COMANDOS CONDICIONAIS ANINHADOS

Composição de estruturas de decisão

Compondo decisões

Cada bloco que compõe um *if*, seja simples ou composto, é formado por um ou mais comandos. Conforme já mencionado, caso seja apenas um comando, o uso de chaves é opcional. Caso seja utilizado mais de um comando, o uso de chaves torna-se obrigatório, para que seja determinado o bloco de comandos que ficará englobado pelo *if* ou *else*.

Mas, ainda, poderão ocorrer outros casos:

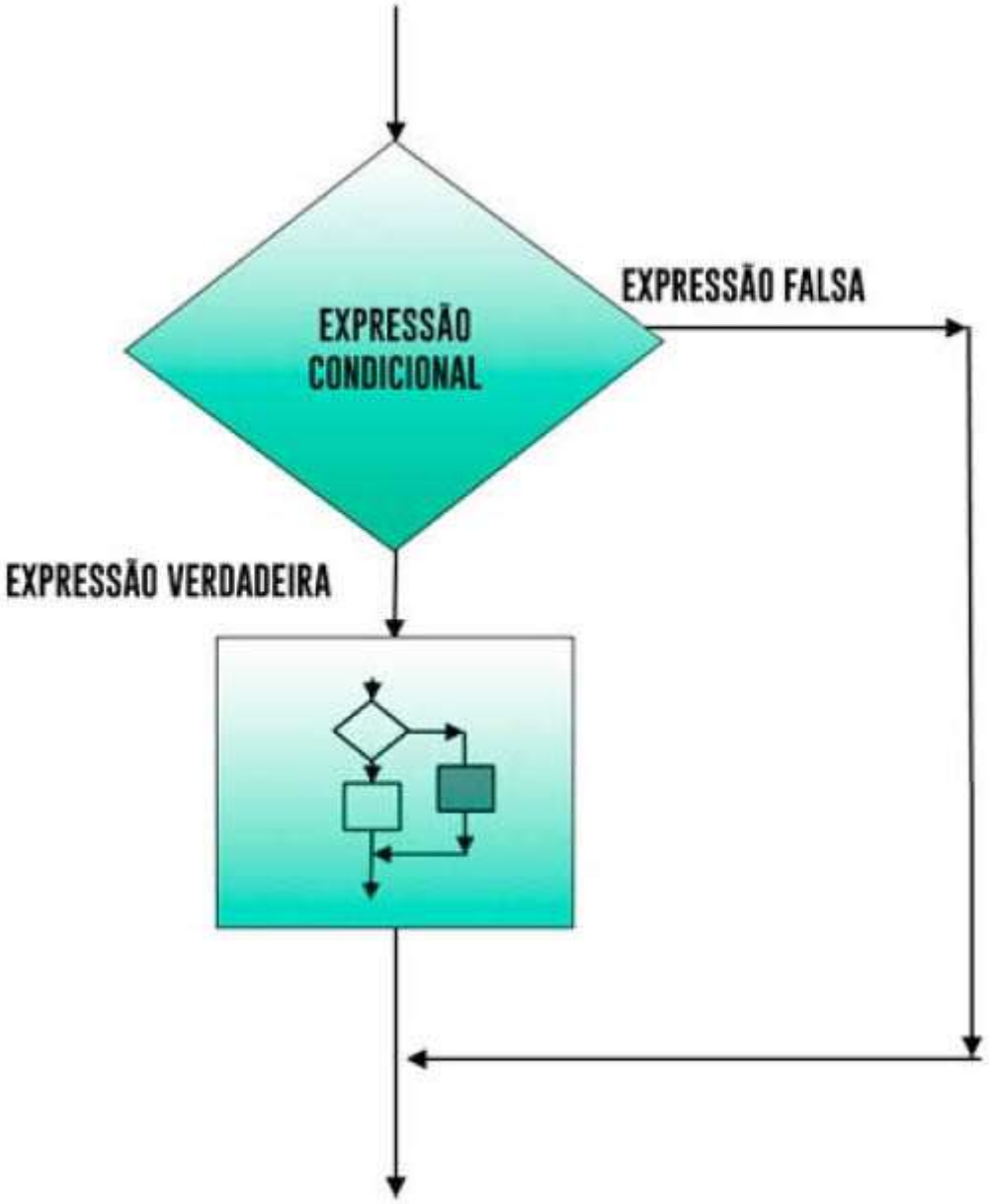
1 Quando um dos comandos que compõe o *if* (ou *else*) for outro comando *if* (simples ou composto), temos o que é chamado de *if* aninhado.

2 Quando dois comandos ifs forem sequenciais, teremos o caso de *if* encadeado.

Descreveremos estes dois tipos nas próximas seções.

Estruturas de decisão aninhadas

Este é o primeiro caso descrito acima, quando um dos comandos que compõem um *if* (ou *else*) é outro comando *if*, podendo estes comandos serem simples ou compostos. Ou seja, as estruturas são dispostas umas dentro das outras. A seguir, apresentamos uma **estrutura de decisão composta internamente de uma estrutura de decisão simples**.



Agora que já vimos a representação gráfica, partiremos para a representação na Linguagem C. Conforme já mencionado, temos um *if* composto internamente a um *if* simples, assim, esta figura representa o seguinte código em C:

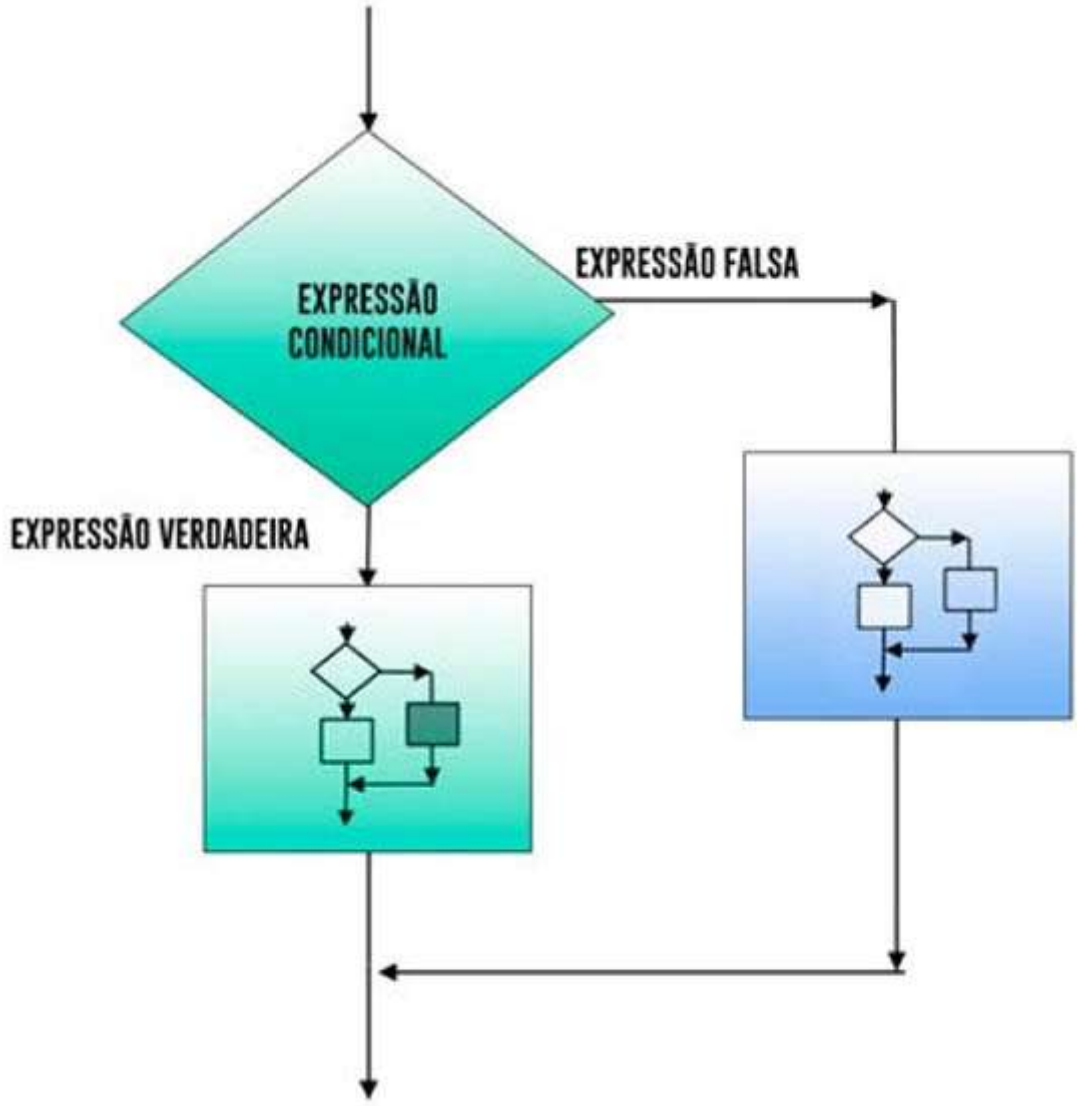
```
if(EXPRESSAO_CONDICIONAL_1){
   if(EXPRESSAO_CONCIDIONAL_2){
      BLOCO_INSTRUCAO_1;
   } else {
      BLOCO_INSTRUCAO_2;
   }
```

}

Se o if interno fosse simples, o código já seria expresso por:

```
if(EXPRESSAO_CONDICIONAL_1){
   if(EXPRESSAO_CONCIDIONAL_2){
      BLOCO_INSTRUCAO_1;
   }
}
```

No caso de as duas estruturas de decisão serem compostas, estas são apresentadas dentro das instruções da estrutura de decisão. Convém salientar que **não há limitação com relação à quantidade de estruturas de decisão** que são dispostas internamente, independentemente do seu tipo, quer sejam simples ou compostas.



Um exemplo de estrutura de decisão aninhada, já na Linguagem C, semelhante ao diagrama anterior, é exibido a seguir.

```
if(EXPRESSAO_CONDICIONAL_1){
   if(EXPRESSAO_CONCIDIONAL_2){
      BLOCO_INSTRUCAO_1;
   } else {
      BLOCO_INSTRUCAO_2;
   }
} else {
   EXPRESSAO_CONCIDIONAL_3
     BLOCO_INSTRUCAO_3;
} else {
   BLOCO_INSTRUCAO_4;
   }
}
```

De forma semelhante, no caso de um if simples ser um comando de uma estrutura else na Linguagem C, seria representado como:

```
if(EXPRESSAO_CONDICIONAL_1){
   BLOCO_INSTRUCAO_1;
} else {
   if(EXPRESSAO_CONCIDIONAL_2){
      BLOCO_INSTRUCAO_2;
   }
}
```

Ou ainda, no caso do if simples ser um comando do if na Linguagem C, seria representado como:

```
if(EXPRESSAO_CONDICIONAL_1){
   if(EXPRESSAO_CONCIDIONAL_2){
      BLOCO_INSTRUCAO_1;
   }
} else {
```

```
    BLOCO_INSTRUCAO_2;
}
```

Exemplo

Numa escola, o aluno é aprovado se sua média é maior que 5,0. O código abaixo valida uma média lida e imprime a mensagem de aprovação. Teste a execução do código, digitando médias inválidas, isto é, menores que zero ou maiores que dez. Em seguida, teste com médias válidas de alunos aprovados e reprovados.

```
#include <stdio.h>


int main (void)
{
   float nota;
   printf ("Digite a nota: ");
   scanf("%f", &nota);
   printf("%f\n",nota);
   if (nota >= 0 && nota <= 10){
      if (nota >= 5)
         printf("Parabéns, você foi aprovado!\n");
      else
         printf("Não desista, você consegue da próxima vez!\n");
   }
   else
      printf("Erro: média inválida!\n");


}
```

Estruturas de decisão encadeadas

Estruturas encadeadas são dispostas de forma sequencial. Independentemente de serem simples ou complexas, estas são dispostas sequencialmente, então, ao término de uma estrutura, é

considerada a entrada da próxima estrutura, conforme o segmento de código a seguir.

```
if(EXPRESSAO_CONDICIONAL_1){
   BLOCO_INSTRUCAO_1;
} else
   if(EXPRESSAO_CONCIDIONAL_2){
      BLOCO_INSTRUCAO_3;
} else {
   BLOCO_INSTRUCAO_4;
   }
```

Neste exemplo, uma estrutura if composta está disposta logo após o else da estrutura anterior. Se esta segunda estrutura fosse um if simples, na Linguagem C seria representado como:

```
if(EXPRESSAO_CONDICIONAL_1){
   BLOCO_INSTRUCAO_1;
} else
   if(EXPRESSAO_CONCIDIONAL_2){
      BLOCO_INSTRUCAO_3;
   }
```

Deve-se notar que em todos os exemplos apresentados até este ponto, BLOCO_INSTRUCAO pode ser representado por um ou mais comandos. Caso seja apenas um comando, o uso das chaves é opcional. Caso possua mais de um comando, o uso das chaves torna-se obrigatório.

Outro ponto que merece destaque é que um comando *if*, conforme o código abaixo, é dito ser apenas um comando, e os blocos de instrução que estão contidos neste código, mesmo que possuam mais de um comando, compõem apenas um, e assim não precisam de chaves antes e depois do *if*.

```
if(EXPRESSAO_CONDICIONAL_1){
   BLOCO_INSTRUCAO_1;
```

```
} else
   BLOCO_INSTRUCAO_2;
   }
```

Exemplo

Vamos repetir o exemplo da análise da aprovação dos alunos, agora com *if* encadeados.

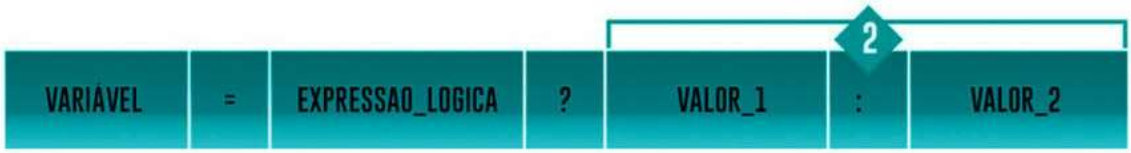
```
#include <stdio.h>
int main (void)
{
float nota;
   printf ("Digite a nota: ");
   scanf("%f", &nota);
   printf("%f\n",nota);
   if (nota < 0)
      printf ("Erro: Média inválida!\n");
   else if (nota > 10)
      printf ("Erro: Média inválida!\n");
   else if (nota >= 5)
      printf ("Aluno aprovado!\n");
   else
      printf("Aluno reprovado!\n");
}
```

Operador ternário

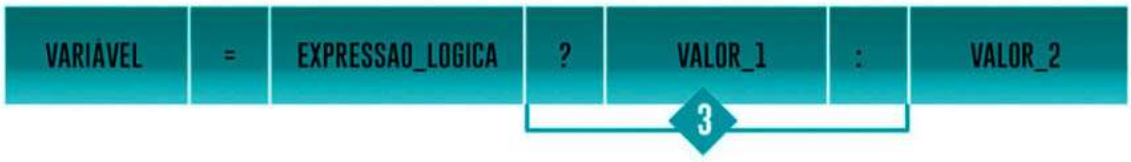
A Linguagem C possui um operador que funciona de forma bem semelhante a um *if*. Este operador utiliza os símbolos de (?) e de (:) para representar a expressão lógica que é utilizada e os valores que comporão o campo do *if* e o campo do *else*, fazendo uma comparação com um *if* composto.

Este operador, conhecido como **operador ternário** por possuir três campos, é apresentado na imagem a seguir.

Esta expressão deverá retornar VERDADEIRO ou FALSO. No caso da Linguagem C, o valor falso deverá ser 0, *null* ou vazio. E verdadeiro será qualquer valor diferente deste.



Valores que serão retornados por este operador. O primeiro será retornado quando o operador for verdadeiro e o segundo no caso de falso.



Os símbolos (?) e (:) deverão ser utilizados nesta ordem, pois, caso contrário, serão identificados pelo compilador como um erro sintático.

VARIAVEL corresponde à variável que receberá os valores, VALOR_1 e VALOR_2, do operador ternário. Assim, caso EXPRESSAO_LOGICA seja verdadeira, VALOR_1 será atribuído à VARIAVEL, caso contrário, VALOR_2 é que será atribuído.

Por exemplo, considere o código abaixo:

```
int a, b, c, d, e;


a=1;

b=2;

c=3;

d=4;


e=(a>b)?c:d;
```

Neste exemplo, temos que **a > b** é falso, pois **a = 1** e **b = 2**. Portanto, conforme já explicado anteriormente, a variável e irá receber o valor **d**.

Caso o comando fosse reescrito como e=(b>a)?c:d; teríamos o valor **c** sendo atribuído à variável **e**.

Atenção!

Operador ternário também permite que seja realizado alinhamento, porém não é uma boa prática de programação.

Estruturas de Múltiplas Alternativas

Esta estrutura, também conhecida como **switch-case**, permite que seja criada uma estrutura condicional que verificará o valor de uma variável de controle, no código abaixo identificado por VARIAVEL, e confrontará a determinados valores A, B e C para que sejam executados os blocos de instrução BLOCO_INSTRUCAO_1, BLOCO_INSTRUCAO_2, BLOCO_INSTRUCAO_3 e BLOCO_INSTRUCAO_4.

```
switch(VARIAVEL){
case A: BLOCO_INSTRUCAO_1;
   break;
case B: BLOCO_INSTRUCAO_2;
   break;
case c: BLOCO_INSTRUCAO_3;
   break;
default: BLOCO_INSTRUCAO_4;
}
```

O campo variável precisará, obrigatoriamente, ser do tipo **char**, **int** e **long**. O que significa que os valores A, B e C precisam também ser deste tipo.

Uma vez que VARIAVEL seja igual a A, por exemplo, a sequência de execução de tarefas continua até que o comando break seja encontrado. Assim, é executado BLOCO_INSTRUCAO_1. Caso a estrutura acima fosse conforme o segmento abaixo, no mesmo valor de VARIAVEL sendo igual a A, o resultado seria a execução de BLOCO_INSTRUCAO_1 e BLOCO_INSTRUCAO_2.

Isto porque o comando break não estava presente.

```
switch(VARIAVEL){
   case A: BLOCO_INSTRUCAO_1;
   case B: BLOCO_INSTRUCAO_2;
      break;
   case c: BLOCO_INSTRUCAO_3;
      break;
```

```
    default: BLOCO_INSTRUCAO_4;
}
```

Outro ponto importante é que no caso de VARIAVEL não ser igual a nenhuma das opções, então o comando DEFAULT é executado. No entanto, deve-se salientar que este comando é opcional.

Exemplo

Imagine que você deseja implementar um menu para a escolha de uma operação em um sistema simples com interface textual. O menu terá as opções abaixo:

1. Inserir novo cliente
2. Consultar cliente por CPF
3. Consultar cliente por nome.
4. Remover cliente da base de clientes.

Qualquer outra tecla para sair do sistema.

Vejamos como podemos implementar isto com a estrutura de múltiplas alternativas

```
#include <stdio.h>

int main (void)

{
    int opcao;

    printf("1 - Inserir novo cliente.\n");

    printf("2 - Consultar cliente por CFP.\n");

    printf("3 - Consultar cliente por nome.\n");

    printf("4 - remover cliente da base de clientes.\n");

    printf("Qualquer outra número para sair.\n");


    printf ("Digite a opção: ");

    scanf("%d", &opcao);


    switch (opcao) {

        case 1:
```

```
            printf("Vamos inserir um novo cliente?\n Digite seu nome:\n Digite seu
CPF:\n");
            break;
        case 2:
            printf("Vamos consultar um cliente?\n Digite seu CPF:\n");
            break;
        case 3:
            printf("Vamos consultar um cliente?\n Digite seu Nome:\n");
            break;
        case 4:
            printf("Vamos remover um cliente?\n Digite seu CPF:\n");
            break;
        default:
            printf("Vamos sair do sistema?\n");
            break;
    }

}
```