

Aula 12 – Operadores Lógicos

Norton T. Roman & Luciano A. Digiampietri

Operadores Lógicos

- Considere a função para cálculo da área da casa

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0) printf("Erro: lateral
                        da sala < 0\n");
    else {
        if (cquarto<0) printf("Erro: lateral
                            do quarto < 0\n");
        else {
            printf("Programa para cálculo ...\n");
            areas = lateral*lateral;
            printf("A área... é %f\n", areas);
            areaq = cquarto*(lateral/2);
            printf("A área... é %f\n", areaq);
            printf("A área... é %f\n", areaq);
            areat = areas + 2*areaq;
            printf("A área total é %f\n", areat);
        }
    }
}
```

Operadores Lógicos

- Considere a função para cálculo da área da casa
- São necessários mesmo 2 IFs para isso?
- O que eles significam?

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0) printf("Erro: lateral
                        da sala < 0\n");
    else {
        if (cquarto<0) printf("Erro: lateral
                            do quarto < 0\n");
        else {
            printf("Programa para cálculo ...\n");
            areas = lateral*lateral;
            printf("A área... é %f\n", areas);
            areaq = cquarto*(lateral/2);
            printf("A área... é %f\n", areaq);
            printf("A área... é %f\n", areaq);
            areat = areas + 2*areaq;
            printf("A área total é %f\n", areat);
        }
    }
}
```

Operadores Lógicos

- Que o método acusará erro quando *lateral* < 0 **ou** *cquarto* < 0

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0) printf("Erro: lateral
                        da sala < 0\n");
    else {
        if (cquarto<0) printf("Erro: lateral
                            do quarto < 0\n");
        else {
            printf("Programa para cálculo ...\n");
            areas = lateral*lateral;
            printf("A área... é %f\n", areas);
            areaq = cquarto*(lateral/2);
            printf("A área... é %f\n", areaq);
            printf("A área... é %f\n", areaq);
            areat = areas + 2*areaq;
            printf("A área total é %f\n", areat);
        }
    }
}
```

Operadores Lógicos

- Que o método acusará erro quando *lateral* < 0 **ou** *cquarto* < 0
- Precisamos de um meio de representar isso

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0) printf("Erro: lateral
                        da sala < 0\n");
    else {
        if (cquarto<0) printf("Erro: lateral
                            do quarto < 0\n");
        else {
            printf("Programa para cálculo ...\n");
            areas = lateral*lateral;
            printf("A área... é %f\n", areas);
            areaq = cquarto*(lateral/2);
            printf("A área... é %f\n", areaq);
            printf("A área... é %f\n", areaq);
            areat = areas + 2*areaq;
            printf("A área total é %f\n", areat);
        }
    }
}
```

Operadores Lógicos

- Que o método acusará erro quando *lateral* < 0 **ou** *cquarto* < 0
- Precisamos de um meio de representar isso
 - **Operador lógico** or

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0) printf("Erro: lateral
                        da sala < 0\n");
    else {
        if (cquarto<0) printf("Erro: lateral
                            do quarto < 0\n");
        else {
            printf("Programa para cálculo ...\n");
            areas = lateral*lateral;
            printf("A área... é %f\n", areas);
            areaq = cquarto*(lateral/2);
            printf("A área... é %f\n", areaq);
            printf("A área... é %f\n", areaq);
            areat = areas + 2*areaq;
            printf("A área total é %f\n", areat);
        }
    }
}
```

Operadores Lógicos

- Em C, **ou** é representado por **||**

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0 || cquarto<0) printf("Erro:
        parâmetro < 0\n");
    else {
        printf("Programa para cálculo ...\n");
        areas = lateral*lateral;
        printf("A área... é %f\n", areas);
        areaq = cquarto*(lateral/2);
        printf("A área... é %f\n", areaq);
        printf("A área... é %f\n", areaq);
        areat = areas + 2*areaq;
        printf("A área total é %f\n", areat);
    }
}
```

Operadores Lógicos

- Em C, **ou** é representado por `||`
- E como o compilador sabe que deve fazer o `<` antes?

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0 || cquarto<0) printf("Erro:
        parâmetro < 0\n");
    else {
        printf("Programa para cálculo ...\n");
        areas = lateral*lateral;
        printf("A área... é %f\n", areas);
        areaq = cquarto*(lateral/2);
        printf("A área... é %f\n", areaq);
        printf("A área... é %f\n", areaq);
        areat = areas + 2*areaq;
        printf("A área total é %f\n", areat);
    }
}
```


Operadores Lógicos

- Em C, **ou** é representado por `||`
- E como o compilador sabe que deve fazer o `<` antes?
- Precedência: operadores relacionais têm precedência sobre operadores lógicos

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0 || cquarto<0) printf("Erro:
        parâmetro < 0\n");
    else {
        printf("Programa para cálculo ...\n");
        areas = lateral*lateral;
        printf("A área... é %f\n", areas);
        areaq = cquarto*(lateral/2);
        printf("A área... é %f\n", areaq);
        printf("A área... é %f\n", areaq);
        areat = areas + 2*areaq;
        printf("A área total é %f\n", areat);
    }
}
```

Operadores Lógicos

- Voltando ao código anterior...
- Haveria alternativa a dizer que o método acusará erro quando *lateral < 0* ou *cquarto < 0*?

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0) printf("Erro: lateral
                        da sala < 0\n");
    else {
        if (cquarto<0) printf("Erro: lateral
                            do quarto < 0\n");
        else {
            printf("Programa para cálculo ...\n");
            areas = lateral*lateral;
            printf("A área... é %f\n", areas);
            areaq = cquarto*(lateral/2);
            printf("A área... é %f\n", areaq);
            printf("A área... é %f\n", areaq);
            areat = areas + 2*areaq;
            printf("A área total é %f\n", areat);
        }
    }
}
```

Operadores Lógicos

- O método irá executar o código somente se $lateral \geq 0$ e $cquarto \geq 0$

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0) printf("Erro: lateral
                        da sala < 0\n");
    else {
        if (cquarto<0) printf("Erro: lateral
                            do quarto < 0\n");
        else {
            printf("Programa para cálculo ...\n");
            areas = lateral*lateral;
            printf("A área... é %f\n", areas);
            areaq = cquarto*(lateral/2);
            printf("A área... é %f\n", areaq);
            printf("A área... é %f\n", areaq);
            areat = areas + 2*areaq;
            printf("A área total é %f\n", areat);
        }
    }
}
```

Operadores Lógicos

- O método irá executar o código somente se $lateral \geq 0$ e $cquarto \geq 0$
- Operador lógico *and*

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral<0) printf("Erro: lateral
                        da sala < 0\n");
    else {
        if (cquarto<0) printf("Erro: lateral
                            do quarto < 0\n");
        else {
            printf("Programa para cálculo ...\n");
            areas = lateral*lateral;
            printf("A área... é %f\n", areas);
            areaq = cquarto*(lateral/2);
            printf("A área... é %f\n", areaq);
            printf("A área... é %f\n", areaq);
            areat = areas + 2*areaq;
            printf("A área total é %f\n", areat);
        }
    }
}
```

Operadores Lógicos

- É como representamos um e em C ?

Operadores Lógicos

- É como representamos um **e** em C?
- **&&**

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral>=0 && cquarto>=0) {
        printf("Programa para cálculo ...\n");
        areas = lateral*lateral;
        printf("A área... é %f\n", areas);
        areaq = cquarto*(lateral/2);
        printf("A área... é %f\n", areaq);
        printf("A área... é %f\n", areaq);
        areat = areas + 2*areaq;
        printf("A área total é %f\n", areat);
    } else printf("Erro: parâmetro < 0\n");
}
```

Operadores Lógicos

- Teríamos ainda outra alternativa?

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral>=0 && cquarto>=0) {
        printf("Programa para cálculo ...\n");
        areas = lateral*lateral;
        printf("A área... é %f\n", areas);
        areaq = cquarto*(lateral/2);
        printf("A área... é %f\n", areaq);
        printf("A área... é %f\n", areaq);
        areat = areas + 2*areaq;
        printf("A área total é %f\n", areat);
    } else printf("Erro: parâmetro < 0\n");
}
```

Operadores Lógicos

- Teríamos ainda outra alternativa?
- Se **não** for verdade que $lateral \geq 0$ e $cquarto \geq 0$, mostre o erro

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral>=0 && cquarto>=0) {
        printf("Programa para cálculo ...\n");
        areas = lateral*lateral;
        printf("A área... é %f\n", areas);
        areaq = cquarto*(lateral/2);
        printf("A área... é %f\n", areaq);
        printf("A área... é %f\n", areaq);
        areat = areas + 2*areaq;
        printf("A área total é %f\n", areat);
    } else printf("Erro: parâmetro < 0\n");
}
```


Operadores Lógicos

- Teríamos ainda outra alternativa?
- Se **não** for verdade que $lateral \geq 0$ e $cquarto \geq 0$, mostre o erro
- Operador lógico *not*

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (lateral>=0 && cquarto>=0) {
        printf("Programa para cálculo ...\n");
        areas = lateral*lateral;
        printf("A área... é %f\n", areas);
        areaq = cquarto*(lateral/2);
        printf("A área... é %f\n", areaq);
        printf("A área... é %f\n", areaq);
        areat = areas + 2*areaq;
        printf("A área total é %f\n", areat);
    } else printf("Erro: parâmetro < 0\n");
}
```

Operadores Lógicos

- E como representamos um **não** em C?

Operadores Lógicos

- E como representamos um **não** em C?
- !

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (!(lateral>=0 && cquarto>=0))
        printf("Erro: parâmetro < 0\n");
    else {
        printf("Programa para cálculo ...\n");
        areas = lateral*lateral;
        printf("A área... é %f\n", areas);
        areaq = cquarto*(lateral/2);
        printf("A área... é %f\n", areaq);
        printf("A área... é %f\n", areaq);
        areat = areas + 2*areaq;
        printf("A área total é %f\n", areat);
    }
}
```

Operadores Lógicos

- E como representamos um **não** em C?
- **!**
- Veja que negamos toda a expressão `lateral>=0 && cquarto>=0`

```
void areaCasa(float lateral, float cquarto){
    float areaq;
    float areas;
    float areat;
    if (!(lateral>=0 && cquarto>=0))
        printf("Erro: parâmetro < 0\n");
    else {
        printf("Programa para cálculo ...\n");
        areas = lateral*lateral;
        printf("A área... é %f\n", areas);
        areaq = cquarto*(lateral/2);
        printf("A área... é %f\n", areaq);
        printf("A área... é %f\n", areaq);
        areat = areas + 2*areaq;
        printf("A área total é %f\n", areat);
    }
}
```

Tabela Verdade

- **E**: *comando1* será executado?

		condição2
condição1		V
	V	

```
if (condição1)
    if (condição2)
        comando1;
```

Tabela Verdade

- **E**: *comando1* será executado?

		condição2
		V
condição1	V	V

```
if (condição1)
    if (condição2)
        comando1;
```

Tabela Verdade

- **E**: *comando1* será executado?

		condição2	
		V	F
condição1	V	V	

```
if (condição1)
    if (condição2)
        comando1;
```

Tabela Verdade

- **E**: *comando1* será executado?

		condição2	
		V	F
condição1	V	V	F
	F	V	F

```
if (condição1)
    if (condição2)
        comando1;
```


Tabela Verdade

- **E**: *comando1* será executado?

		condição2	
		V	F
condição1	V	V	F
	F		

```
if (condição1)
    if (condição2)
        comando1;
```

Tabela Verdade

- **E**: *comando1* será executado?

		condição2	
		V	F
condição1	V	V	F
	F	F	

```
if (condição1)
    if (condição2)
        comando1;
```

Tabela Verdade

- **E**: *comando1* será executado?

		condição2	
		V	F
condição1	V	V	F
	F	F	F

```
if (condição1)
    if (condição2)
        comando1;
```

Tabela Verdade

- **E**: *comando1* será executado?

		condição2	
		V	F
condição1	V	V	F
	F	F	F

```
if (condição1)
    if (condição2)
        comando1;
```

- Equivale ao operador aritmético $*$, com $V > 0$ e $F = 0$

Tabela Verdade

- **OU**: *comando1* será executado?

		condição2
	V	V
condição1	V	

```
if (condição1)
    comando1;
else
    if (condição2)
        comando1;
    else
        comando2;
```

Tabela Verdade

- **OU**: *comando1* será executado?

		condição2
		V
	V	V

```
if (condição1)
    comando1;
else
    if (condição2)
        comando1;
    else
        comando2;
```

Tabela Verdade

- **OU**: *comando1* será executado?

condição1	V	condição2	
		V	F
	V	V	

```
if (condição1)
    comando1;
else
    if (condição2)
        comando1;
    else
        comando2;
```

Tabela Verdade

- **OU**: *comando1* será executado?

		condição2	
		V	F
condição1	V	V	V

```
if (condição1)
    comando1;
else
    if (condição2)
        comando1;
    else
        comando2;
```


Tabela Verdade

- **OU**: *comando1* será executado?

		condição2	
		V	F
	V	V	V
	F		

```
if (condição1)
    comando1;
else
    if (condição2)
        comando1;
    else
        comando2;
```

Tabela Verdade

- **OU**: *comando1* será executado?

		condição2	
		V	F
condição1	V	V	V
	F	V	

```
if (condição1)
    comando1;
else
    if (condição2)
        comando1;
    else
        comando2;
```

Tabela Verdade

- **OU**: *comando1* será executado?

		condição2	
		V	F
condição1	V	V	V
	F	V	F

```
if (condição1)
    comando1;
else
    if (condição2)
        comando1;
    else
        comando2;
```

Tabela Verdade

- **OU**: *comando1* será executado?

		condição2	
		V	F
condição1	V	V	V
	F	V	F

```
if (condição1)
    comando1;
else
    if (condição2)
        comando1;
    else
        comando2;
```

- Equivale ao operador aritmético $+$, com $V > 0$ e $F = 0$

Tabela Verdade

- **NÃO**: *comando1* será executado?

condição1
V

```
if (!condição1)  
    comando1;
```

Tabela Verdade

- **NÃO**: *comando1* será executado?

condição1	V	F
-----------	---	---

```
if (!condição1)
    comando1;
```

Tabela Verdade

- **NÃO**: *comando1* será executado?

condição1	V		F
	F		

```
if (!condição1)
    comando1;
```

Tabela Verdade

- **NÃO**: *comando1* será executado?

condição1	V	F
	F	V

```
if (!condição1)
    comando1;
```


Operadores

- Operadores aritméticos, relacionais e lógicos podem ser misturados

Operadores

- Operadores aritméticos, relacionais e lógicos podem ser misturados
- Como isso será entendido?

```
int x = 3;  
if (((20-x)>5) &&  
    ((4/x) == 1) ||  
    ((16-x)>10))  
    printf("passou\n");
```

Operadores

- Precedência:

<i>maior</i>	– (unário) !
	* / %
	+ –
↓	== != > < >= <=
	&&
<i>menor</i>	=

```
int x = 3;
if (((20-x)>5) &&
    ((4/x) == 1) ||
    ((16-x)>10))
    printf("passou\n");
```

- E a resposta é ...

Operadores

- Precedência:

<i>maior</i>	– (unário) !
	* / %
	+ -
↓	== != > < >= <=
	&&
<i>menor</i>	=

```
int x = 3;
if (((20-x)>5) &&
    ((4/x) == 1) ||
    ((16-x)>10))
    printf("passou\n");
```

- E a resposta é “passou”

Operadores

<i>maior</i>	- (unário) !
	* / %
	+ -
↓	== != > < >= <=
	&&
<i>menor</i>	=

```
int x = ?;  
if (((20-x)>5) &&  
    ((4/x) == 1) ||  
    ((16-x)>10))  
    printf("passou\n");
```

- $x = 5$

Operadores

<i>maior</i>	- (unário) !
	* / %
	+ -
↓	== != > < >= <=
	&&
<i>menor</i>	=

```
int x = ?;  
if (((20-x)>5) &&  
    ((4/x) == 1) ||  
    ((16-x)>10))  
    printf("passou\n");
```

- $x = 5$
 - $V \ \&\& \ F \ || \ V \rightarrow$ "passou"

Operadores

<i>maior</i>	- (unário) !
	* / %
	+ -
↓	== != > < >= <=
	&&
<i>menor</i>	=

```
int x = ?;  
if (((20-x)>5) &&  
    ((4/x) == 1) ||  
    ((16-x)>10))  
    printf("passou\n");
```

- $x = 5$
 - $V \ \&\& \ F \ || \ V \rightarrow$ "passou"
- $x = 6$

Operadores

<i>maior</i>	- (unário) !
	* / %
	+ -
↓	== != > < >= <=
	&&
<i>menor</i>	=

```
int x = ?;  
if (((20-x)>5) &&  
    ((4/x) == 1) ||  
    ((16-x)>10))  
    printf("passou\n");
```

- $x = 5$
 - V && F || V → "passou"
- $x = 6$
 - V && F || F → \emptyset

Atenção!

- Use e abuse de parênteses

Atenção!

- Use e abuse de parênteses
- O que é mais fácil de entender?

Atenção!

- Use e abuse de parênteses
- O que é mais fácil de entender?
 - $20 - x > 5 \ \&\& \ 4 / x == 1 \ || \ 16 - x > 10$

Atenção!

- Use e abuse de parênteses
- O que é mais fácil de entender?
 - $20 - x > 5 \ \&\& \ 4 / x == 1 \ || \ 16 - x > 10$
 - $(((20 - x) > 5) \ \&\& \ ((4 / x) == 1)) \ || \ ((16 - x) > 10)$

Atenção!

- Use e abuse de parênteses
- O que é mais fácil de entender?
 - $20 - x > 5 \ \&\& \ 4 / x == 1 \ || \ 16 - x > 10$
 - $(((20 - x) > 5) \ \&\& \ ((4 / x) == 1)) \ || \ ((16 - x) > 10)$
- Ou então, removendo os espaços...
 - $20-x>5\&\&4/x==1\|\|16-x>10$

Atenção!

- Use e abuse de parênteses
- O que é mais fácil de entender?
 - $20 - x > 5 \ \&\& \ 4 / x == 1 \ || \ 16 - x > 10$
 - $(((20 - x) > 5) \ \&\& \ ((4 / x) == 1)) \ || \ ((16 - x) > 10)$
- Ou então, removendo os espaços...
 - $20-x>5\&\&4/x==1\|\|16-x>10$
 - $(((20-x)>5) \ \&\& \ ((4/x) == 1)) \ || \ ((16-x) > 10)$

O operador '?'

- A linguagem C possui um atalho para condicionais:

O operador '?'

- A linguagem C possui um atalho para condicionais:
- O operador '?'
 - `var = condição ? expressão 1 : expressão 2;`

O operador '?'

- A linguagem C possui um atalho para condicionais:
- O operador '?'
 - `var = condição ? expressão 1 : expressão 2;`
- Correspondendo a
 - `if (condição) var = expressão 1;`
`else var = expressão 2;`

O operador '?'

- Pode ser usado como substituição a esse tipo de condicional em qualquer parte do código:

```
double areaPiscina(double raio){  
    double resp;  
    if (raio >= 0)  
        resp = M_PI*pow(raio,2);  
    else resp = -1;  
    return(resp);  
}
```

```
double areaPiscina(double raio){  
    double resp;  
    resp = (raio >= 0) ?  
        M_PI*pow(raio,2) :  
        -1;  
    return(resp);  
}
```

O operador '?'

- Pode ser usado como substituição a esse tipo de condicional em qualquer parte do código:

```
double areaPiscina(double raio){  
    if (raio >= 0)  
        return M_PI*pow(raio,2);  
    else return(-1);  
}  
  
double areaPiscina(double raio){  
    return((raio >= 0) ?  
        M_PI*pow(raio,2) :  
        -1);  
}
```

Aula 12 – Operadores Lógicos

Norton T. Roman & Luciano A. Digiampietri