C++
Básico ao Avançado
E se...

Heitor Rodrigues Savegnago

UFABC Rocket Design

2017.3

Fluxo

Fluxo

- Ordem de comandos

Fluxo

- Ordem de comandos
- Não é padrão a multiexecução

Fluxo

- Ordem de comandos
- Não é padrão a multiexecução
- Fluxos lineares nem sempre são úteis

Fluxo

- Ordem de comandos
- Não é padrão a multiexecução
- Fluxos lineares nem sempre são úteis
- Fluxograma

Fluxo

- Ordem de comandos
- Não é padrão a multiexecução
- Fluxos lineares nem sempre são úteis
- Fluxograma
- Algoritmo

Fluxo

- Ordem de comandos
- Não é padrão a multiexecução
- Fluxos lineares nem sempre são úteis
- Fluxograma
- Algoritmo

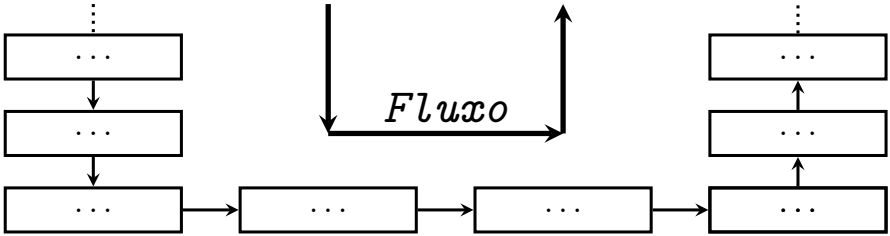

Figura: Fluxograma de fluxo simples

Ambiguidade

Ambiguidade

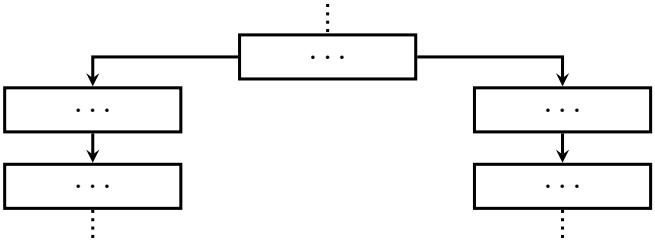

Figura: Fluxograma de fluxo ambíguo

Ambiguidade

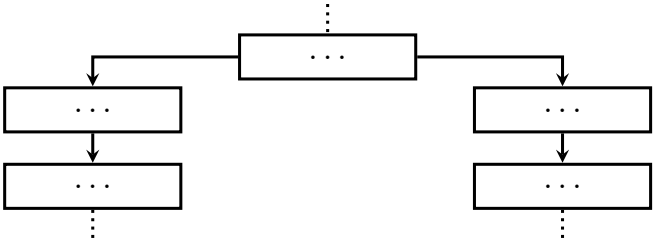

Figura: Fluxograma de fluxo ambíguo

- Qual escolher?

Ambiguidade

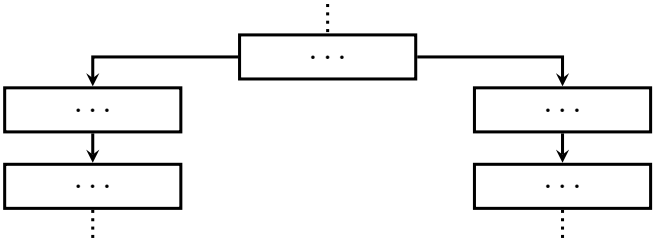

Figura: Fluxograma de fluxo ambíguo

- Qual escolher?
- *A luta do bem contra o mal*

Ambiguidade

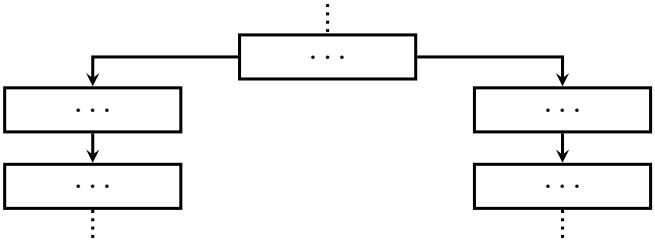

Figura: Fluxograma de fluxo ambíguo

- Qual escolher?
- *A luta do bem contra o mal*
- *Inverno ou verão*?

Ambiguidade

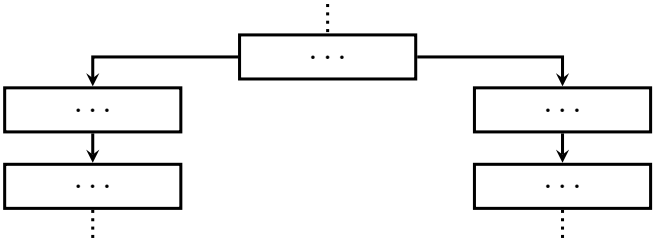

Figura: Fluxograma de fluxo ambíguo

- Qual escolher?
- *A luta do bem contra o mal*
- *Inverno ou verão*?
- *Dia ou noite*?

Ambiguidade

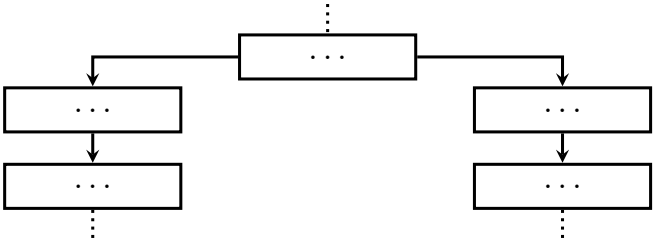

Figura: Fluxograma de fluxo ambíguo

- Qual escolher?
- *A luta do bem contra o mal*
- *Inverno ou verão*?
- *Dia ou noite*?
- *Biscoito ou bolacha*?

O *if*

O *if*

- Estrutura primordial:

O *if*

- Estrutura primordial: o `if`

O *if*

- Estrutura primordial: o `if`
- Recebe um *argumento* tipo `bool`

O *if*

- Estrutura primordial: o `if`
- Recebe um *argumento* tipo `bool`
- Não é um comando

O *if*

- Estrutura primordial: o `if`
- Recebe um *argumento* tipo `bool`
- Não é um comando
- É um controlador de fluxo

O *if*

- Estrutura primordial: o `if`
- Recebe um *argumento* tipo `bool`
- Não é um comando
- É um controlador de fluxo
- Pode ser construído com comando ou bloco de código

O *if*

- Estrutura primordial: o `if`
- Recebe um *argumento* tipo `bool`
- Não é um comando
- É um controlador de fluxo
- Pode ser construído com comando ou bloco de código

```
if(<cond>) <comand>;
```

O *if*

- Estrutura primordial: o `if`
- Recebe um *argumento* tipo `bool`
- Não é um comando
- É um controlador de fluxo
- Pode ser construído com comando ou bloco de código

```
if(<cond>) <comand>;
```

```
if(<cond>)
{
    <comand1>;
    //...
    <comandN>;
}
```

O *if* linear

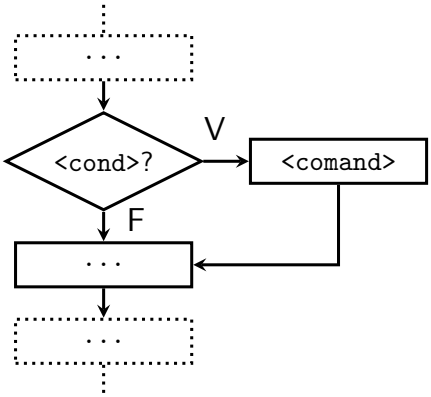

Figura: Fluxograma de *if* simples linear

O *if* blocular

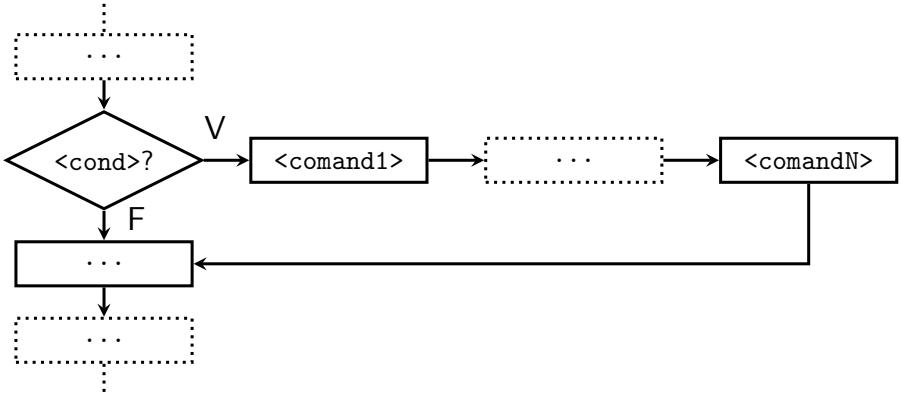

Figura: Fluxograma de *if* simples blocular

O *else*

O *else*

- Quando ser quer algo oposto ao `if`

O *else*

- Quando ser quer algo oposto ao `if`
- Só entra quando o `if` tem argumento falso

O *else*

- Quando ser quer algo oposto ao `if`
- Só entra quando o `if` tem argumento falso
- Só pode ser usado com `if`

O *else*

- Quando ser quer algo oposto ao `if`
- Só entra quando o `if` tem argumento falso
- Só pode ser usado com `if`
- Melhor que fazer comparação oposta

O *else*

- Quando ser quer algo oposto ao `if`
- Só entra quando o `if` tem argumento falso
- Só pode ser usado com `if`
- Melhor que fazer comparação oposta

```
if(<cond>) <comandA>;
else <comandB>;
```

O *else*

- Quando ser quer algo oposto ao `if`
- Só entra quando o `if` tem argumento falso
- Só pode ser usado com `if`
- Melhor que fazer comparação oposta

```
if(<cond>) <comandA>;
else <comandB>;
```

```
if(<cond>)
{
    <comandA1>;
    //...
    <comandAN>;
}
else
{
    <comandB1>;
    //...
    <comandBM>;
}
```

O *if else* linear

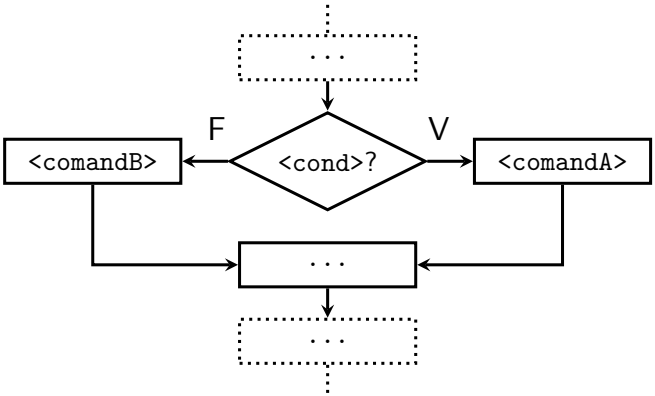

Figura: Fluxograma de *if else* simples linear

O *if else* blocular

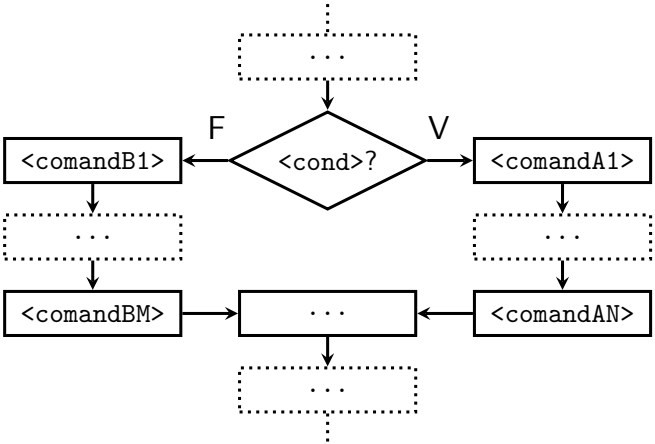

Figura: Fluxograma de *if else* simples blocular

Estruturas aninhadas

Estruturas aninhadas

- Estruturas em qualquer lugar

```
1  //...
2  float nota(8.7);
3  char conceito('\0');
4  //...
5       if (nota >= 9.5)   conceito = 'A';
6  else if (nota >= 8.0)   conceito = 'B';
7  else if (nota >= 7.0)   conceito = 'C';
8  else if (nota >= 4.5)   conceito = 'D';
9  else                    conceito = 'F';
10 //...
```

Estruturas aninhadas

- Estruturas em qualquer lugar

```
4  //...
5  if (nota >= 9.5)  conceito = 'A';
6  else
7  {
8    if (nota >= 8.0)  conceito = 'B';
9    else
10   {
11     if (nota >= 7.0)  conceito = 'C';
12     else
13     {
14       if (nota >= 4.5)  conceito = 'D';
15       else  conceito = 'F';
16     }
17   }
18 }
19 //...
```

Estruturas aninhadas

- Estruturas em qualquer lugar

```
4  //...
5       if (nota < 4.5)  conceito = 'F';
6  else if (nota < 7.0)  conceito = 'D';
7  else if (nota < 8.0)  conceito = 'C';
8  else if (nota < 9.5)  conceito = 'B';
9  else                  conceito = 'A';
10 //...
```

Estruturas aninhadas

- Estruturas em qualquer lugar

```
4  //...
5  if (10.0>=nota && nota >= 9.5) conceito = 'A';
6  if (9.5 > nota && nota >= 8.0) conceito = 'B';
7  if (8.0 > nota && nota >= 7.0) conceito = 'C';
8  if (7.0 > nota && nota >= 4.5) conceito = 'D';
9  if (4.5 > nota && nota >= 0.0) conceito = 'F';
10 //...
```

- Repetição de comandos

- Repetição de comandos
- Algoritmo de Euclides

```
int a, b, X, Y;
a=45; b=93;
//...
X = a>b?a:b;   //#1
Y = a<b?a:b;
a = X-Y;
b = Y;

X = a>b?a:b;   //#2
Y = a<b?a:b;
a = X-Y;
b = Y;

//Repete-se 18 vezes!

X = a>b?a:b;   //#17
Y = a<b?a:b;
a = X-Y;
b = Y;

X = a>b?a:b;   //#18
Y = a<b?a:b;
a = X-Y;       //a vale 3, o resultado
```

Ambiguidade

Ambiguidade

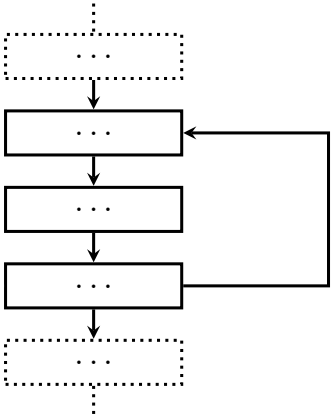

Figura: Fluxograma de fluxo repetitivo

Ambiguidade

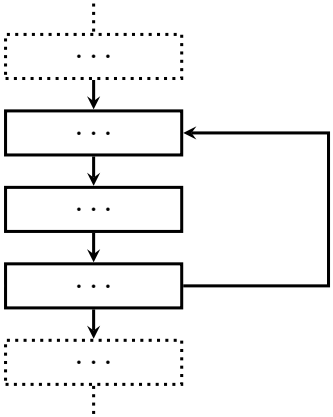

Figura: Fluxograma de fluxo repetitivo

- O que fazer?

Ambiguidade

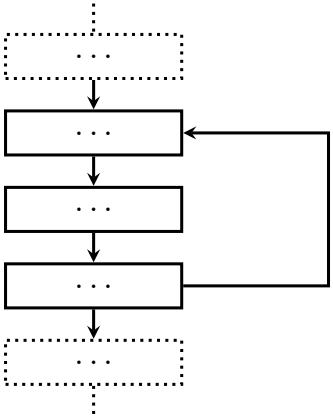

Figura: Fluxograma de fluxo repetitivo

- O que fazer?
- *Ficar na zona de conforto ou enfrentar a realidade*?

Ambiguidade

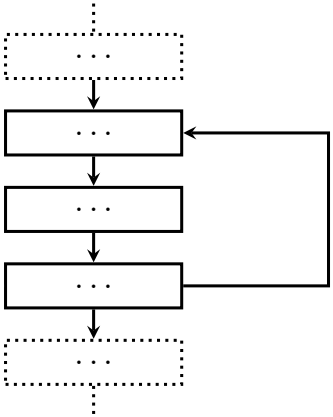

Figura: Fluxograma de fluxo repetitivo

- O que fazer?
- *Ficar na zona de conforto ou enfrentar a realidade*?
- *Repetição eterna ou faz uma vez a vai embora*?

Mas. . .

Mas...

- Não faz sentido repetir algo assim

Mas...

- Não faz sentido repetir algo assim
- O programador precisa saber o valor inicial antes de tudo

Mas...

- Não faz sentido repetir algo assim
- O programador precisa saber o valor inicial antes de tudo
- Por que então já não calcula antes?

Mas...

- Não faz sentido repetir algo assim
- O programador precisa saber o valor inicial antes de tudo
- Por que então já não calcula antes?
- `ctrl c`, `ctrl v`

Mas...

- Não faz sentido repetir algo assim
- O programador precisa saber o valor inicial antes de tudo
- Por que então já não calcula antes?
- `ctrl c`, `ctrl v`
- Alterar um valor comum a todos os casos

Mas...

- Não faz sentido repetir algo assim
- O programador precisa saber o valor inicial antes de tudo
- Por que então já não calcula antes?
- `ctrl c`, `ctrl v`
- Alterar um valor comum a todos os casos
- Um `if` já não serve?

O *while*

O *while*

- Sintaxe idêntica ao `if`

O *while*

- Sintaxe idêntica ao `if`
- Recebe argumento tipo `bool`

O *while*

- Sintaxe idêntica ao `if`
- Recebe argumento tipo `bool`
- Pergunta é realizada *antes* da repetição

O *while*

- Sintaxe idêntica ao `if`
- Recebe argumento tipo `bool`
- Pergunta é realizada *antes* da repetição
- Não é um comando

O *while*

- Sintaxe idêntica ao `if`
- Recebe argumento tipo `bool`
- Pergunta é realizada *antes* da repetição
- Não é um comando
- É um controlador de fluxo

O *while*

- Sintaxe idêntica ao `if`
- Recebe argumento tipo `bool`
- Pergunta é realizada *antes* da repetição
- Não é um comando
- É um controlador de fluxo
- Pode ser construído com comando ou bloco de código

O *while*

- Sintaxe idêntica ao `if`
- Recebe argumento tipo `bool`
- Pergunta é realizada *antes* da repetição
- Não é um comando
- É um controlador de fluxo
- Pode ser construído com comando ou bloco de código

```
while(<cond>) <comand>;
```

O *while*

- Sintaxe idêntica ao `if`
- Recebe argumento tipo `bool`
- Pergunta é realizada *antes* da repetição
- Não é um comando
- É um controlador de fluxo
- Pode ser construído com comando ou bloco de código

```
while(<cond>) <comand>;
```

```
while(<cond>)
{
  <comand1>;
  //...
  <comandN>;
}
```

O *while* linear

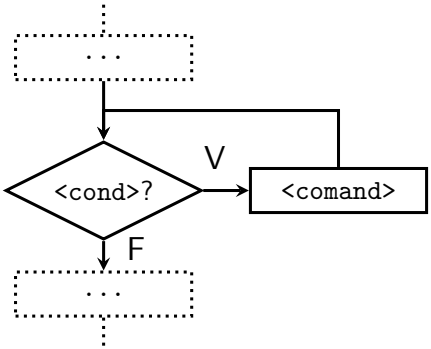

Figura: Fluxograma de *while* simples linear

O *while* blocular

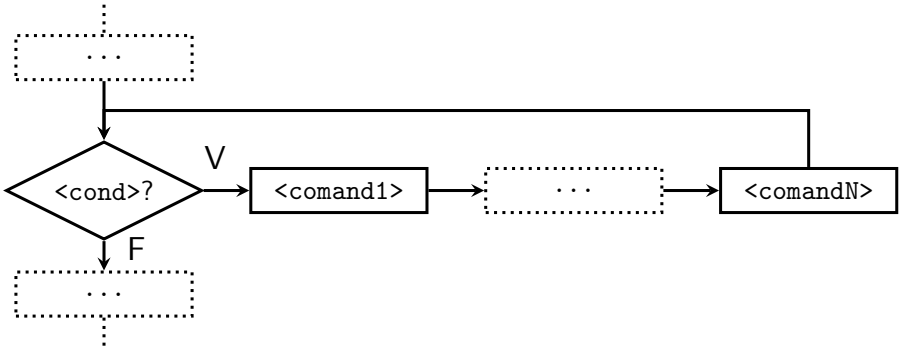

Figura: Fluxograma de *while* simples blocular

Algoritmo de Euclides

```
while(b!=0)
{
  X = a>b?a:b;
  Y = a<b?a:b;
  a = X-Y;
  b = Y;
}
```

O *do*

O *do*

- Sintaxe semelhanto ao `while` normal

O *do*

- Sintaxe semelhanto ao `while` normal
- Recebe argumento tipo `bool`

O *do*

- Sintaxe semelhanto ao `while` normal
- Recebe argumento tipo `bool`
- Pergunta é realizada *depois* da repetição

O *do*

- Sintaxe semelhanto ao `while` normal
- Recebe argumento tipo `bool`
- Pergunta é realizada *depois* da repetição
- Não é um comando

O *do*

- Sintaxe semelhanto ao `while` normal
- Recebe argumento tipo `bool`
- Pergunta é realizada *depois* da repetição
- Não é um comando
- É um controlador de fluxo

O *do*

- Sintaxe semelhanto ao `while` normal
- Recebe argumento tipo `bool`
- Pergunta é realizada *depois* da repetição
- Não é um comando
- É um controlador de fluxo
- Acaba com ponto-e-vírgula

O *do*

- Sintaxe semelhanto ao `while` normal
- Recebe argumento tipo `bool`
- Pergunta é realizada *depois* da repetição
- Não é um comando
- É um controlador de fluxo
- Acaba com ponto-e-vírgula
- Pode ser construído com comando ou bloco de código

O *do*

- Sintaxe semelhanto ao `while` normal
- Recebe argumento tipo `bool`
- Pergunta é realizada *depois* da repetição
- Não é um comando
- É um controlador de fluxo
- Acaba com ponto-e-vírgula
- Pode ser construído com comando ou bloco de código

```
do <comand>; while(<cond>);
```

O *do*

- Sintaxe semelhanto ao `while` normal
- Recebe argumento tipo `bool`
- Pergunta é realizada *depois* da repetição
- Não é um comando
- É um controlador de fluxo
- Acaba com ponto-e-vírgula
- Pode ser construído com comando ou bloco de código

```
do <comand>; while(<cond>);
```

```
do
{
  <comand1>;
  //...
  <comandN>;
}
while(<cond>);
```

O *do while* linear

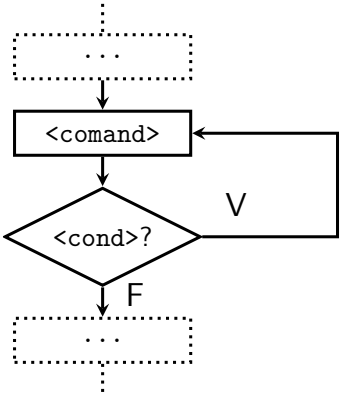

Figura: Fluxograma de *do while* simples linear

O *do while* blocular

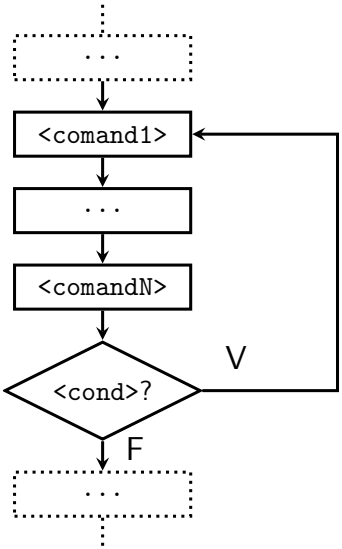

Figura: Fluxograma de *do while* simples blocular

Vamos testar!