Unidade 9: Introdução às Estruturas de Decisão

Prof. Daniel Caetano

Objetivo: Tomando decisões no código de programação.

Bibliografia: ASCENCIO, 2007; MEDINA, 2006; SILVA, 2010; SILVA, 2006.

INTRODUÇÃO

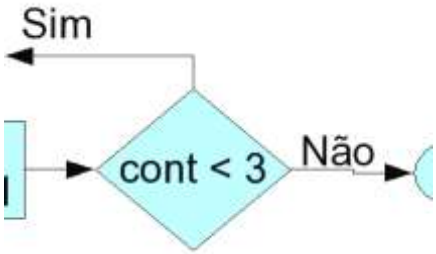
Até agora temos visto uma porção de recursos que nos permitem, basicamente, usar o computador como uma calculadora de luxo. Isso porque aprendemos a comandá-lo a receber informações, fazer cáclulos e apresentar respostas, sempre da mesma forma, sem nenhuma variação.

Isso ocorre porque, até o momento, não vimos como solicitar ao computador que **tome decisões**, isto é, ainda não vimos como solicitar que o computador tome algumas medidas diferentes em situações específicas.

Nesta aula, veremos exatamente como fazer isso.

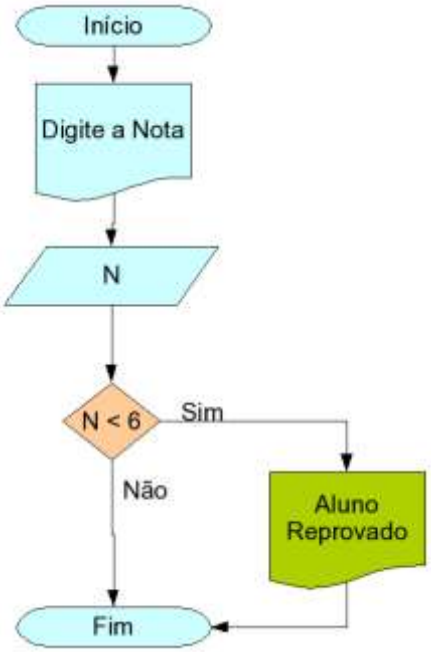
1. ESTRUTURA DE DECISÃO SIMPLES

Já vimos, em um fluxograma, que a estrutura de decisão é representada por um losango, como indicado a seguir:



O fluxo chega ao losango que apresenta a decisão a ser tomada (no caso, representada pela proposição **cont < 3**) e temos duas saídas: uma delas indica o fluxo para o caso de a **proposição** ser verdadeira (indicada por **sim** no fluxograma) e outra para o caso de a **proposição** ser falsa (indicada por **não** no fluxograma).

Nesta primeira abordagem, veremos como lidar com um tipo simplificado de decisão, em que algo de diferente acontece apenas quando a **proposição é verdadeira**. Observe o fluxograma a seguir:



Note que a decisão, indicada no bloco laranja, se positiva, resulta na execução de um **bloco adicional** (o bloco em verde), no caso, imprimindo a mensagem "Aluno Reprovado". Como poderíamos programar isso em C/C++? Observe o código a seguir e tente identificar a relação com o fluxograma:

```
# Decide se aluno foi reprovado

N = float( input ("Digite a nota: ") )

if N < 6.0 :
      print ("Aluno Reprovado")
```

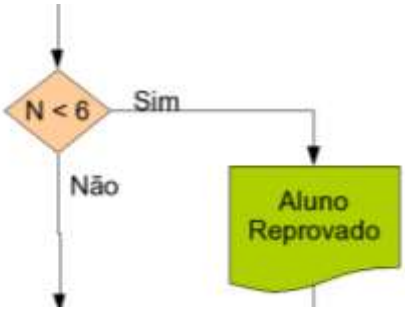
A instrução **if** serve para que o computador tome uma decisão de executar um trecho de código ou não (traduzindo para o português, "if" significa "se"). **Sempre** indicamos a decisão como uma **proposição lógica** (isto é, o resultado é avaliado como **falso** ou **verdadeiro**) seguida de dois pontos... O bloco de código indicado **após** a instrução **if** (em verde, no exemplo), deslocado de alguns espaços, e será executado **somente se a proposição for verdadeira**.

Assim, o código abaixo...

```
if N < 6.0 :
      print ("Aluno Reprovado")
```

...deve ser lido da seguinte forma: "Se N < 6.0, então execute o bloco em verde".

Comparando o programa com o fluxograma, a linha do **if**, em laranja no código, representa o losango laranja no fluxograma. O bloco a ser executado caso o resultado da proposição seja **verdadeiro** está em verde no código, assim como o bloco em verde do fluxograma. Observe!

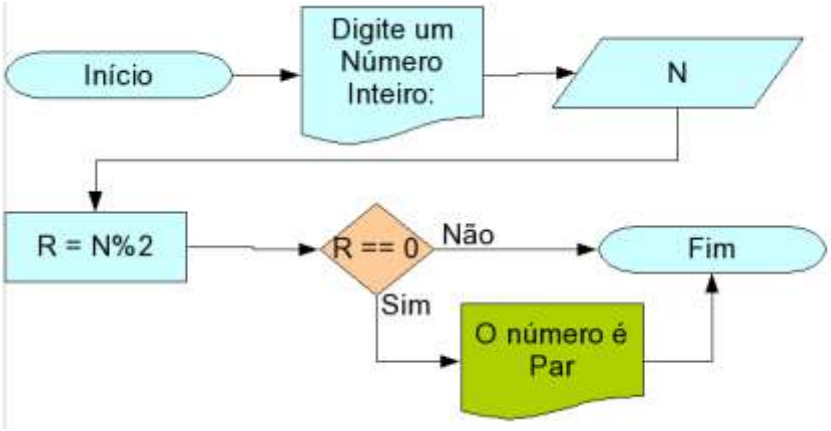


Genericamente, a instrução **if** tem a seguinte estrutura (os termos em cinza devem ser substituídos por código!):

if proposição_lógica **:**

código a executar se a proposição for verdadeira

Vejamos um outro exemplo.



Como seria o código para o programa acima?

Vejamos!

```
# Verifica se número é par

N = int( input ("Digite um número inteiro: ") )

R = N % 2

if R == 0 :
      print ("O número é par")
```

1.1. Comparadores Usados em Proposiçoes Lógicas

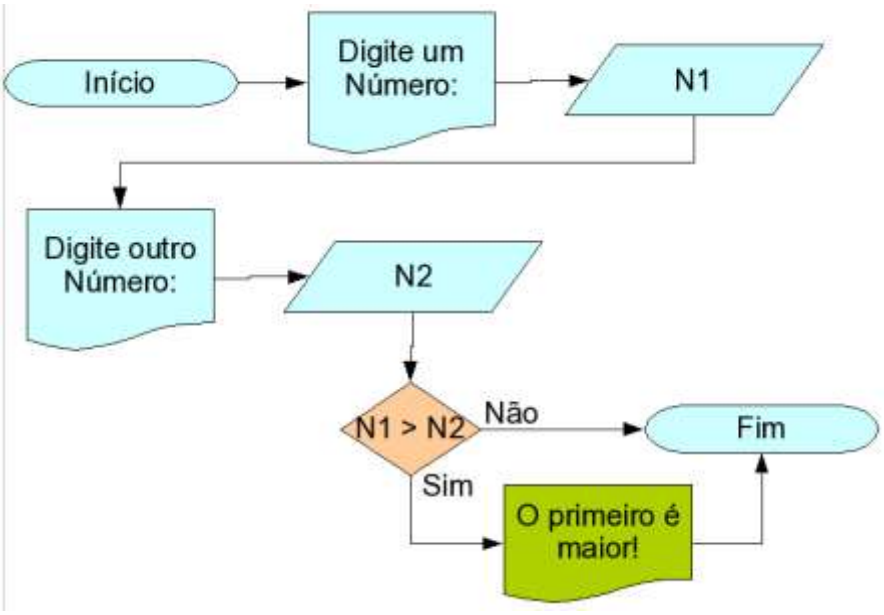
Nas primeiras aulas vimos o que eram expressões lógicas. Vejamos agora a simbologia usada pelo Pyhton. Primeiramente, vejamos os **comparadores**, que são os sinais de operação. Eles estão indicados na tabela abaixo, juntamente com sua descrição:

Comparador	Exemplo	Significado
==	x == 2	Testa igualdade entre os elementos
>	x > 2	Testa se lado esquerdo é maior que lado direito
>=	x >= 2	Testa se lado esquerdo é maior ou igual ao lado direito
<	x < 2	Testa se lado esquerdo é menor que lado direito
<=	x <= 2	Testa se lado esquerdo é menor ou igual ao lado direito
!=	x != 2	Testa se elementos são diferentes

2. EXERCÍCIO

Faça um programa que leia dois números e responda se o primeiro número é maior que o segundo!

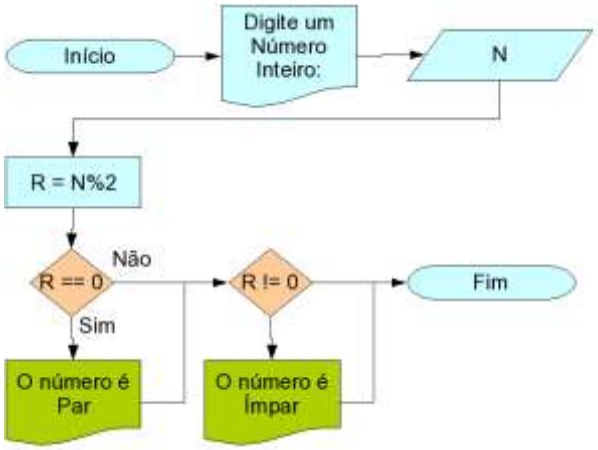
SOLUÇÃO



```
# Verifica se primeiro número é maior

N1 = int( input ("Digite um número: ") )
N2 = int( input ("Digite outro número: ") )

R = N % 2

if N1 > N2 :
      print ("O primeiro é maior!")
```

3. ESTRUTURA DE DECISÃO SIMPLES - MÚLTIPLAS DECISÕES

Nos exemplos vistos até agora, tomávamos uma única decisão... mas e se quisermos fazer várias verificações, como indicado a seguir?



Não há problema algum! Observe o código:

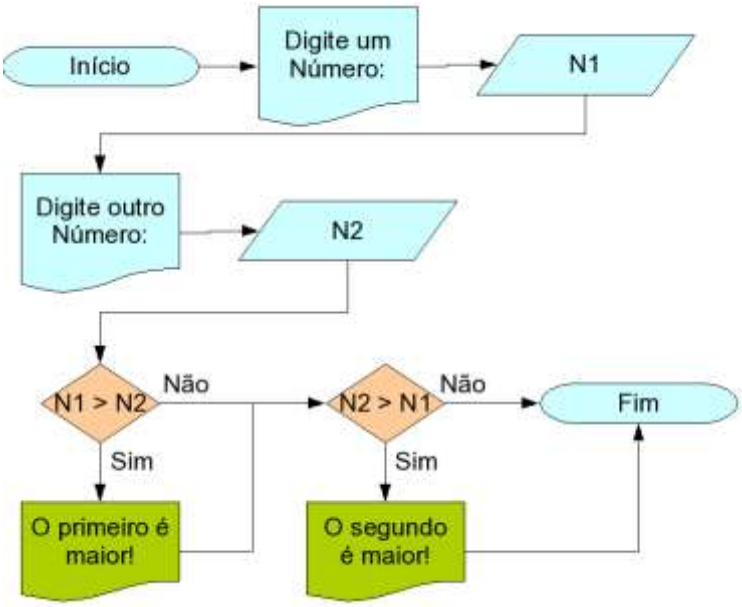
```
# Verifica se número é par ou ímpar

N = int( input ("Digite um número inteiro: ") )

R = N % 2

if R == 0 :
      print ("O número é par")

if R != 0 :
      print ("O número é impar")
```

Ou seja, para tomar várias decisões em seguida, basta usar vários **if**s em sequência!

4. EXERCÍCIO

Faça um programa que leia dois números e responda se o primeiro número é o maior ou se o segundo número é o maior!

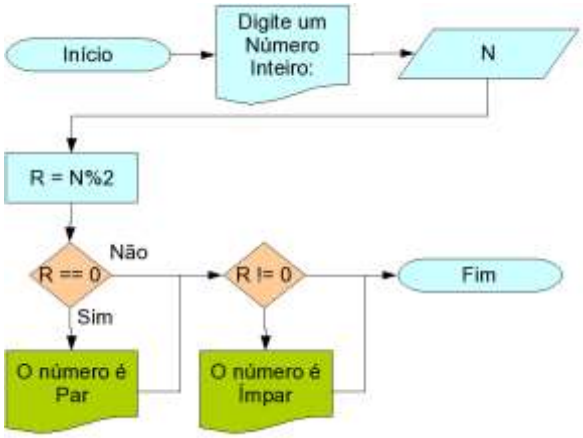
SOLUÇÃO



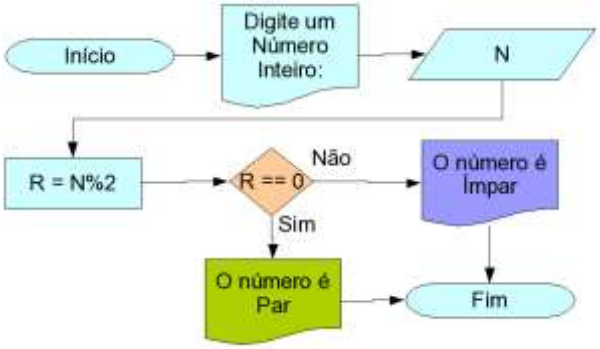

```
# Verifica se primeiro ou segundo número é o maior

N1 = int( input ("Digite um número: ") )
N2 = int( input ("Digite outro número: ") )

R = N % 2

if N1 > N2 :
      print ("O primeiro é maior!")

if N2 > N1 :
      print ("O segundo é maior!")
```

5. ESTRUTURA DE DECISÃO COMPLETA

Em algumas situações queremos que algo especial ocorra tanto quando a proposição é verdeira quanto quando a proposição é falsa. Vimos, anteriormente, que é possível resolver este tipo de problema usando dois **if**s:

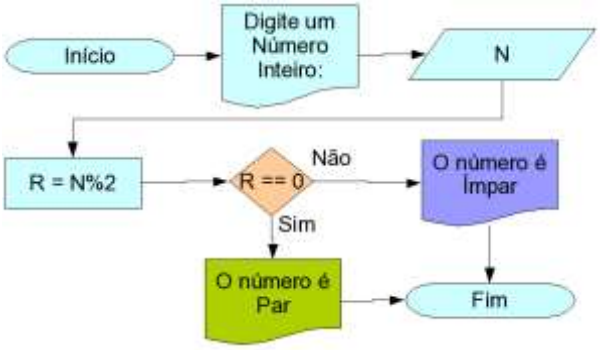


Observe que as proposições são inversas: se R==0 é verdadeiro, R!= 0 **nunca** será verdadeira (e vice-versa). Isso ocorre porque as proposições são complementares. Será que não é possível indicar como apresentado abaixo?



Compare os fluxogramas... veja como o resultado é absolutamente o mesmo!

Além disso, o último fluxograma, com uma única estrutura de decisão, é muito mais claro e fácil de entender do que a versão com duas decisões.



Bonito na teoria, mas... como implementar isso no código? Observe o código a seguir, identificando os blocos pelas cores:

```
# Verifica se número é par ou ímpar

N = int( input ("Digite um número inteiro: ") )

R = N % 2

if R == 0 :
      print ("O número é par")

else :
      print ("O número é impar")
```

Enquanto a instrução **if** indica que o primeiro bloco deve ser executado caso a proposição seja verdadeira, a instrucão **else** indica que o bloco seguinte deve ser executado se a proposição for falsa (em português, "else" significa "senão"). Observe, porém, que a instrução **else** não indica uma proposição lógica. Isso ocorre porque a instrução **else** é **complementar** à instrução if.

As linhas abaixo...

```
if R == 0 :
   print ("O número é par")
else :
   print ("O número é ímpar")
```

...devem ser lidas assim: "**Se R é 0, então execute o bloco em verde. Se não, execute o bloco em roxo**".

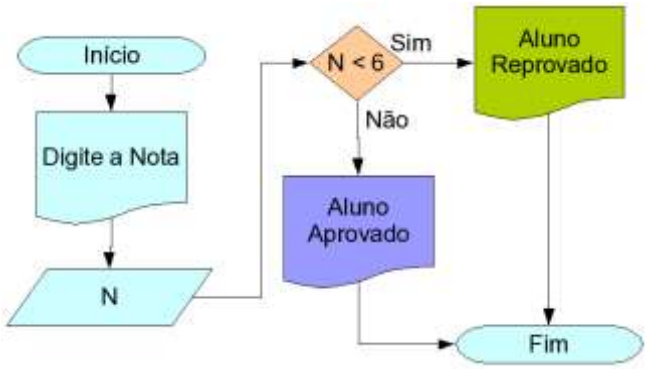
Assim, a instrução **if** completa tem a seguinte estrutura (os termos em cinza devem ser substituídos por código!):

if proposição_lógica **:**
    código a executar se a proposição for verdadeira
else :
    código a executar se a proposição for falsa

6. EXERCÍCIO

Faça um programa que receba a nota do aluno (float) e responda que o aluno está reprovado se a nota for menor que 6.0 e, caso contrário, que o aluno está aprovado.

SOLUÇÃO




```
# Decide se aluno foi reprovado ou aprovado

N = float( input ("Digite a nota: ") )

if N < 6.0 :
      print ("Aluno Reprovado")
else :
      print ("Aluno Aprovado")
```