# Aula 10 – Condicionais (parte 1)

Norton T. Roman & Luciano A. Digiampietri

# Testando os Parâmetros

- Não testamos o valor passado ao parâmetro
- Por se tratar de uma área, não poderia aceitar valores negativos

```
#include <stdio.h>

double valorM2 = 1500;

double valor(double area) {
    return(valorM2*area);
}

int main() {
    double preco;
    preco = valor(-20);
    printf("O valor da construção
                é %f\n", preco);
    return 0;
}
```

# Testando os Parâmetros

- Não testamos o valor passado ao parâmetro
- Por se tratar de uma área, não poderia aceitar valores negativos
- E como podemos testar?

```
#include <stdio.h>

double valorM2 = 1500;

double valor(double area) {
    return(valorM2*area);
}

int main() {
    double preco;
    preco = valor(-20);
    printf("O valor da construção
                é %f\n", preco);
    return 0;
}
```

# Testando os Parâmetros

- SE o parâmetro for positivo ENTÃO calcule a área
- SENÃO, retorne um valor indicando erro

# Testando os Parâmetros

- SE o parâmetro for positivo ENTÃO calcule a área
- SENÃO, retorne um valor indicando erro
  - Esse valor é algo que definimos como, por exemplo, -1

# Testando os Parâmetros

- SE o parâmetro for positivo ENTÃO calcule a área
- SENÃO, retorne um valor indicando erro
  - Esse valor é algo que definimos como, por exemplo, -1
- E como codificar isso?

# Testando os Parâmetros

```
#include <stdio.h>

double valorM2 = 1500;

double valor(double area) {
    if (area >= 0) {
        return(valorM2*area);
    } else {
        return(-1);
    }
}

int main() {
    double preco;
    preco = valor(-20);
    printf("O valor da construção é %f\n", preco);
    return 0;
}
```

# Testando os Parâmetros

```
#include <stdio.h>

double valorM2 = 1500;

double valor(double area) {
    if (area >= 0) {
        return(valorM2*area);
    } else {
        return(-1);
    }
}

int main() {
    double preco;
    preco = valor(-20);
    printf("O valor da construção é %f\n", preco);
    return 0;
}
```

- >=?

# Testando os Parâmetros

```
#include <stdio.h>

double valorM2 = 1500;

double valor(double area) {
    if (area >= 0) {
        return(valorM2*area);
    } else {
        return(-1);
    }
}

int main() {
    double preco;
    preco = valor(-20);
    printf("O valor da construção é %f\n", preco);
    return 0;
}
```

- $>=$?
- Operador relacional:

| *Matemática* | *Computação* |
|---|---|
| $>$ | $>$ |
| $<$ | $<$ |
| $=$ | $==$ |
| $\neq$ | $!=$ |
| $\leq$ | $<=$ |
| $\geq$ | $>=$ |

# Testando os Parâmetros

- O que o código no `if` diz?

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  } else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- O que o código no `if` diz?
- Se area $\geq 0$, então faça o cálculo e retorne

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  } else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- O que o código no `if` diz?
- Se `area` $\geq 0$, então faça o cálculo e retorne
- E o `else`?

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  } else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- O que o código no `if` diz?
- Se `area` $\geq 0$, então faça o cálculo e retorne
- E o `else`?
- Senão retorne -1

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  } else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- O código dentro do `if` é executado somente se a condição entre parênteses for verdadeira
- Se a condição for falsa, o código no `if` é ignorado

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  } else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- O código dentro do *else* é executado somente se a condição for falsa
- Se a condição for verdadeira, o código no *else* é ignorado

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  }
  else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- Será que tem como melhorar esse código?

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  }
  else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- Será que tem como melhorar esse código?
  - Precisa realmente do `else` nesse caso? Ou ele sempre é ignorado quando a condição for verdadeira?

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  }
  else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- Será que tem como melhorar esse código?
  - Precisa realmente do `else` nesse caso? Ou ele sempre é ignorado quando a condição for verdadeira?
- Se a condição for verdadeira, há o retorno

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  }
  else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- Será que tem como melhorar esse código?
  - Precisa realmente do `else` nesse caso? Ou ele sempre é ignorado quando a condição for verdadeira?
- Se a condição for verdadeira, há o retorno
  - Nada mais será executado, e o else é ignorado de qualquer forma

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  }
  else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- Note que, nesse caso, devido ao condicional, o compilador permite que haja código após o `return`
  - Não há como dizer de antemão se haverá o retorno

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  }
  else {
    return(-1);
  }
}
```

# Testando os Parâmetros

- Note que, nesse caso, devido ao condicional, o compilador permite que haja código após o `return`
  - Não há como dizer de antemão se haverá o retorno

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  }
  else {
    return(-1);
  }
}
```

- Então vamos reduzir o código um pouco

# Testando os Parâmetros

- Note que, nesse caso, devido ao condicional, o compilador permite que haja código após o `return`
  - Não há como dizer de antemão se haverá o retorno

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  }
  return(-1);
}
```

- Então vamos reduzir o código um pouco

# Testando os Parâmetros

- Note que, nesse caso, devido ao condicional, o compilador permite que haja código após o `return`
  - Não há como dizer de antemão se haverá o retorno

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  }
  return(-1);
}
```

- Então vamos reduzir o código um pouco
- Mas ainda dá pra deixar mais enxuto...

# Testando os Parâmetros

- Lembre que os {} denotam um bloco de comandos
- E que basta um ; para denotar o fim de um único comando

```
double valor(double area) {
  if (area >= 0) {
    return(valorM2*area);
  }
  return(-1);
}
```

# Testando os Parâmetros

Então, em vez de

```
double valor(double area){
  if (area >= 0) {
     return(valorM2*area);
  }
  return(-1);
}
```

Podemos fazer

```
double valor(double area){
  if (area >= 0)
     return(valorM2*area);
  return(-1);
}
```

# Testando os Parâmetros

- E como usamos isso no `main`?

# Testando os Parâmetros

- E como usamos isso no `main`?

```
int main() {
    double preco;
    preco = valor(-20);

    if (preco >= 0) printf("O valor da construção é %f\n", preco);
    else printf("Valor de área negativo\n");
    return 0;
}
```

# Testando os Parâmetros

- E como usamos isso no `main`?
- O condicional evita que usemos um resultado inválido do método

```
int main() {
    double preco;
    preco = valor(-20);

    if (preco >= 0) printf("O valor da construção é %f\n", preco);
    else printf("Valor de área negativo\n");
    return 0;
}
```

# Testando os Parâmetros

- E como usamos isso no `main`?

- O condicional evita que usemos um resultado inválido do método

- Evita inconsistências futuras

```
int main() {
    double preco;
    preco = valor(-20);

    if (preco >= 0) printf("O valor da construção é %f\n", preco);
    else printf("Valor de área negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Vejamos novamente o que está dentro do *if*

```
if (condição) {
   ...
}
else {
   ...
}
```

# Variáveis Booleanas

- Vejamos novamente o que está dentro do *if*
- O que significa condição?

```
if (condição) {
   ...
}
else {
   ...
}
```

# Variáveis Booleanas

- Vejamos novamente o que está dentro do *if*
- O que significa condição?
  - Expressão que resulta em verdadeiro ou falso

```
if (condição) {
   ...
}
else {
   ...
}
```

# Variáveis Booleanas

- Vejamos novamente o que está dentro do *if*
- O que significa condição?
  - Expressão que resulta em verdadeiro ou falso

```
if (condição) {
   ...
}
else {
   ...
}
```

- Usando esse conceito, haveria uma maneira alternativa (não necessariamente melhor) de escrever o main?

# Variáveis Booleanas

- Na linguagem C não exite um tipo de dados específico/exclusivo para variáveis boolenas (cujos valores são 0 ou 1, ou verdadeiro ou falso)

# Variáveis Booleanas

- Na linguagem C não exite um tipo de dados específico/exclusivo para variáveis boolenas (cujos valores são 0 ou 1, ou verdadeiro ou falso)
  - Mas variáveis do tipo inteiro (int) são utilizadas para isso

# Variáveis Booleanas

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    if (preco >= 0) valorOK = 1;
    else valorOK = 0;

    if (valorOK) printf("O valor da construção é %f\n", preco);
    else printf("Valor de área negativo\n");
    return 0;
}
```

# Variáveis Booleanas

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    if (preco >= 0) valorOK = 1;
    else valorOK = 0;

    if (valorOK) printf("O valor da construção é %f\n", preco);
    else printf("Valor de área negativo\n");
    return 0;
}
```

- Usando o tipo *int* para armazenar dois valores:
  - Verdadeiro (1)
  - Falso (0)

# Variáveis Booleanas

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    if (preco >= 0) valorOK = 1;
    else valorOK = 0;

    if (valorOK) printf("O valor da construção é %f\n", preco);
    else printf("Valor de área negativo\n");
    return 0;
}
```

- Usando o tipo *int* para armazenar dois valores:
  - Verdadeiro (1)
  - Falso (0)
- Valores lógicos

# Variáveis Booleanas

- Analisando o código ao lado, precisamos mesmo do `else`?

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    if (preco >= 0) valorOK = 1;
    else valorOK = 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Analisando o código ao lado, precisamos mesmo do `else`?
  - Se *preco* $\geq 0$, então `valorOK` recebe 1
  - Senão, `valorOK` recebe 0 ... mas ela já era 0

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    if (preco >= 0) valorOK = 1;
    else valorOK = 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Podemos então nos livrar dele sem problemas
  - Se *preco* $\geq 0$, então `valorOK` recebe 1
  - Senão, `valorOK` continua 0

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    if (preco >= 0) valorOK = 1;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Atribuímos os valores numéricos e, em especial, 0 (significando falso) e 1 (significando verdadeiro).
- Apenas isso?

# Variáveis Booleanas

- Atribuímos os valores numéricos e, em especial, 0 (significando falso) e 1 (significando verdadeiro).
- Apenas isso?
  - Por padrão, como valores só atribuímos esses dois.

# Variáveis Booleanas

- Atribuímos os valores numéricos e, em especial, 0 (significando falso) e 1 (significando verdadeiro).
- Apenas isso?
  - Por padrão, como valores só atribuímos esses dois.
- Mas também podemos atribuir resultados de expressões:
  - Ex: `int valorOK = 12 > 10;`
  - Nesse caso, *valorOK* conterá...

# Variáveis Booleanas

- Atribuímos os valores numéricos e, em especial, 0 (significando falso) e 1 (significando verdadeiro).
- Apenas isso?
  - Por padrão, como valores só atribuímos esses dois.
- Mas também podemos atribuir resultados de expressões:
  - Ex: `int valorOK = 12 > 10;`
  - Nesse caso, *valorOK* conterá... 1, pois é verdadeiro que $12 > 10$

# Variáveis Booleanas

- Em vista disso, poderíamos reescrever o momento de atribuição de valor de `valorOK`

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    if (preco >= 0) valorOK = 1;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Em vista disso, poderíamos reescrever o momento de atribuição de valor de `valorOK`

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Em vista disso, poderíamos reescrever o momento de atribuição de valor de `valorOK`
- Se *preco* $>= 0$, então `valorOK` conterá 1

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Em vista disso, poderíamos reescrever o momento de atribuição de valor de `valorOK`
- Se *preco* $>= 0$, então `valorOK` conterá 1
- Senão, `valorOK` conterá 0

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Considere agora o `if`
- O que acontece no condicional?

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Considere agora o `if`
- O que acontece no condicional?
  - A expressão dentro dos parênteses é testada

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Considere agora o `if`
- O que acontece no condicional?
  - A expressão dentro dos parênteses é testada
  - Se seu resultado for verdadeiro, o código no corpo do `if` é executado

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Considere agora o `if`
- O que acontece no condicional?
  - A expressão dentro dos parênteses é testada
  - Se seu resultado for verdadeiro, o código no corpo do `if` é executado
  - Se for falso, o código no corpo do `else` é executado

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Se não houver `else`, o programa continua normalmente

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Não apenas expressões, mas também variáveis booleanas podem estar no `if`
- E são analisadas do mesmo modo

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Se a variável contiver um verdadeiro, então o corpo do `if` será executado
- Se contiver um falso, será o corpo do `else` (se existir)

```
int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
        construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Para deixar o código mais claro podemos criar constantes chamadas de *false* e *true*

```
#include <stdio.h>



...

int main() {
    double preco;
    int valorOK = 0;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da construção é %f\n", preco);
    else printf("Valor de área negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Para deixar o código mais claro podemos criar constantes chamadas de *false* e *true*

```
#include <stdio.h>
#define false 0
#define true 1

...

int main() {
    double preco;
    int valorOK = false;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da construção é %f\n", preco);
    else printf("Valor de área negativo\n");
    return 0;
}
```

# Variáveis Booleanas - typedef

- Podemos criar um tipo booleano (*bool* ou *Boolean*)?

# Variáveis Booleanas - typedef

- Podemos criar um tipo booleano (*bool* ou *Boolean*)?
  - Não exatamente ...

# Variáveis Booleanas - typedef

- Podemos criar um tipo booleano (*bool* ou *Boolean*)?
  - Não exatamente ...
  - Mas podemos dar um nome adicional (*alias*) ao tipo *int*

# Variáveis Booleanas - typedef

- Podemos criar um tipo booleano (*bool* ou *Boolean*)?
  - Não exatamente ...
  - Mas podemos dar um nome adicional (*alias*) ao tipo *int*
  - Sintaxe: **typedef** <**tipo de dado/estrutura**> <**novo nome**>**;**

# Variáveis Booleanas - typedef

- Podemos criar um tipo booleano (*bool* ou *Boolean*)?
  - Não exatamente ...
  - Mas podemos dar um nome adicional (*alias*) ao tipo *int*
  - Sintaxe: **typedef <tipo de dado/estrutura> <novo nome>;**
  - Exemplo: **typedef int bool;**

# Variáveis Booleanas

- Daremos um novo novo nome ao tipo *int* para deixar o código mais claro.

```
#include <stdio.h>
#define false 0
#define true 1


...

int main() {
    double preco;
    int valorOK = false;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da construção é %f\n", preco);
    else printf("Valor de área negativo\n");
    return 0;
}
```

# Variáveis Booleanas

- Daremos um novo novo nome ao tipo *int* para deixar o código mais claro.

```
#include <stdio.h>
#define false 0
#define true 1

typedef int bool;
...

int main() {
    double preco;
    bool valorOK = false;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da construção é %f\n", preco);
    else printf("Valor de área negativo\n");
    return 0;
}
```

# Visão Geral do Código

```
#include <stdio.h>
#include <math.h>
#define false 0
#define true 1

typedef int bool;
int valorM2 = 1500;

void areaCasa(float lateral,
                  float cquarto){
    float areaq;
    float areas;
    float areat;
    printf("Programa para cálculo
                da área da casa\n");
    areas = lateral*lateral;
    printf("A área da sala é %f\n", areas);
    areaq = cquarto*(lateral/2);
    printf("A área do quarto é %f\n", areaq);
    printf("A área do banheiro é %f\n", areaq);
    areat = areas + 2*areaq;
    printf("A área total é %f\n", areat);
}
```

```
double areaPiscina(double raio){
    return M_PI * pow(raio,2);
}

double valor(double area) {
    if (area >= 0) {
        return(valorM2*area);
    }
    return(-1);
}

int main() {
    double preco;
    bool valorOK = false;

    preco = valor(-20);
    valorOK = preco >= 0;

    if (valorOK) printf("O valor da
          construção é %f\n", preco);
    else printf("Valor de área
                        negativo\n");
    return 0;
}
```

# Aula 10 – Condicionais (parte 1)

Norton T. Roman & Luciano A. Digiampietri