Unidade 11: Estrutura de Repetição Simples

Prof. Daniel Caetano

Objetivo: Realizando decisões de repetição simples no código de programação.

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

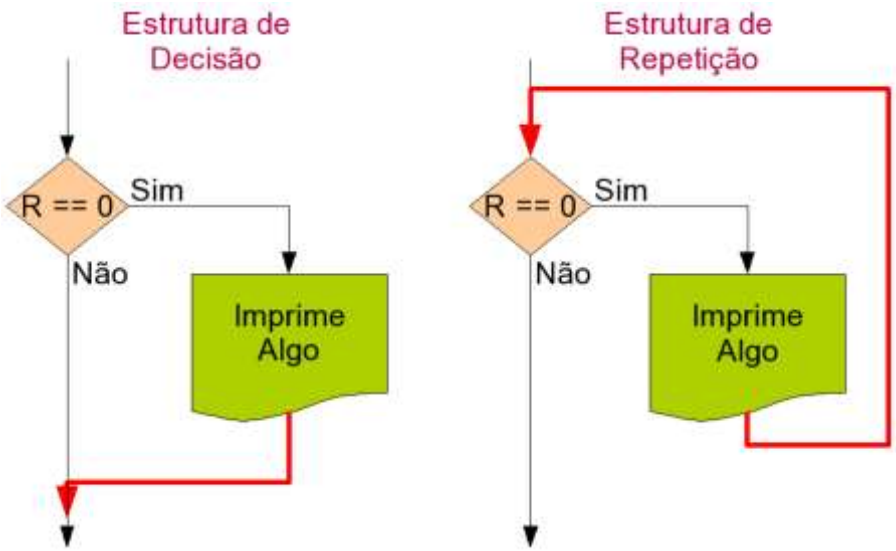
INTRODUÇÃO

Com o conteúdo da aula anterior, já é possível construir programas bastante complexos, pois já descobrimos como fazer com que o computador tome decisões. Existe, no entanto, um tipo de tarefa que ainda não aprendemos: como orientá-lo a repetir um determinado bloco de código.

Nesta aula, veremos exatamente como fazer isso.

1. ESTRUTURA DE REPETIÇÃO SIMPLES: WHILE

Já vimos, há muitas aulas atrás, uma estrutura de repetição em um fluxograma. Você provavelmente não se lembra dela porque ela não possui um símbolo próprio: no fluxograma ela aparece como uma decisão. Observe:



Note que, em ambos os casos, a decisão é tomada para decidir se o bloco será executado: se a proposição for falsa, o bloco é, simplesmente, "pulado". A única diferença entre a estrutura de decisão tradicional (com **if**) e a estrutura de repetição (ou *estrutura de decisão de repetição*) é o que ocorre com o programa **depois** que o bloco é executado: no

caso do **if**, a exeução volta ao fluxo principal; no caso da repetição, a execução volta para a decisão... isto é, volta para trás!

E no código, qual a mudança necessária?

A mudança é uma só: trocar a palavra **if** (que pode ser traduzida como "se") pela palavra **while** (que pode ser traduzida como "enquanto"). Observe como ficam os códigos para as duas estruturas indicadas anteriormente:

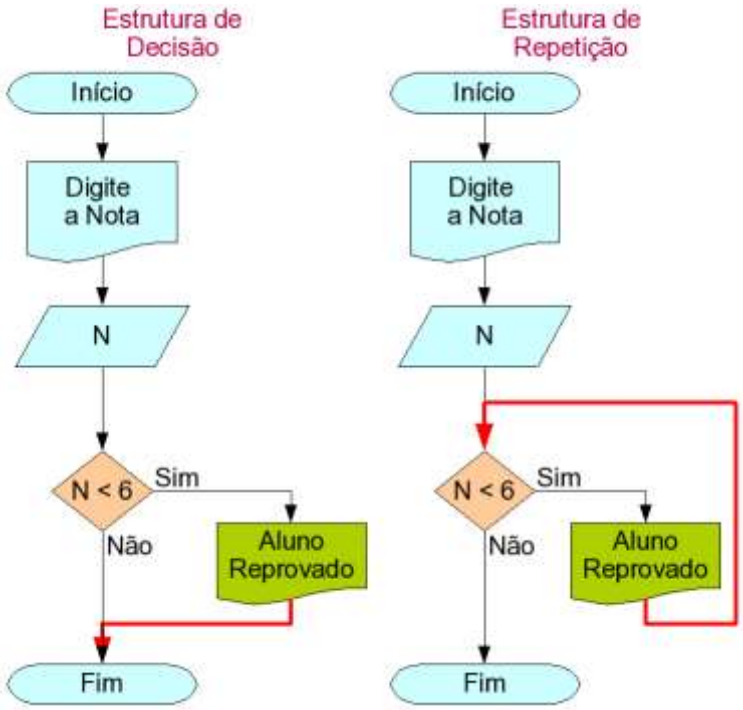
Estrutura de Decisão:

```
if R == 0:
   print("Imprime Algo")
```

Estrutura de Repetição

```
while R == 0:
   print("Imprime Algo")
```

Vamos ver isso atuando na prática? Em uma das aulas anteriores, construímos um programa que recebia a nota de um aluno e imprimia se o aluno havia sido reprovado. Vamos modificá-lo para que, ao invés de usar **if**, use **while** e ver o que ocorre. A modificação está indicada no fluxograma a seguir:



A seguir temos o programa original e o programa alterado. Observe atentamente: a única diferença entre ambos é a troca da palavra **if** pela palavra **while**.

Para testá-lo, primeiramente digite a versão original e se lembre do que acontece, digitando uma nota maior que 6,0 e outra menor que 6,0. Depois de se lembrar do que ocorre, modifique-o e execute-o novamente com uma nota maior que 6,0 e outra menor que 6,0.

Versão com Estrutura de Decisão

```
# Verifica se aluno está reprovado

N = float(input("Digite a nota: "))
if N < 6.0:
      print("Aluno Reprovado")
```

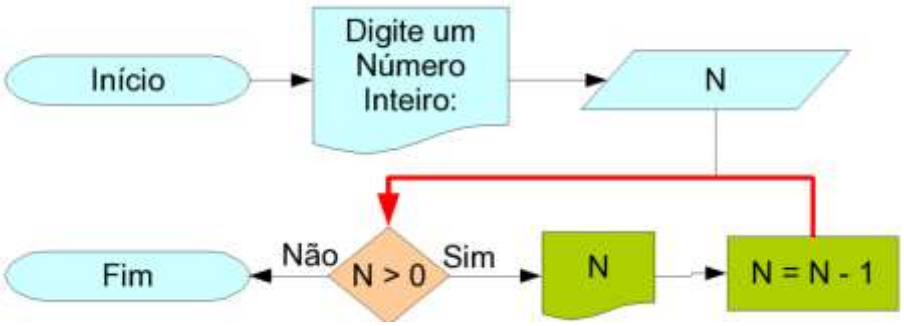
Versão com Estrutura de Repetição

```
# Verifica se aluno está reprovado

N = float(input("Digite a nota: "))
while N < 6.0:
      print("Aluno Reprovado")
```

Percebeu a diferença? (Para "matar" o programa de repetição, você terá que fechar o Spyder!) Uma pequena diferença no código provoca uma enorme mudança no resultado final.

Obviamente este caso é útil para mostrar como o **while** funciona, mas não é uma aplicação lá muito útil do while, não é? Vejamos, então, uma aplicação diferente:



Este programa recebe um número inteiro e... tente entender o que ele faz. Depois de tentar, digite o código que o representa, a seguir, e veja se acertou!

```
# Programa misterioso com repetição

N = int(input("Digite um Número Inteiro: ")

while N > 0:
      print( N )
      N = N - 1
```

Observe o que o código faz! Teste o programa com valores diferentes: 5, 1, 0, -10...

Experimente modificar a condição do while para:

while **N >= 0** :

E veja o que ocorre. O que mudou? Note como isso pode ser a diferença entre o programa fazer exatamente o que você quer ... ou não! **Atenção no momento de construir as proposições lógicas**!

Por fim, observe que é **responsabilidade do programador** fazer com que o laço de repetição (também chamado de *loop*) seja finalizado, de alguma forma. A linha que é responsável por isso no código anterior é a linha

N = N - 1;

É ela quem faz com que o valor de N seja alterado para que, em algum momento no futuro, a condição N > 0 do **while** se torne inválida. Tente mudar a linha N = N - 1; por:

N = N + 1;

E veja o que ocorre! Teremos um problema. Você sabe o por quê? Porque agora o valor de N não se tornará menor que zero dentro do laço no momento esperado, fazendo com que o programa se comporte de um jeito inadequado!

A estrutura da instrução while é:

while proposição_lógica **:**
    código a repetir enquanto a proposição for verdadeira

A instrução (ou as instruções) a serem repetidas devem aparecer indentadas (recuadas), conforme a indicação acima, como na estrutura de decisão **if**. Além disso, qualquer expressão válida em um **if** pode ser usada em um **while**.

2. EXERCÍCIO

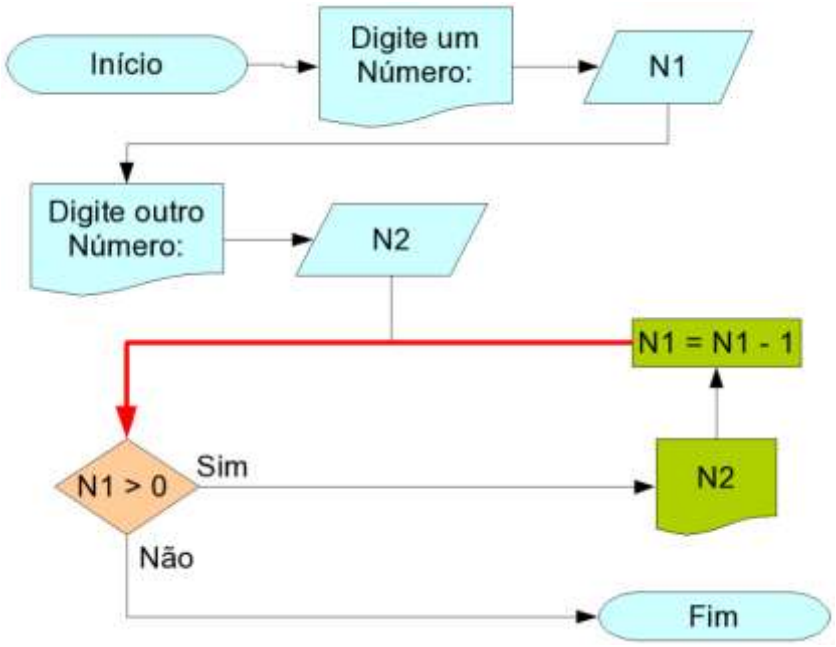
a) Faça um programa que leia dois números: N1 e N2. Faça agora com que o programa imprima imprima o número N2 por N1 vezes.

b) Modifique o programa para que, em cada passo de contagem do N1, o programa imprima o valor atual de N1 e o valor de N2.

c) Modifique agora o programa para que, em cada passo de contagem do N1, o programa imprima o valor atual de N1 multiplicado por N2.

d) Modifique o programa para que ele faça a contagem de 0 a N1.

SOLUÇÃO A

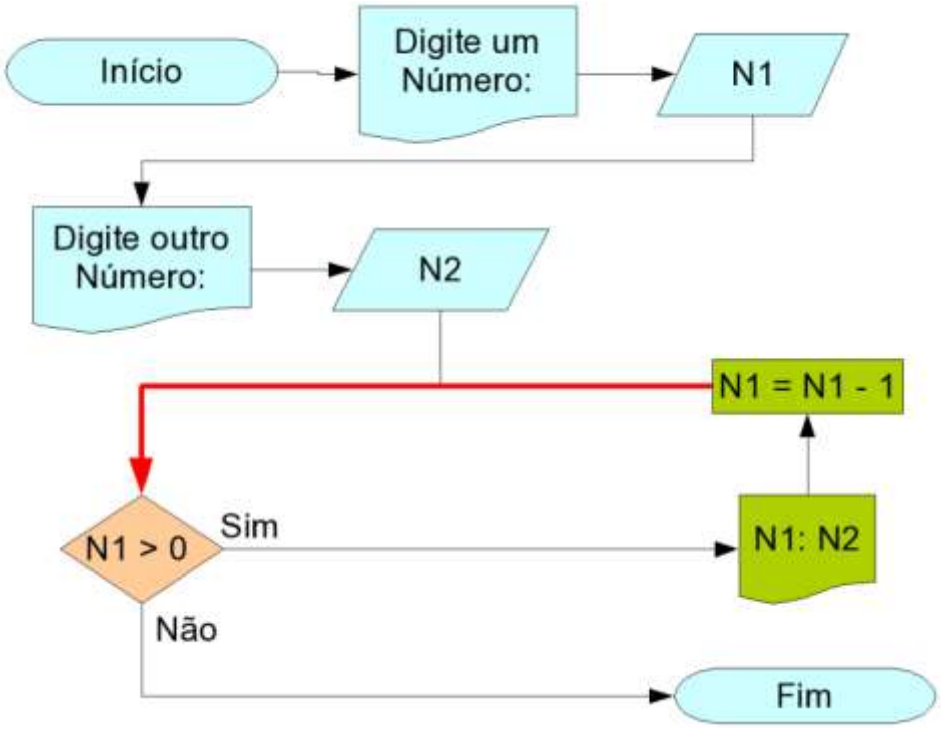



```
# Etapa A

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

while N1 > 0 :
      print ( N2 )
      N1 = N1 - 1
```

SOLUÇÃO B

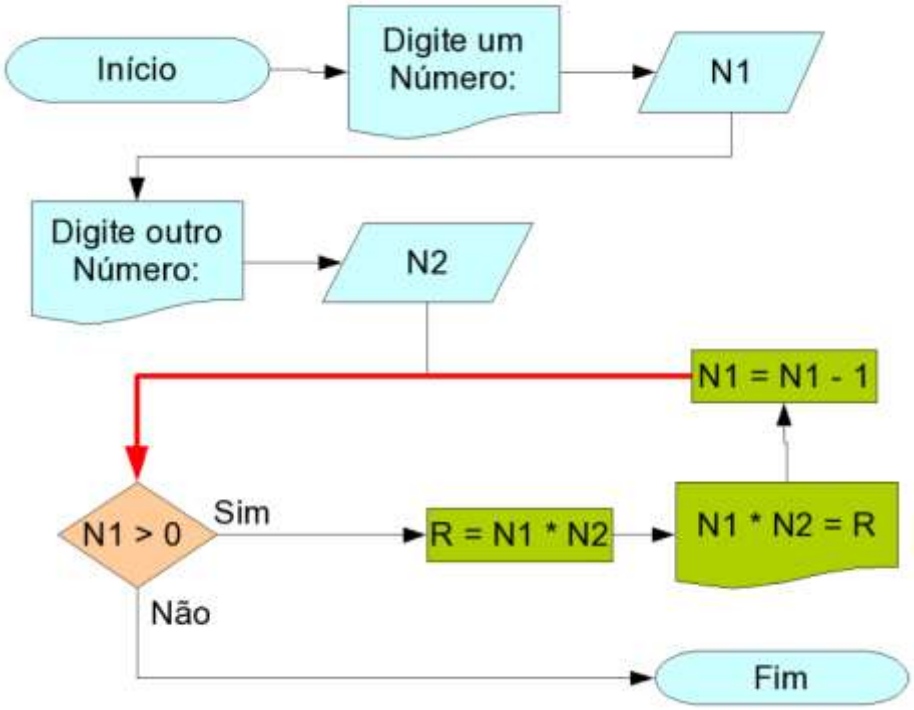



```
# Etapa B

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

while N1 > 0 :
      print ( N1, ": ", N2 )
      N1 = N1 - 1
```

SOLUÇÃO C

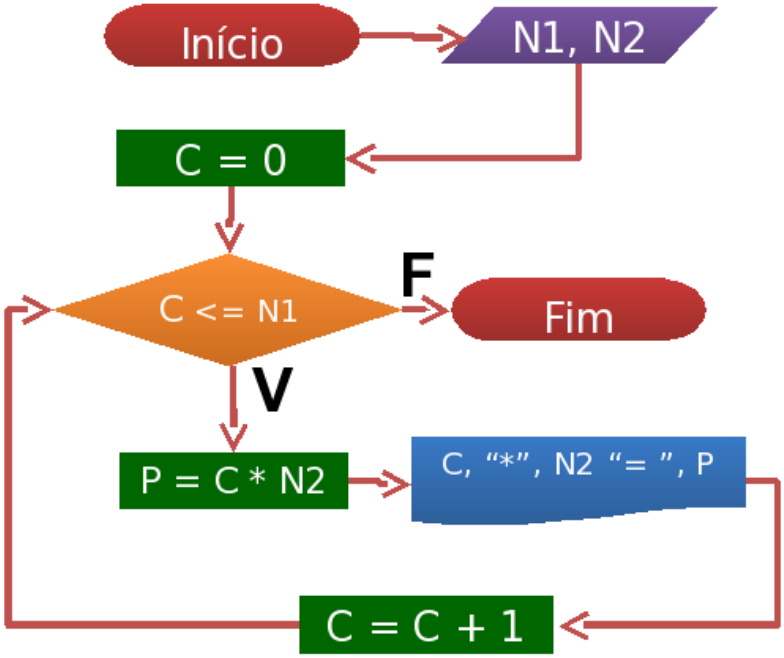



```
# Etapa C

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

while N1 > 0 :
      R = N1 * N2
      print ( N1, "*", N2, "=", R )
      N1 = N1 - 1
```

SOLUÇÃO D




```
# Etapa C

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

C = 0

while C <= N1 :
      R = C * N2
      print ( C, "*", N2, "=", R )
      C = C + 1
```