

# Fundamentos de Programação

## Estruturas condicionais: erros comuns

Dainf - UTFPR

Profa. Leyza Baldo Dorini  
Prof. Bogdan Tomoyuki Nassu

# Usos “equivocados” das estruturas condicionais

Embora o princípio das estruturas condicionais seja muito simples, algumas vezes são usadas de forma inconsistente ou mesmo errada. Vamos analisar alguns exemplos.



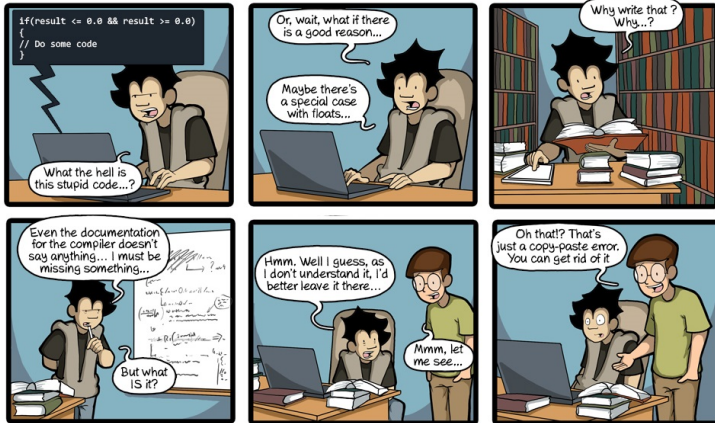
# Erro muito comum (parte 1): trocar == por =

Cuidado para não trocar o operador relacional igual (==) pelo operador de atribuição =. Ao fazer `if (var=35)`, por exemplo, estamos testando se a atribuição do valor 35 para `var` pode ser realizada (o que retorna `true`).



# Erro muito comum (parte 2): condições inconsistentes

Analise as condições: muitas vezes erros simples podem torná-las sempre true ou sempre false!



CommitStrip.com

# Erro muito comum (parte 3): condições erradas

Analise as condições (mas não como na tirinha!!!). Aprenda a analisar a lógica implementada para identificar erros!

GOOD CODERS...



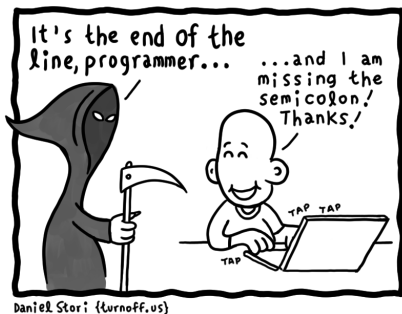
... KNOW WHAT THEY'RE DOING

# Erro muito comum (parte 3): colocar ; depois da condição

Não deve-se usar ; na estrutura condicional. Ao fazer

```
1 if (a > 5);  
2     printf("OI");
```

a mensagem OI será impressa sempre, e não apenas quando  $a > 5$ . Isso pq o símbolo de ; atua como um finalizador (é como se a instrução fosse: se  $a > 5$ , não faça nada).



# Uso das chaves (parte 1)

**Cuidado! Em C, não basta a indentação! É preciso delimitar com chaves os blocos de comandos com mais de uma instrução.**

```
1  if(a>=b)
2      printf("Mensagem A ");
3  if(a<b)
4      printf("Mensagem B ");
5      printf("Mensagem C ");
6
7  printf("Mensagem D ");
```

apesar de estar indentado, este comando **não pertence** ao bloco de comandos do if da linha 3 (como não estamos delimitando com chaves, apenas o primeiro comando será considerado). Portanto, será impresso sempre, assim como o printf da linha 7

Por exemplo:

- ❶ Para a=5, b=1, imprime Mensagem A Mensagem C Mensagem D
- ❷ Para a=1, b=2, imprime Mensagem B Mensagem C Mensagem D

# Uso das chaves (parte 1)

Para que o comando da linha 5 seja executado somente quando a condição da linha 3 for verdadeira, precisamos delimitar o bloco de comandos com chaves.

```
1  if(a>=b)
2      printf("Mensagem A ");
3  if(a<b){
4      printf("Mensagem B ");
5      printf("Mensagem C ");
6  }
7  printf("Mensagem D ");
```

desta forma, o bloco de comandos da estrutura condicional da linha 3 é composto por duas instruções.

Agora,

- ① Para a=5, b=1, imprime Mensagem A Mensagem D
- ② Para a=1, b=2, imprime Mensagem B Mensagem C Mensagem D



## Uso das chaves (parte 2)

O else da linha 5 está vinculado a qual if? Ao da linha 1 ou ao da linha 2?

```
1  if(num % 2 == 1)
2      if(num > 50)
3          printf(Ímpar > que 50. ");
4
5  else
6      printf("Par ");
```

pela indentação, parece que ele está vinculado ao if da linha 1, ou seja, para num=10 (por exemplo), seria impressa a mensagem "Par". Entretanto, não é o que acontece.

para este trecho de código, a mensagem "Par" seria impressa para ímpares menores ou iguais a 50 (ou seja, sem as chaves para delimitar o escopo, o else está vinculado ao if mais próximo, no caso o da linha 2)


E agora? Como corrigir este problema?

## Uso das chaves (parte 2)

Para corrigir este erro (de execução), precisamos colocar as chaves para delimitar corretamente o bloco de comandos do `if`.

```
1  if(num % 2 == 1){  
2      if(num > 50)  
3          printf(Impar > que 50. ");  
4  }  
5  else  
6      printf("Par ");
```

agora este `else` está  
corretamente vinculado ao  
`if` da linha 1.



portanto, lembre-se: em C, não basta  
a indentação: pode ser necessário usar chaves  
para delimitar corretamente o bloco de  
comandos das estruturas condicionais

## Uso das chaves (parte 3)

Mas cuidado: o “excesso” de chaves também pode ser um problema! O trecho de código abaixo resulta no seguinte erro de compilação: error: expected '}' before 'else'

```
1 if(num % 2 == 1){  
2     printf("Mensagem A. ");  
3     else if(num > 50)  
4         printf("Mensagem B. ");  
5 }  
6 else  
7     printf("Mensagem C. ");
```

o erro está aqui: como este `else if` pertence ao bloco de comandos do `if` da linha 1 (veja as chaves), ele **não está** encadeado, apesar de a indentação levar a pensar que sim. Portanto, é como se tentássemos iniciar uma estrutura condicional com `else if`, o que não faz sentido (erro de compilação).

lembre-se: usar chaves corretamente é essencial para delimitar o bloco de comandos das estruturas condicionais

## Checkpoint - parte 1

Para o trecho de código abaixo, qual o efeito de tirarmos as chaves das linhas 2 e 10? Qual a saída para idade=25?

```
1  if (idade >= 35)
2  //{
3      printf("categoria master ");
4      if (idade < 42)
5          printf("35+");
6      else if (idade < 50)
7          printf("42+");
8      else
9          printf("50+");
10 //}
```

## Checkpoint - parte 1

Para o trecho de código abaixo, qual o efeito de tirarmos as chaves das linhas 2 e 10? Qual a saída para idade=25?

```
1  if (idade >= 35)
2  //{
3  printf("categoria master ");
4  if (idade < 42)
5      printf("35+");
6  else if (idade < 50)
7      printf("42+");
8  else
9      printf("50+");
10 //}
```

ao tirar as chaves, apenas o primeiro comando seria vinculado ao bloco de instruções do if da linha 1

## Checkpoint - parte 1

Para o trecho de código abaixo, qual o efeito de tirarmos as chaves das linhas 2 e 10? Qual a saída para idade=25?

```
1  if (idade >= 35)
2  //{
3      printf("categoria master ");
4      if (idade < 42)
5          printf("35+");
6      else if (idade < 50)
7          printf("42+");
8      else
9          printf("50+");
10 //}
```

ao tirar as chaves, apenas o primeiro comando seria vinculado ao bloco de instruções do if da linha 1

essa estrutura seria executada independentemente da condição do if da linha 1. Portanto, para idade=25 a mensagem 35+ seria impressa, gerando um erro de execução.

## Checkpoint - parte 2

Corrija o programa do slide "Uso de chaves (parte 3)"

```
1 if(num % 2 == 1){  
2     printf("Mensagem A. ");  
3     else if(num > 50)  
4         printf("Mensagem B. ");  
5 }  
6 else  
7     printf("Mensagem C. ");
```

o erro está aqui



## Checkpoint - parte 2

Corrija o programa do slide “Uso de chaves (parte 3)”

```
1  if(num % 2 == 1){  
2      printf("Mensagem A. ");  
3  }  
4  else if(num > 50)  
5      printf("Mensagem B. ");  
6  else  
7      printf("Mensagem C. ");
```

seria preciso fechar chaves **antes**  
do `else if`, como ilustrado acima.

Além disso, como o `if` da linha 1 tem apenas  
um comando, podemos inclusive omitir as chaves.



# Indentação!

Entretanto, nunca é demais lembrar da importância da indentação.



CommitStrip.com

## Cuidado com o else (parte 1)

O else executa um comando (ou bloco de comandos) *quando a condição do if for falsa*. Ele não faz teste algum! Isso pode causar erros de execução. Por exemplo:

```
1  if(n1>n2)
2      printf("Primeira nota é maior");
3  else
4      printf("Segunda nota é maior");
5
```

Note que, quando as notas são iguais ( $n1==n2$ ), o programa irá imprimir que a segunda nota é a maior. Em outras palavras, é responsabilidade do programador garantir que o bloco de comandos do else seja coerente.

## Cuidado com o else (parte 2)

Não custa lembrar: na ausência de chaves para delimitar blocos, o else sempre vai se referir ao último if! Considere o exemplo:

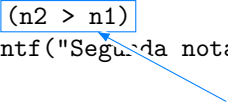
```
1  if(n1>n2)
2      printf("Primeira nota é maior");
3  if(n2>n1)
4      printf("Segunda nota é maior");
5  else
6      printf("As notas são iguais");
7
```

Para o caso  $n1 > n2$ , ocorre um erro de execução!! Imprime tanto “Primeira nota é maior” quanto “As notas são iguais”. Isso ocorre porque as estruturas não estão corretamente encadeadas (o else está vinculado apenas ao if da linha 3).

## Cuidado com o else (parte 3)

O else **não tem condição!**

```
1 if(n1>n2)
2     printf("Primeira nota é maior");
3 else (n2 > n1)
4     printf("Segunda nota é maior");
```



isso causa uma sucessão de erros de execução...  
A comparação `n2 > n1` vai ser interpretada  
como uma instrução. Além disso, como  
não delimitamos o bloco com chaves, `n2 > n1`  
será a única instrução executada (ou seja,  
a mensagem da linha 4 será impressa sempre)

## Cuidado com o else: exemplo de solução

Uma possível solução para o problema anterior é:

```
1  if(n1>n2)
2      printf("Primeira nota é maior");
3  else if(n2>n1)
4      printf("Segunda nota é maior");
5  else
6      printf("As notas são iguais");
7
```

Agora está correto porque o else está vinculado ao bloco de condições todo (e não apenas à segunda condição). Além disso, todas as comparações estão corretamente contempladas.

# Cuidado ao combinar expressões relacionais

Suponha que precisamos comparar se uma idade está entre 18 e 60. A seguinte comparação gera um erro de execução:

```
if(18 <= idade <= 60)
```

Neste caso as comparações são realizadas da esquerda para a direita: `18 < idade` será avaliado como `true` (retornando 1) ou `false` (0). Portanto, o que será comparado com o valor 60 será 1 ou 0, e não o conteúdo de `idade`, como esperado.

## Atenção!

Portanto, ao comparar um valor com outros dois (operadores relacionais), é preciso separar as comparações com operadores lógicos. Para exemplo anterior: `if(18 <= idade && idade <= 60)`

# Cuidado ao combinar expressões relacionais

Suponha que precisamos comparar se uma idade está entre 18 e 60. A seguinte comparação gera um erro de execução:

```
if(18 <= idade <= 60)
```

Neste caso as comparações são realizadas da esquerda para a direita: `18 < idade` será avaliado como `true` (retornando 1) ou `false` (0). Portanto, o que será comparado com o valor 60 será 1 ou 0, e não o conteúdo de `idade`, como esperado.

## Atenção!

Portanto, ao comparar um valor com outros dois (operadores relacionais), é preciso separar as comparações com operadores lógicos. Para exemplo anterior: `if(18 <= idade && idade <= 60)`

# Checkpoint

Corrija as seguintes comparações:

```
if(a > b > c)
```

```
if(5 <= a <= 10)
```

```
if(5.5 < taxa < 10.5)
```

Possíveis soluções:

```
if(a > b && b > c)
```

```
if(5 <= a && a <= 10) ou
```

```
if(a >= 5 && a <= 10)
```

```
if(5.5 < taxa && taxa < 10.5) ou
```

```
if(taxa > 5.5 && taxa < 10.5)
```



# Checkpoint

Corrija as seguintes comparações:

```
if(a > b > c)
```

```
if(5 <= a <= 10)
```

```
if(5.5 < taxa < 10.5)
```

Possíveis soluções:

```
if(a > b && b > c)
```

```
if(5 <= a && a <= 10) ou
```

```
if(a >= 5 && a <= 10)
```

```
if(5.5 < taxa && taxa < 10.5) ou
```

```
if(taxa > 5.5 && taxa < 10.5)
```

# Evite redundâncias!

```
1  #include<stdio.h>
2  int main(){
3
4      int saldo;
5
6      printf("Digite o saldo inicial: ");
7      scanf("%d", &saldo);
8
9      if (saldo < 50){
10         saldo = saldo + 100;
11         printf("Saldo: %d", saldo);
12     }
13     else if (saldo > 100){
14         saldo = saldo - 50;
15         printf("Saldo: %d", saldo);
16     }
17     else
18         printf("Saldo: %d", saldo);
19
20     return 0;
21 }
```

note que este printf  
aparece em todos os blocos.  
Portanto, como será executado  
de qualquer forma, pode ficar  
após a estrutura condicional

# Evite redundâncias!

Tudo que é comum às condições pode sair do bloco de comandos!

```
1 #include<stdio.h>
2 int main(){
3
4     int saldo;
5
6     printf("Digite o saldo inicial: ");
7     scanf("%d", &saldo);
8
9     if (saldo < 50){
10         saldo = saldo + 100;
11     }
12     else if (saldo > 100){
13         saldo = saldo - 50;
14     }
15
16     printf("Saldo: %d", saldo);
17
18     return 0;
19 }
```

ao retirar o printf do bloco, temos um programa mais objetivo. Além disso, caso a mensagem precise ser alterada, é preciso fazê-lo em apenas um local (e não em três)

## Algumas condições podem ser mais custosas

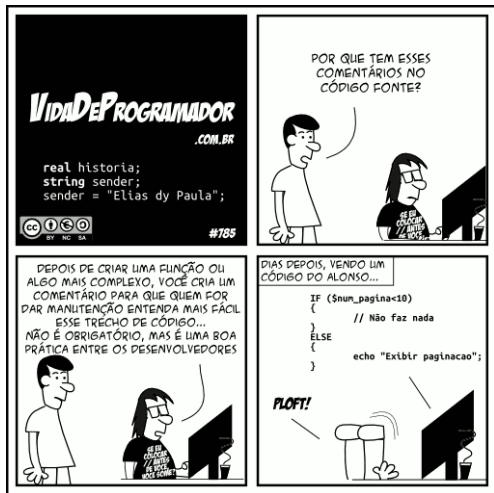
Suponha o seguinte exemplo: dados três lados de um triângulo, determinar se é equilátero, isóceles ou escaleno. Uma possível solução é:

```
1  if(testar se dois lados iguais e um diferente)
2      printf("Isoceles");
3  else if (testar se três lados diferentes)
4      printf("Escaleno");
5  else
6      printf("Equilatero");
```

Note que o teste mais difícil (para o isóceles) está no if, ao passo que o mais simples (equilátero) é contemplado no else. O programa seria simplificado se o caso isóceles fosse deixado para o else, concorda?

# Use corretamente a estrutura condicional (parte 1)

E claro... nem sempre é preciso colocar o else, basta um if...



# Use corretamente a estrutura condicional (parte 2)

... em outras, nem o próprio if é necessário.



# Use corretamente a estrutura condicional (parte 3)

Ao “tirar” o if, não se esqueça de fazer isso corretamente ;-)



# Para finalizar...

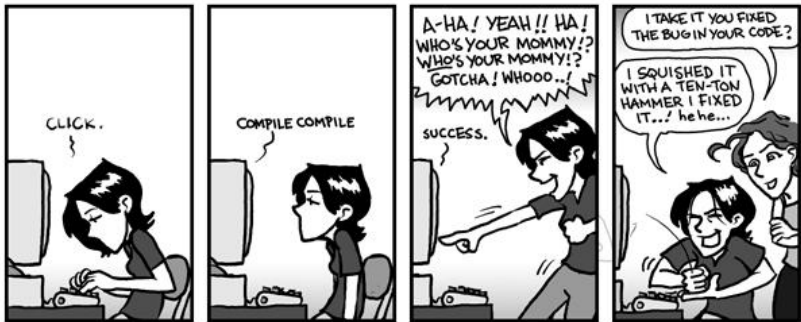
Lembre-se que, nesta etapa de aprendizado de uma linguagem de programação, os erros podem ser mais básicos do que você imagina... É preciso atenção ao programar!





# Faça os exercícios disponibilizados

Antes de prosseguir para o próximo assunto, faça a lista de exercícios disponibilizada! Praticar é fundamental para o aprendizado!



phd.stanford.edu/