
    lista de comandos  
fim se
```

C-like

```
if(expressão){  
    lista de comandos  
}
```

Comando Se – Senão

```
se(expressao) então  
    lista de comandos 1  
senão  
    lista de comandos 2  
fim se
```

C-like

```
if(expressao){  
    lista de comandos 1  
} else {  
    lista de comandos 2  
}
```


Comando Se – Senão – Se

```
se (expressão 1) então
    lista de comandos 1
senão se (expressão 2) então
    lista de comandos 2
senão se (expressão 3) então
    lista de comandos 3
    ...
senão
    lista de comandos n
fim se
```

C-like

```
if(expressão 1){
    lista de comandos 1
} else if(expressão 2) {
    lista de comandos 2
} else if(expressão 3){
    lista de comandos 3
    ...
} else {
    lista de comandos n
}
```

Estruturas Condicionais

- Também chamadas de Estruturas de Seleção.
- Possuem uma expressão condicional (ou condição), cujo resultado lógico da sua avaliação (true/false) controlará a ordem do fluxo de execução.
- Dentro de cada bloco podem existir quantos comandos forem necessários.
- Tipos: if, if-else, if-else if

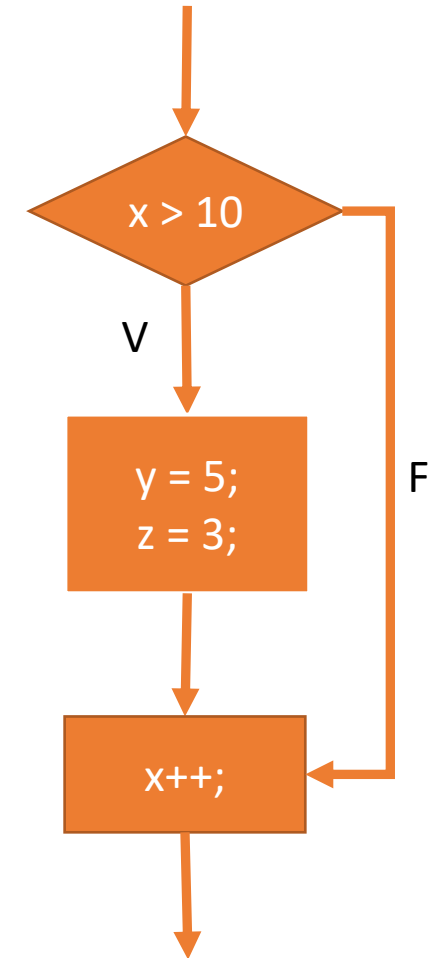
Estruturas Condicionais

Estrutura if

- O bloco de comandos 1 somente será executado se a condição for verdadeira.
- Caso contrário, o bloco inteiro será ignorado e o próximo comando depois da estrutura é avaliado.

```
if (condição A)
{
    ... comandos 1
}
```

```
if (x > 10)
{
    y = 5;
    z = 3;
}
x++;
```



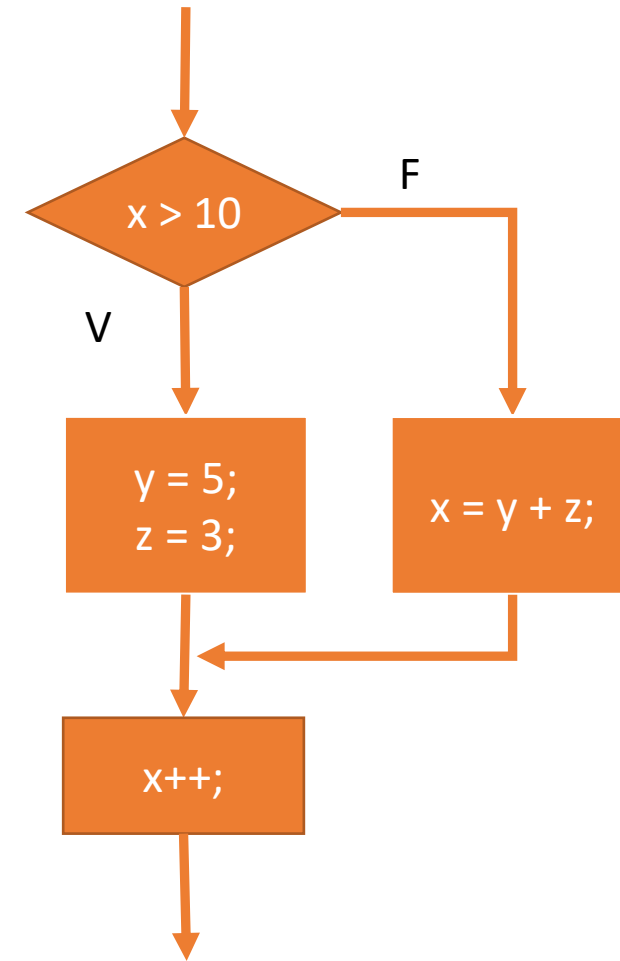
Estruturas Condicionais

Estrutura if-else

- Caso condição inicial for falsa, o bloco inteiro de comandos 1 será ignorado e o bloco de comandos 2 será executado.

```
if (condição A)
{
    ... comandos 1
}
else
{
    ... comandos 2
}
```

```
if (x > 10)
{
    y = 5;
    z = 3;
} else
{
    x = y + z;
}
x++;
```



Estruturas Condicionais

Estrutura if-else if

- Os blocos são avaliados na ordem de cima para baixo, desde que nenhuma condição anterior tenha sido verdadeira. Nesse caso, o bloco da condição verdadeira é executado e os demais são ignorados.
- O bloco do else é executado automaticamente caso todas as condições tenham sido avaliadas como falsas.
- O número de blocos else-if ou mesmo o bloco else são opcionais.

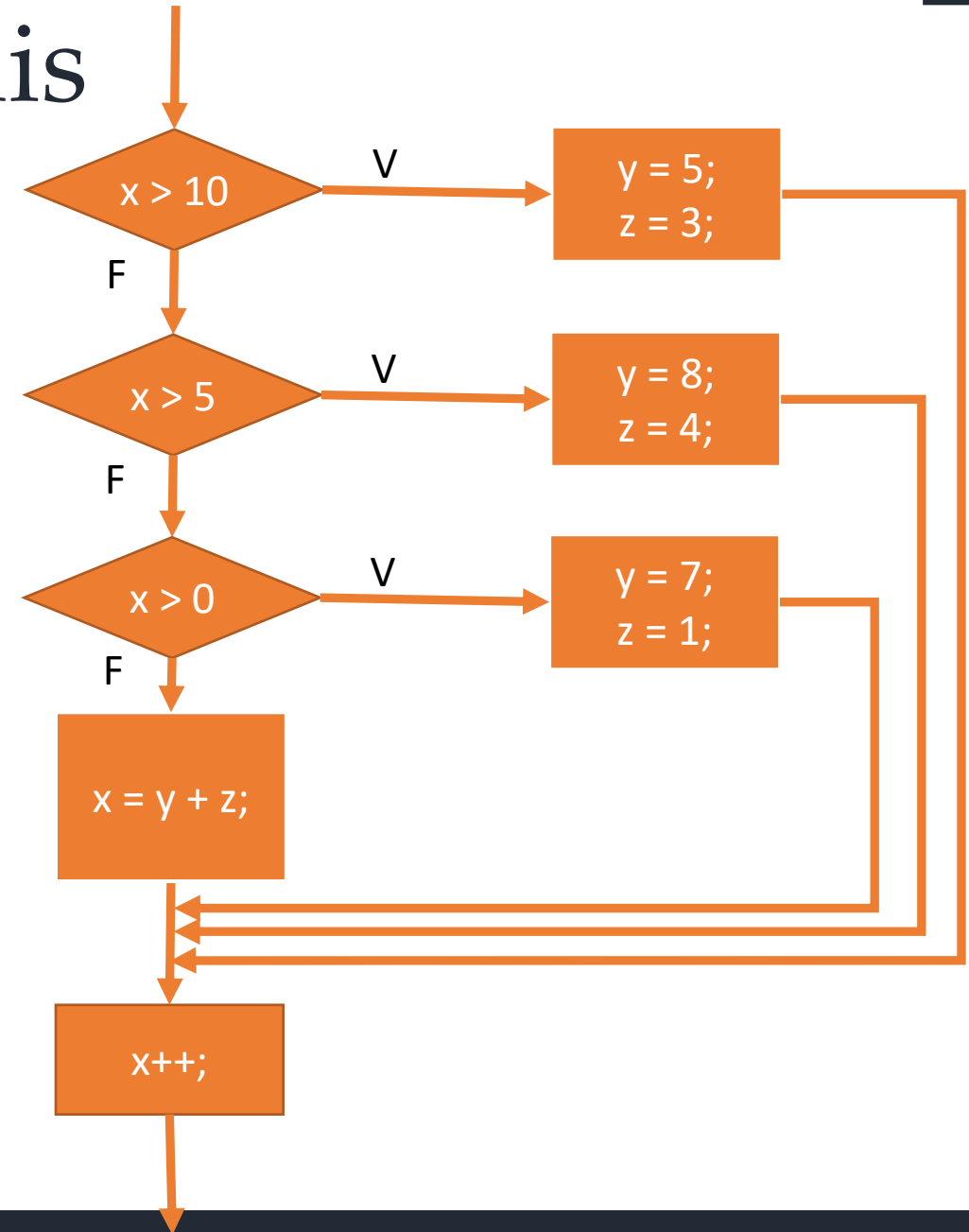
```
if (condição A)
{
    ... comandos 1
}
else if (condição B)
{
    ... comandos 2
}
...
else
{
    ... comandos N
}
```

```
if (x > 10)
{
    y = 5;
    z = 3;
}
else if (x > 5)
{
    y = 8;
    z = 4;
}
else if (x > 0)
{
    y = 7;
    z = 1;
}
else
{
    x = y + z;
}
x++;
```

Estruturas Condicionais

Estrutura if-else if

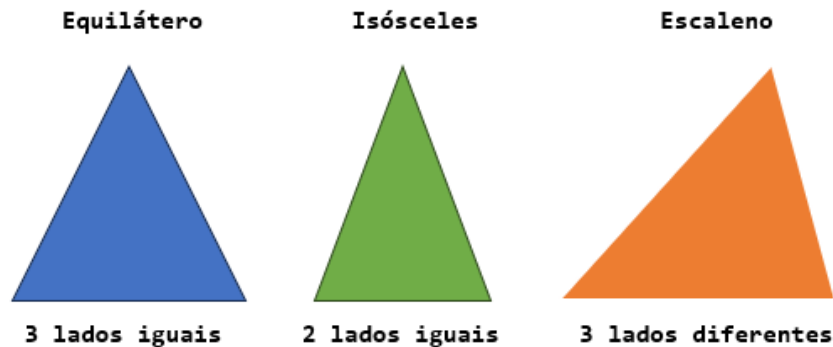
- Os blocos são avaliados na ordem de cima para baixo, desde que nenhuma condição anterior tenha sido verdadeira. Nesse caso, o bloco da condição verdadeira é executado e os demais são ignorados.
- O bloco do else é executado automaticamente caso todas as condições tenham sido avaliadas como falsas.
- O número de blocos else-if ou mesmo o bloco else são opcionais.



Estruturas Condicionais

Estrutura if-else if

- Após avaliar uma condição e ela ser falsa, esse estado permanece ao testar as próximas condições.



```
if (lado1 == lado2 && lado2 == lado3)
{
    tipo = 1; // equilátero e isósceles
}
else if (lado1 != lado2 && lado1 != lado3
&& lado2 != lado3)
{
    tipo = 2; // escaleno
}
else
{
    tipo = 3; // isósceles
}
```

Redundante testar:

```
else if (lado1 == lado2 || lado1 == lado3
```

Exercício

- Leia 3 números inteiros, selecione o menor e o maior e imprima os seus respectivos valores na tela.

Exercício

- Leia **dois** números.
- Se um deles for maior que 45, realize a soma dos mesmos.
- Caso contrário, se os dois forem maior que 20, realize a subtração do maior pelo menor,
- Senão, se um deles for menor do que 10 e o outro diferente de 0 realize a divisão do primeiro pelo segundo.
- Finalmente, se nenhum dos casos solicitados for válido, mostre seu nome na tela.

Exercícios

- O banco do Zé abriu uma linha de crédito para os seus clientes. O valor máximo da prestação não poderá ultrapassar 40% do salário bruto. Faça um algoritmo que permita entrar com o salário bruto e o valor da prestação e informar se o empréstimo será concedido.

Para casa.

Aninhamento do comando Se

```
se ( expressão ) então
    se ( expressão ) então
        ...
    senão
        ...
fim se
senão
    se ( expressão ) então
        ...
    senão
        se ( expressão ) então
            ...
        fim se
    fim se
fim se
```

Estruturas Condicionais

Estruturas Aninhadas

- Como dentro de qualquer bloco podem ser inseridos comandos, a estrutura condicional também é um comando.
- Dessa forma podem existir estruturas condicionais dentro de outras do mesmo tipo, quantos níveis forem necessários.

```
// Deixar conteúdos em ordem  
// decrescente em a, b, c
```

```
if (b > a && b > c)  
{  
    // troca conteúdo  
    // de b com a  
    aux = a;  
    a = b;  
    b = aux;  
}  
else if (c > a && c > b)  
{  
    // troca conteúdo  
    // de c com a  
    aux = a;  
    a = c;  
    c = aux;  
}
```

Considerações sobre o comando Se

O { e } é obrigatório quando o if ou o else tiver mais de um comando

Quando eles tiverem exatamente um comando, o { e } é facultativo

Uma ótima prática de programação é sempre utilizá-los

Onde se lê ótima prática de programação entende-se sempre faça isso
CUIDADO com ifs aninhados

O else a seguir pertence a qual if?

```
if (n > 0)
    if (a > b)
        z = a;
else
    z = b;
```

O else a seguir pertence a qual if?

Sempre associamos o else ao if mais interno

```
if (n > 0)
    if (a > b)
        z = a;
else
    z = b;
```

E agora?

```
if (n > 0) {  
    if (a > b)  
        z = a;  
} else  
    z = b;
```


Operador Ternário: Comando condicional enxuto

```
(expressão) ? valor1 : valor2 ;
```

Operador Ternário: Comando condicional enxuto

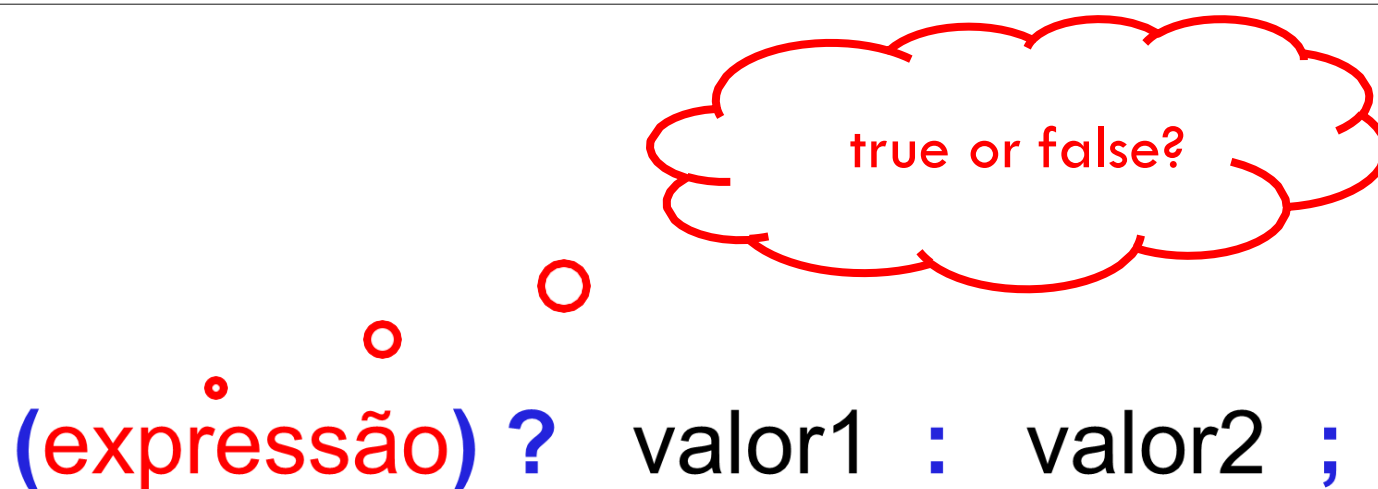


```
(expressão) ? valor1 : valor2 ;
```

Operador Ternário: Comando condicional enxuto

```
(expressão) ? valor1 : valor2 ;
```

Operador Ternário: Comando condicional enxuto




The diagram illustrates the syntax of the ternary operator. It shows the expression `(expressão) ? valor1 : valor2 ;` where `(expressão)` is in red, `?` is in blue, `valor1` and `valor2` are in black, and `:` and `;` are in blue. Above the `?` is a small red circle, and above that is another small red circle. A red thought bubble is connected to the second small circle, containing the text `true or false?`.

`(expressão) ? valor1 : valor2 ;`

true or false?

Operador Ternário: Comando condicional enxuto



The diagram illustrates the logic of the ternary operator. It shows the expression `(expressão) ? valor1 : valor2 ;` in blue text. Below the opening parenthesis, the word `true` is written in red. A red arrow points from `true` up to the opening parenthesis. Another red arrow points from the opening parenthesis to the opening curly brace of the first branch, `valor1`. A third red arrow points from the opening curly brace of `valor1` down to the text `valor1`.


`(expressão) ? valor1 : valor2 ;`
`true`

Operador Ternário: Comando condicional enxuto

(expressão) ? valor1 : valor2 ;



Operador Ternário: Comando condicional enxuto



(expressão) ? valor1 : valor2 ;

false

The diagram illustrates the ternary operator syntax: (expressão) ? valor1 : valor2 ;. A red arrow originates from the 'false' label below the condition '(expressão)', points upwards to the opening curly brace of the condition, then horizontally to the right, and finally downwards to the colon separator, indicating the path for the false branch.

Operador Ternário: Comando condicional enxuto

(expressão) ? valor1 : valor2 ;



Exemplo

```
if (a > b) {  
    c = a*a;  
} else {  
    c = b;  
}
```

- Como reescrever usando o operador ternário?

Exemplo

```
if (a > b) {  
    c = a*a;  
} else {  
    c = b;  
}
```

```
c = (a > b) ? a*a : b;
```

Exercício

- Faça um programa que leia dois números **a** e **b** e mostre o maior deles na tela. Resolva usando o if ou ef/else e, depois, resolva usando o operador ternário.

Switch case



Comando Escolhe

```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```

Comando Escolhe



escolhe (valor)

caso valor1 :

lista de comandos 1

caso valor2 :

lista de comandos 2

caso valor3 :

lista de comandos 3

⋮

...

padrão:

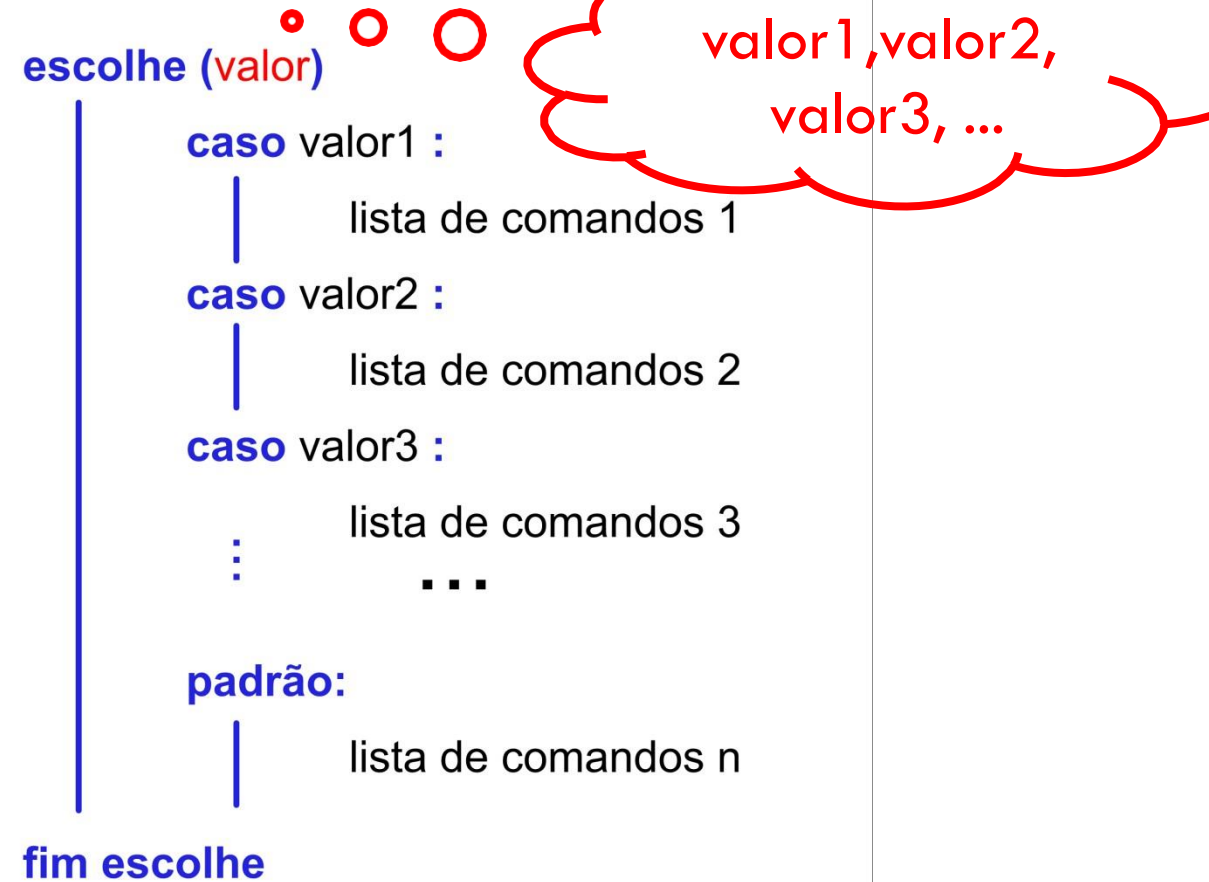
lista de comandos n

fim escolhe

Comando Escolhe

```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```

Comando Escolhe



Comando Escolhe

```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```

Comando Escolhe

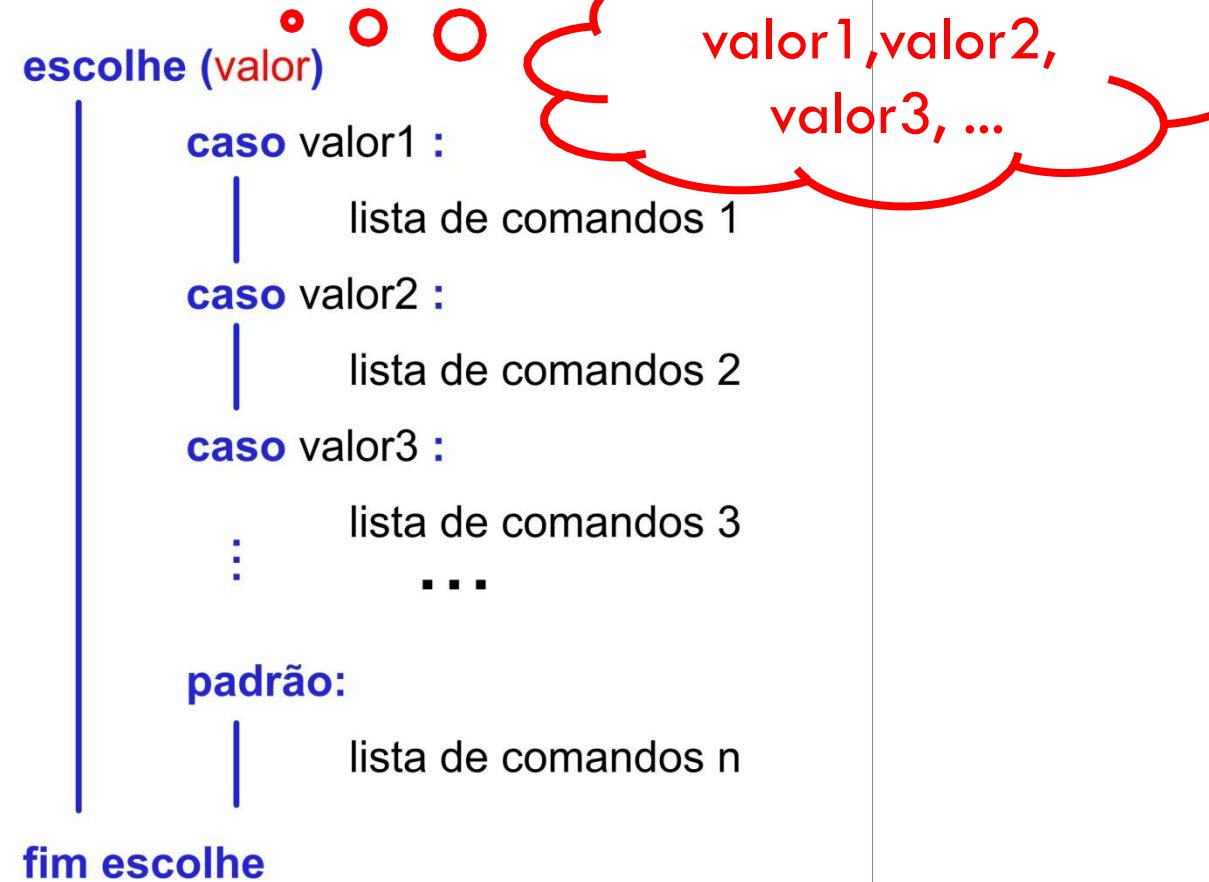
```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```

Comando Escolhe

```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```



Comando Escolhe



Comando Escolhe

```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```

Comando Escolhe

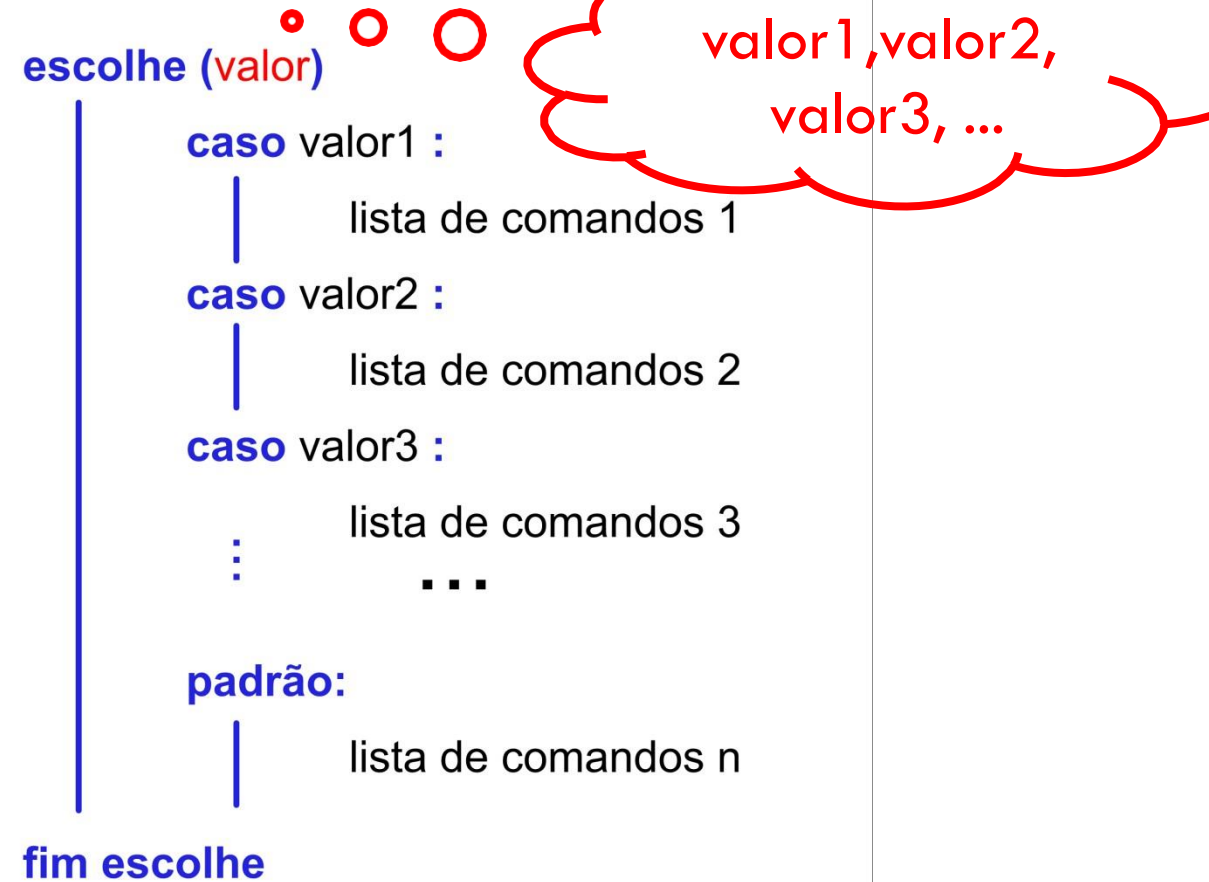
```
escolhe (valor)
|
|      caso valor1 :
|          lista de comandos 1
|
|      caso valor2 :
|          lista de comandos 2
|
|      caso valor3 :
|          lista de comandos 3
|          ...
|
|      padrão:
|          lista de comandos n
|
fim escolhe
```

Comando Escolhe

```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```



Comando Escolhe



Comando Escolhe

```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```

Comando Escolhe

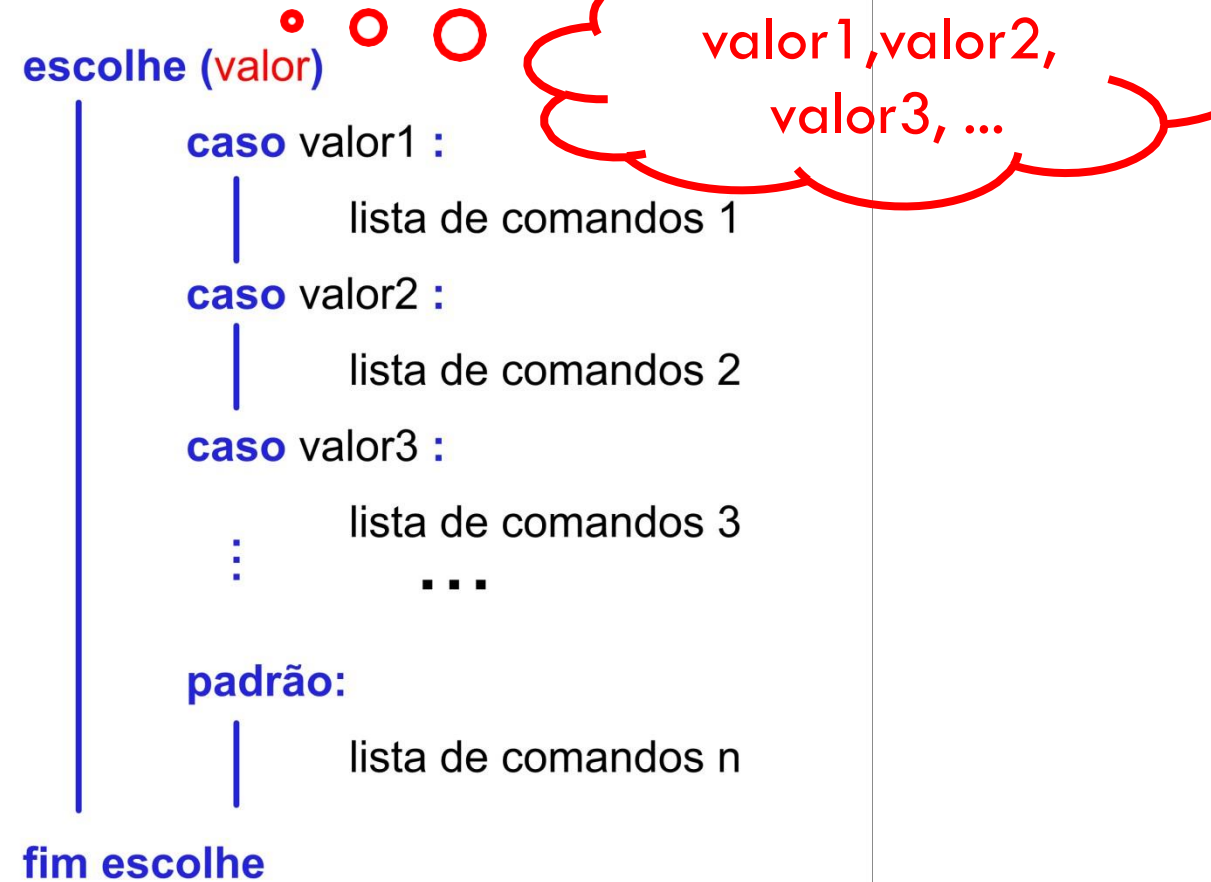
```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```

Comando Escolhe

```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```



Comando Escolhe



Comando Escolhe

```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```

Comando Escolhe

```
escolhe (valor)
|
|  caso valor1 :
|      lista de comandos 1
|
|  caso valor2 :
|      lista de comandos 2
|
|  caso valor3 :
|      lista de comandos 3
|      ...
|
|  padrão:
|      lista de comandos n
|
fim escolhe
```



Comando Escolhe em C-like: **switch**

```
switch (valor) {  
    case valor1 :  
        lista de comandos 1  
        break;  
    case valor2 :  
        lista de comandos 2  
        break;  
    case valor3 : case valor4 :  
        lista de comandos 3  
        break;  
  
    default:  
        lista de comandos n  
}
```

Estrutura Switch

- Esse é um tipo de estrutura de seleção, na qual ocorre a avaliação da correspondência de uma expressão com as opções disponíveis.
- Utilizaremos o padrão de uma constante, ou seja, o conteúdo de uma variável é comparado às constantes disponíveis nas opções. Apenas uma opção é executada ou nenhuma delas.
- É opcional indicar a situação *default*, que é executada no caso de não ocorrer correspondência com as demais opções.

```
switch (variável)
{
    case const1: ... comandos 1....
                    break;
    case const2: ... comandos 2....
                    break;
    ....
    case constN: ... comandos N ....
                    break;
    default: ... comandos Z....
                    break;
}
```


Estrutura Switch

- Em C, a expressão tem que resultar em um char ou int.
- O uso de *break* no final de cada caso é opcional, inclusive para *default*. Mas ele é necessário para se ter situações mutuamente exclusivas.

```
int num;
printf("Digite um número: ");
scanf("%d",&num);
switch(num)
{
    case 9:
        printf("\n0 número é igual a 9");
        break;
    case 10:
        printf("\n0 número é igual a 10");
        break;
    default:
        printf("\n0 número não eh nem 9 nem 10\n");
}
return 0;
```

Exercício

Faça um programa que leia um caractere, identifique-o e escreva na tela se ele é um ponto, uma vírgula ou outro sinal. Use o comando switch-case.

Exercício

```
char ch;
    ler ch;

switch( ch ) {
    case '.':
        escrever: "Ponto";
        break;
    case ',':
        escrever: "Vírgula";
        break;
    case ';':
        escrever: "Ponto e vírgula";
        break;
    default:
        escrever: "Não é pontuação";
}
```

Exercício

Faça um programa que leia um número inteiro, garanta que o mesmo está entre 1 e 12 e escreva o nome do mês correspondente. Use o comando switch-case.

Exercício

```
int mes;
```

```
ler mes;
```

```
while ( (mes >= 1 && mes <= 12)
        == false) {
    ler mes;
}
```

```
switch (mes) {
    case 1:
        escrever: "janeiro";
        break;

    case 2:
        escrever: "fevereiro";
        break;
```

• • •

```
case 12:
    escrever: "dezembro";
    break;

default:
    escrever: "Mês invalido";
}
```

Exercícios

Para o programa anterior, comente os breaks dos meses 2, 4, 5 e 9, compile seu código e execute seu programa para o mês 2. Descreva o que aconteceu

Em seguida, execute para o mês 3. Descreva o que aconteceu.

Depois, para o mês 5. Descreva o que aconteceu.