

# Fundamentos de Programação

## Qual estrutura condicional usar?

Dainf - UTFPR

Profa. Leyza Baldo Dorini  
Prof. Bogdan Tomoyuki Nassu

# Qual a melhor estrutura em cada caso?



A resposta para esta pergunta não é trivial. Vamos discutir alguns aspectos importantes!

## O que usar - parte 01

### sequência de ifs ou else if?

Considere o seguinte exemplo: dado o IMC calculado, imprimir uma mensagem

- se menor que 18.5, abaixo do peso
- se maior ou igual a 18.5 e menor que 25, normal
- se maior ou igual a 25 e menor que 30, acima do peso
- se maior ou igual a 30, acima de 30

# IMC: solução apenas com if

```
1  #include<stdio.h>
2  int main(){
3      float peso, altura, imc;
4
5      printf("Digite peso e altura");
6      scanf("%f %f", &peso, &altura);
7
8      imc = peso / (altura*altura);
9
10     printf("Seu IMC eh de: %.2f.\n", imc);
11     if (imc < 18.5)
12         printf("abaixo do peso");
13     if (imc >= 18.5 && imc < 25)
14         printf("normal");
15     if (imc >= 25 && imc < 30)
16         printf("acima do peso");
17     if (imc >= 30)
18         printf("30 ou mais");
19
20     return 0;
21 }
```

essa solução não é eficiente porque  
pode fazer muitas comparações  
desnecessárias. Mesmo que o IMC  
calculado seja 15, todas as demais  
condições (linhas 13, 15, e 17)  
serão testadas

portanto, neste código,  
para qualquer valor de IMC  
serão realizadas 8 comparações  
(6 relacionais e 2 lógicas)

# IMC: solução com else if

```
1  #include<stdio.h>
2  int main(){
3      float peso, altura, imc;
4
5      printf("Digite peso e altura");
6      scanf("%f %f", &peso, &altura);
7
8      imc = peso / (altura*altura);
9
10     printf("Seu IMC eh de: %.2f.\n", imc);
11     if (imc < 18.5)
12         printf("abaixo do peso");
13     else if (imc < 25)
14         printf("normal");
15     else if (imc < 30)
16         printf("acima do peso");
17     else
18         printf("30 ou mais");
19
20     return 0;
21 }
```

nesta solução, como as condições estão encadeadas, os testes param assim que a faixa do IMC calculado for encontrada

por exemplo, só vai realizar esta comparação se aquela da linha 11 for falsa

# IMC: solução com else if

```
1  #include<stdio.h>
2  int main(){
3      float peso, altura, imc;
4
5      printf("Digite peso e altura");
6      scanf("%f %f", &peso, &altura);
7
8      imc = peso / (altura*altura);
9
10     printf("Seu IMC eh de: %.2f.\n", imc);
11     if (imc < 18.5)
12         printf("abaixo do peso");
13     else if (imc < 25)
14         printf("normal");
15     else if (imc < 30)
16         printf("acima do peso");
17     else
18         printf("30 ou mais");
19
20     return 0;
21 }
```

além disso, o encadeamento  
também permite omitir  
algumas comparações.

Por exemplo, aqui não é preciso  
comparar que  $imc \geq 18.5$   
(se chegou aqui, é pq a comparação  
da linha 11 é falsa, concorda?)

## *Dica*

Ao analisar uma sequência de `ifs`, faça a seguinte pergunta: “mais que uma condição pode ser verdadeira em uma dada execução do programa”?

- ① Se a resposta for não, o programa pode parar de testar assim que achar a condição adequada. Portanto, encadear as condições usando `else if` é uma boa opção.
- ② Se a resposta for sim, a sequência de `ifs` é necessária (dado que mais de uma condição pode ser verdadeira, é preciso testar todas).

## Checkpoint 1: apenas if ou else if?

A sequência de ifs abaixo seria mais eficiente (em termos de menos comparações) se as condições fossem encadeadas com else if?

```
1 if (salario <= 1500)
2     taxa = 5;
3 if (salario > 1500 && salario <= 3000)
4     taxa = 10;
5 if (salario > 3000 && salario <= 5000)
6     taxa = 20;
```

Observe que apenas uma condição pode ser verdadeira em uma dada execução do programa. Portanto, **SIM**, o programa fará menos comparações se as condições forem encadeadas com o else if.



## Checkpoint 1: apenas if ou else if?

A sequência de ifs abaixo seria mais eficiente (em termos de menos comparações) se as condições fossem encadeadas com else if?

```
1 if (salario <= 1500)
2     taxa = 5;
3 if (salario > 1500 && salario <= 3000)
4     taxa = 10;
5 if (salario > 3000 && salario <= 5000)
6     taxa = 20;
```

Observe que apenas uma condição pode ser verdadeira em uma dada execução do programa. Portanto, **SIM**, o programa fará menos comparações se as condições forem encadeadas com o else if.

## Checkpoint 2

Analise o encadeamento abaixo: existe alguma comparação desnecessária?

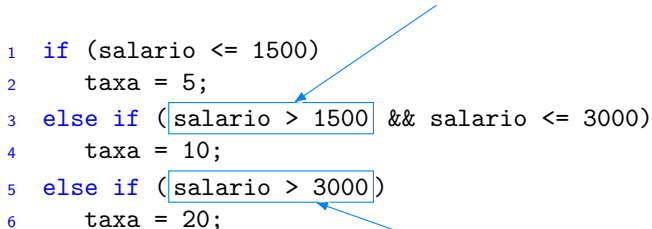
```
1  if (salario <= 1500)
2      taxa = 5;
3  else if (salario > 1500 && salario <= 3000)
4      taxa = 10;
5  else if (salario > 3000)
6      taxa = 20;
```

## Checkpoint 2

Analise o encadeamento abaixo: existe alguma comparação desnecessária?

observe que este teste (`salario > 1500`) é desnecessário, dado que essa condição só será avaliada se a anterior for falsa, ou seja, para esta condição ser analisada, o conteúdo da variável `salario` é obrigatoriamente maior que 1500

```
1 if (salario <= 1500)
2     taxa = 5;
3 else if (salario > 1500 && salario <= 3000)
4     taxa = 10;
5 else if (salario > 3000)
6     taxa = 20;
```




a mesma lógica vale para esta comparação.  
Portanto, podemos usar apenas `else`,  
sem testar nenhuma condição específica!

## Checkpoint 3

É também possível omitir os testes anteriores se não utilizarmos condições encadeadas?

ao omitir o teste `salario > 1500`  
teríamos um erro de execução, pois um salário  
menor que 1500 é também menor que 3000.  
Portanto, para um salário de R\$ 800, a taxa  
atribuída seria de 10, e não de 5 como esperado

```
1 if (salario <= 1500)
2     taxa = 5;
3 if (salario <= 3000)
4     taxa = 10;
5 if (salario > 3000)
6     taxa = 20;
```



## O que usar - parte 02

### ifs, ifs aninhados ou else if?

Considere o seguinte exemplo: escrever um programa que verifica se um número inteiro é

- par e menor que 100
- par e maior ou igual a 100
- ímpar e menor que 100
- ímpar e maior ou igual a 100

# Solução 01: apenas ifs

Est solução apresenta a limitação já discutida aqui: dependendo do valor de num, pode realizar muitas comparações desnecessárias. Para num=10 (par e menor que 100), por exemplo, não seria necessário continuar testando as condições das linhas 7, 10 e 13.

```
1  int main(){
2      int num = _____;
3
4      if(num % 2 == 0 && num<100)
5          printf("Par e menor que 100");
6
7      if(num % 2 == 0 && num>=100)
8          printf("Par e maior ou igual a 100");
9
10     if(num % 2 != 0 && num<100)
11         printf("Ímpar e menor que 100");
12
13     if(num % 2 != 0 && num>=100)
14         printf("Ímpar e maior que 100");
15
16     return 0;
17 }
```

## Solução 2A: else if

Ao encadear as condições, evitamos comparações desnecessárias. Agora, para num=10, apenas a comparação da linha 5 é realizada.

```
1  int main(){
2
3      int num = _____;
4
5      if(num % 2 == 0 && num<100)
6          printf("Par e menor que 100");
7      else if(num % 2 == 0 && num>=100)
8          printf("Par e maior ou igual a 100");
9      else if(num % 2 != 0 && num<100)
10         printf("Ímpar e menor que 100");
11     else if(num % 2 != 0 && num>=100)
12         printf("Ímpar e maior que 100");
13
14     return 0;
15 }
```

## Solução 2B: else if

Analisando a solução anterior, é possível identificar que algumas comparações são redundantes.

```
1  int main(){
2
3      int num = _____;
4
5      if(num % 2 == 0 && num<100)
6          printf("Par e menor que 100");
7      else if(num % 2 == 0 && num>=100)
8          printf("Par e maior ou igual a 100");
9      else if(num<100) // se chegou aqui, é ímpar
10         printf("Ímpar e menor que 100");
11     else //única possibilidade restante
12         printf("Ímpar e maior que 100");
13
14     return 0;
15 }
```



## Solução 03A: ifs aninhados

Observe como essa solução realiza comparações de forma mais eficiente. O bloco if-else mais externo testa se o valor é par ou ímpar, enquanto a estrutura aninhada faz a comparação com 100.

```
1  int main(){
2      int num = _____;
3
4      if(num % 2 == 0){
5          if(num < 100)
6              printf("Par e menor que 100");
7          else
8              printf("Par e maior ou igual a 100");
9      }else{
10         if (num < 100)
11             printf("Ímpar e menor que 100");
12         else
13             printf("Ímpar e maior ou igual a 100");
14     }
15     return 0;
16 }
```

## Solução 03B: ifs aninhados

Variação do programa anterior (agora o bloco if-else mais externo testa se o valor é menor ou maior/igual que 100).

```
1  int main(){
2      int num = _____;
3
4      if(num < 100){
5          if(num % 2 == 0)
6              printf("Par e menor que 100");
7          else
8              printf("Ímpar e menor que 100");
9      }else{
10         if (num % 2 == 0)
11             printf("Par e maior ou igual a 100");
12         else
13             printf("Ímpar e maior ou igual a 100");
14     }
15     return 0;
16 }
```

# Faça a lista de exercícios

Antes de prosseguir para o próximo tópico, faça os exercícios disponibilizados. Preste atenção na lógica utilizada, analisando se a sua solução está correta e se é eficiente. Observe que os programas estão ficando mais complexos (usar uma lógica clara e robusta é fundamental).



Praticar é fundamental para o aprendizado!