Aula 12 – Operadores Lógicos

Norton T. Roman & Luciano A. Digiampietri

 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");</pre>
      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);
```

- 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</pre>
                            do quarto < 0\n");</pre>
      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", areag);
         areat = areas + 2*areaq;
         printf("A área total é %f\n", areat);
```

 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</pre>
                            do quarto < 0\n");</pre>
      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);
```

- 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</pre>
                            do quarto < 0\n");</pre>
      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);
```

- 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</pre>
                            da sala < 0 \n"):
   else {
      if (cquarto<0) printf("Erro: lateral</pre>
                            do quarto < 0\n");</pre>
      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);
```

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");</pre>
   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*areag;
      printf("A área total é %f\n", areat);
```

- 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");</pre>
  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*areag;
      printf("A área total é %f\n", areat);
```

- 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");</pre>
  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*areag;
      printf("A área total é %f\n", areat);
```

- 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</pre>
                            do quarto < 0\n");</pre>
      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);
```

 O método irá executar o código somente se lateral > 0 e 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</pre>
                            do quarto < 0\n");</pre>
      else {
         printf("Programa para cálculo ...\n");
         areas = lateral*lateral:
         printf("A área... é %f\n", areas);
         areag = 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);
```

- O método irá executar o código somente se lateral ≥ 0 e cquarto ≥ 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</pre>
                            do quarto < 0\n");</pre>
      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);
```

E como representamos ume em C?

- E como representamos ume 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);
      areag = cguarto*(lateral/2);
      printf("A área... é %f\n", areaq);
      printf("A área... é %f\n", areaq);
      areat = areas + 2*areag;
      printf("A área total é %f\n", areat);
   } else printf("Erro: parâmetro < 0\n");</pre>
```

 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");</pre>
```

- Teríamos ainda outra alternativa?
- Se não for verdade que lateral ≥ 0 e cquarto ≥ 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");</pre>
```

- Teríamos ainda outra alternativa?
- Se não for verdade que lateral ≥ 0 e cquarto ≥ 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");</pre>
```

 E como representamos um não em C?

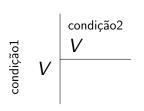
 E como representamos um não em C?

•

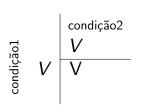
```
void areaCasa(float lateral, float cquarto){
   float areaq;
   float areas:
   float areat:
   if (!(lateral>=0 && cquarto>=0))
      printf("Erro: parâmetro < 0\n");</pre>
   else {
      printf("Programa para cálculo ...\n");
      areas = lateral*lateral:
      printf("A área... é %f\n", areas);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

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

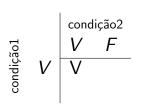
```
void areaCasa(float lateral, float cquarto){
   float areaq;
   float areas:
   float areat:
   if (!(lateral>=0 && cquarto>=0))
      printf("Erro: parâmetro < 0\n");</pre>
   else {
      printf("Programa para cálculo ...\n");
      areas = lateral*lateral:
      printf("A área... é %f\n", areas);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



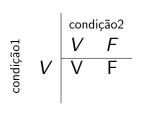
```
if (condição1)
    if (condição2)
        comando1;
```



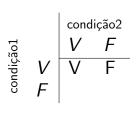
```
if (condição1)
   if (condição2)
      comando1;
```



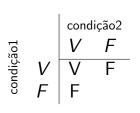
```
if (condição1)
   if (condição2)
        comando1;
```



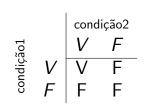
```
if (condição1)
   if (condição2)
      comando1;
```



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



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



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

• E: comando1 será executado?

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

```
condição2

V
```

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

```
condição2
V
V
```

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

```
condição2
V
V
V
```

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

```
condição2
V F
V V
```

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

```
condição2
V F

V V V

F
```

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

```
condição2
V F

vector V V V

vector V V V

vector V V V
```

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

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

• OU: comando1 será executado?

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

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



```
if (!condição1)
comando1;
```

```
condição1
```

```
if (!condição1)
comando1;
```

```
condição1
F V
```

```
if (!condição1)
comando1;
```

 Operadores aritméticos, relacionais e lógicos podem ser misturados

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

• Precedência:

• E a resposta é . . .

• Precedência:

maior
$$- (unário) !$$
 $* / \%$
 $+ + &&$
 $&&$
 $||$
menor $=$

• E a resposta é "passou"

```
maior - (unário)!

* / %

+ -

- >= != > < >= <=

&&

||

menor =
```

```
- (un\acute{a}rio) ! * / \% + - \\ == ! = > < > = <= \\ \&\& \\ || \\ menor = =
```

- x = 5
 - ullet V && F || V ightarrow "passou"

- x = 5
 - ullet V && F || V ightarrow "passou"
- x = 6

```
maior - (unário) ! * / % * / % * + - * >= <= && * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & * | & *
```

- x = 5
 - ullet V && F || V o "passou"
- x = 6
 - V && F || F → ∅

• Use e abuse de parênteses

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

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

- 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)

- 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

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

 - (((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)

• A linguagem C possui um atalho para condicionais:

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

- 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;

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

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

Aula 12 – Operadores Lógicos

Norton T. Roman & Luciano A. Digiampietri