

# C++

## Básico ao Avançado

E se...

Heitor Rodrigues Savegnago

UFABC Rocket Design

2017.3

- 1 Fluxo
- 2 Decisões
- 3 Aninhamento
- 4 Repetições
- 5 Hora de brincar

# Fluxo

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

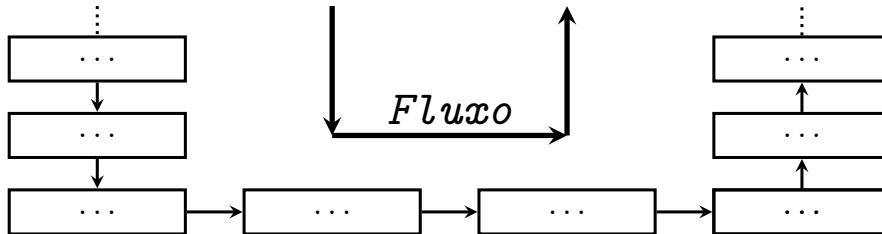


Figura: Fluxograma de fluxo simples

# Ambiguidade

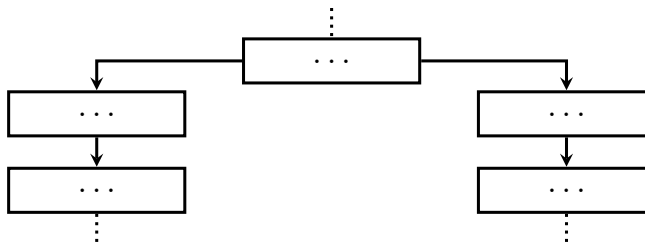


Figura: Fluxograma de fluxo ambíguo

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

# O *if*

- 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

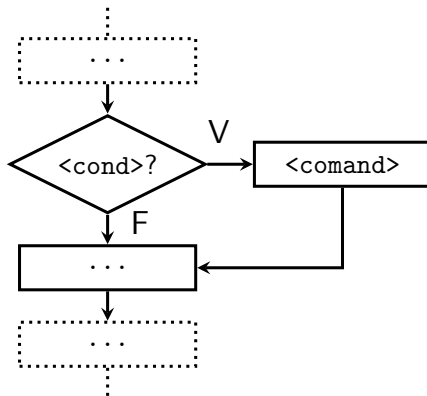


Figura: Fluxograma de *if* simples linear

# O *if* blocular

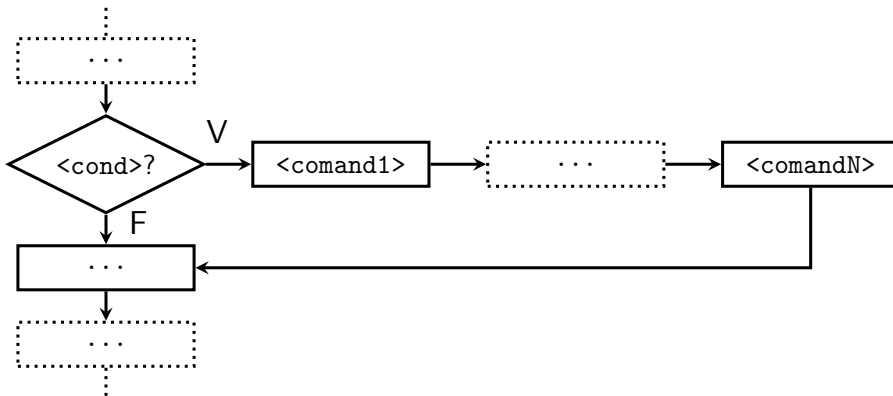


Figura: Fluxograma de *if* simples blocular

# O *else*

- Quando se 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

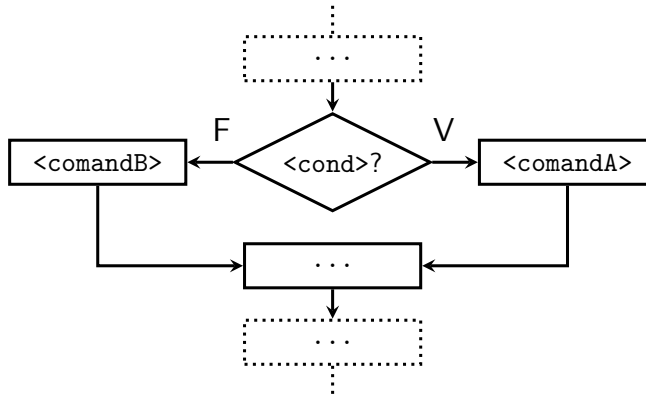


Figura: Fluxograma de *if else* simples linear

# O *if else* blocular

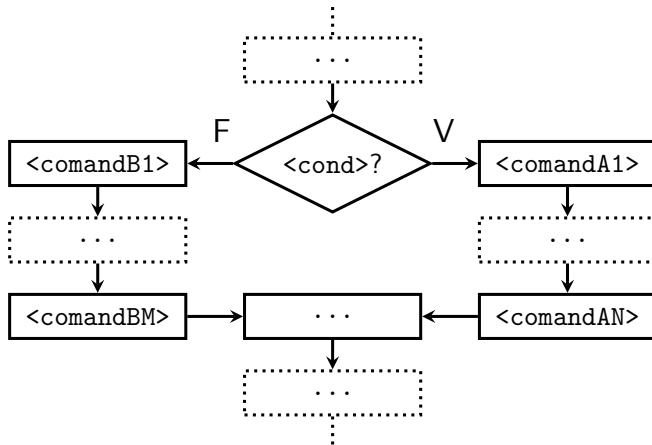


Figura: Fluxograma de *if else* simples blocular

# Estruturas aninhadas

## ■ Estruturas em qualquer lugar

```
if (<cond0>)
{
    //...
    if (<condA>)
    {
        <comandA1>;
        //...
        <comandAN>;
    }
    else
    {
        <comandB1>;
        //...
        <comandBM>;
    }
    //...
}
```

```
else
{
    //...
    if (<condB>)
    {
        <comandC1>;
        //...
        <comandCP>;
    }
    else
    {
        <comandD1>;
        //...
        <comandDQ>;
    }
    //...
}
```

# 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
- 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

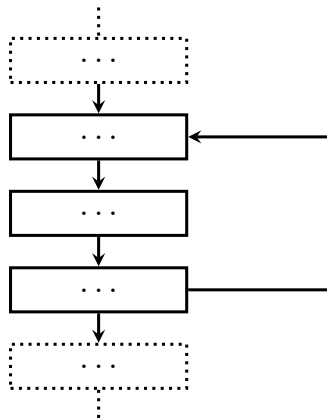


Figura: Fluxograma de fluxo repetitivo

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

# Mas...

- 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*

- 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

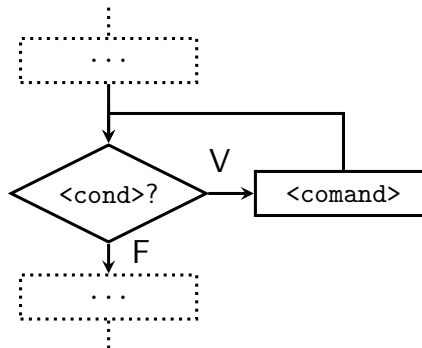


Figura: Fluxograma de *while* simples linear

# O *while* blocular

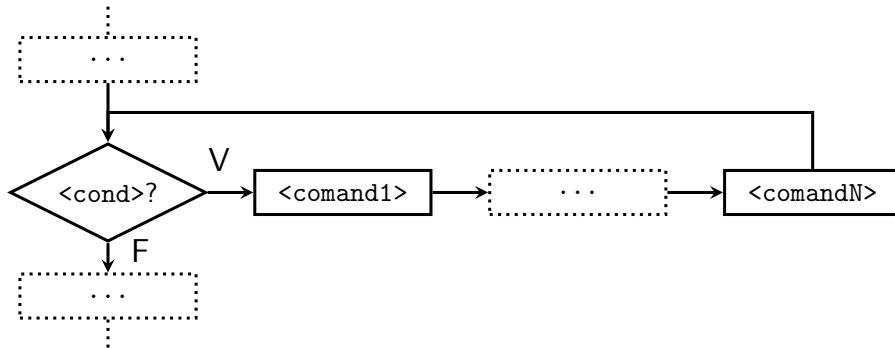


Figura: Fluxograma de *while* simples blocular

# Algoritmo de Euclides

```
while (b!=0)
```

```
{
```

```
    X = a>b?a:b;
```

```
    Y = a<b?a:b;
```

```
    a = X-Y;
```

```
    b = Y;
```

```
}
```

# O do

- Sintaxe semelhante 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

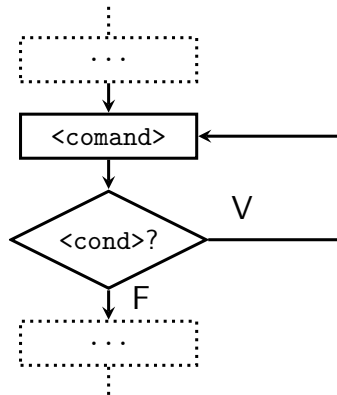


Figura: Fluxograma de *do while* simples linear

# O *do while* blocular

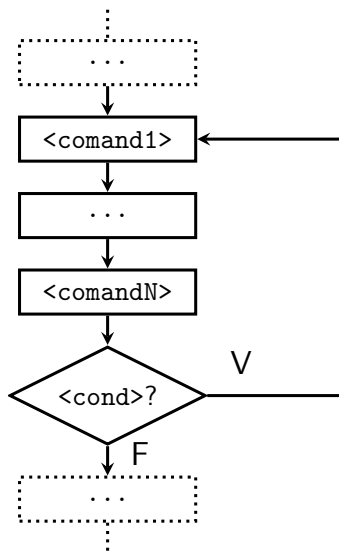


Figura: Fluxograma de *do while* simples blocular

# Vamos testar!