MC-102 — Aula 08
Comandos Repetitivos

Eduardo C. Xavier

Instituto de Computação – Unicamp

2 de Abril de 2019

Roteiro

1. Laços Encaixados
   - Números Primos
   - Dados
   - Mega-Sena
2. Exercícios

Laços Encaixados: Primos

- A geração de números primos é uma parte fundamental em sistemas criptográficos como os utilizados em *internetbanking*.
- Um sistema de criptografia muito utilizado é o RSA, onde, dados 2 números primos (grandes), p e q, calcula-se $n = pq$.
- Dado n, calcula-se uma chave privada e e outra pública d para trocas de informações.
- Pode-se demonstrar que, dado n, se for "fácil" fatorar n em p e q, quebra-se o sistema RSA.

Laços Encaixados: Primos

- RSA Factoring Challenge: A RSA-Laboratories oferece prêmios para aqueles que conseguirem fatorar um dado n em seus primos.
- Por exemplo, há um prêmio de US$100.000,00 para quem conseguir fatorar um dado n de 1024 bits.
- O número RSA-1024 fornecido é

```
RSA-1024 = 135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563
```

Laços Encaixados: Primos

- Até 2002 não se conhecia algoritmos rápidos para se verificar se um número é primo ou não.
- Em 2002 o Prof. Manindra Agrawal da IIT, Kanpur junto com os alunos Neeraj Kayal e Nitin Saxena foram os primeiros a propor um algoritmo rápido para se testar se um número é primo ou não.
- Eles receberam diversos prêmios pela invenção como o Fulkerson Prize 2006, e o Gödel Prize 2006.
- O problema de **fatoração em primos** permanece em aberto, ou seja, não se conhecem algoritmos rápidos para resolve-lo.

Laços Encaixados: Primos

- O algoritmo que vimos para testar se um número é primo ou não é muuuuuuito lento para resolver casos práticos.
- Um computador i7 pode executar ≈300.000 MIPS (Milhões de Instruções Por Segundo).
- Usando nosso algoritmo para testar se um número n de 1024 bits é primo ou não, espera-se executar $\approx \sqrt{2^{1024}} = 2^{512}$ instruções.
- O tempo esperado será de $\frac{2^{512}}{300000 \times 10^6} \approx 4 \times 10^{142}$ segundos ou $\approx 1.41 \times 10^{135}$ anos!! (O Universo tem $\approx 15 \times 10^9$ anos)

Laços Encaixados: Primos

- Sabemos testar se um determinado número é ou não primo (com um algoritmo lento).
- Imagine agora que queremos imprimir os n primeiros números primos.
- Como resolver este problema?

Laços Encaixados: Primos

- O programa abaixo verifica se o valor candidato a primo na variável **cand** corresponde a um primo:

```
eprimo = True
for div in range(2,cand//2 + 1):
    if cand % div == 0:
        eprimo = False
        break
if eprimo:
    print(cand)
```

Laços Encaixados: Primos

- Criamos um laço externo e usamos uma variável contadora **impressos**, que contará o número de primos impressos durante a execução deste laço.

```
while impressos < n:

  #trecho do código anterior que
  #checa se candidato é ou não é primo

  if eprimo:
    print(cand)
    impressos = impressos + 1

  cand = cand + 1 #Testa próximo número candidato a primo
```

Laços Encaixados: Primos

- Incluímos uma parte inicial de código para leitura de **n** e inicialização de variáveis.
- Para finalizar, basta incluir o trecho de código que checa se um número é primo ou não.

```
n = int(input('Quandidade de primos:'))

impressos = 0
cand = 2
while impressos < n:
    #trecho do código que checa
    #se candidato é ou não é primo

    if eprimo:
        print(cand)
        impressos = impressos + 1

    cand = cand + 1 #Testa próximo número candidato a primo
```

Laços Encaixados: Primos

Código completo:

```
n = int(input('Quandidade de primos:'))

impressos = 0
cand = 2
while impressos < n:
    eprimo =  True
    for div in range(2,cand//2+1):
        if cand % div == 0:
            eprimo = False
            break
    if eprimo:
        print(cand)
        impressos = impressos + 1

    cand = cand + 1 #Testa próximo número candidato a primo
```

Laços Encaixados: Primos

- O que acontece se mudarmos a variável indicadora **eprimo** para fora do primeiro laço **while**? Faz diferença?

```
n = int(input('Quandidade de primos:'))
impressos = 0
cand = 2
eprimo = True    # <--Saiu do laço, faz diferença?
while impressos < n:
    for div in range(2,cand//2+1):
        if cand % div == 0:
            eprimo = False
            break
    if eprimo:
        print(cand)
        impressos = impressos + 1

    cand = cand + 1 #Testa próximo número candidato a primo
```

Laços Encaixados: Primos

- O que acontece se mudarmos a variável indicadora **eprimo** para fora do primeiro laço **while**? Faz diferença?
- Resposta: Quando testarmos um **cand** que não é primo, a variável **eprimo** será setada para **False** e nunca mais será setada para **True**.
- Logo nenhum outro **cand** posterior será identificado como primo.

```
n = int(input('Quandidade de primos:'))
impressos = 0
cand = 2
eprimo = True    # <--Saiu do laço, faz diferença?
while impressos < n:
    for div in range(2,cand//2+1):
        if cand % div == 0:
            eprimo = False
            break
    if eprimo:
        print(cand)
        impressos = impressos + 1

    cand = cand + 1 #Testa próximo número candidato a primo
```

Laços Encaixados: Primos

- Note que o número 2 é o único número par que é primo.
- Podemos alterar o programa para sempre imprimir o número 2:

```
n = int(input('Quandidade de primos:'))

if n>0:
    print(2)
    impressos = 1
    cand = 3
        .
        .
        .
```

Laços Encaixados: Primos

- Podemos alterar o programa para testar apenas números ímpares como candidatos a primo:

```
n = int(input('Quandidade de primos:'))

if n>0:
    print(2)
    impressos = 1
    cand = 3
    while impressos < n:
        eprimo =  True
        for div in range(2,cand//2+1):
            if cand % div == 0:
                eprimo = False
                break
        if eprimo:
            print(cand)
            impressos = impressos + 1

        cand = cand + 2#Testa próximo número candidato a primo
```

Laços Encaixados: Primos

Além disso sabendo que **cand** é sempre um número ímpar:

- Não precisamos mais testar os divisores (var. **div**) que são pares.
- Se **cand** é sempre um número ímpar, ele não pode ser divisível por um número par, pois seria divisível por 2 também.
- Portanto basta testar divisores ímpares.

Laços Encaixados: Primos

```
n = int(input('Quandidade de primos:'))

if n>0:
    print(2)
    impressos = 1
    cand = 3
    while impressos < n:
        eprimo =  True
        for div in range(3, cand//2+1, 2): #só div ímpar
            if cand % div == 0:
                eprimo = False
                break
        if eprimo:
            print(cand)
            impressos = impressos + 1

        cand = cand + 2#Testa próximo número candidato a primo
```

Laços Encaixados: Dados

Problema

Imprimir todas as possibilidades de resultados ao se jogar 4 dados de 6 faces.

- Para cada possibilidade do primeiro dado, devemos imprimir todas as possibilidades dos 3 dados restantes.
- Para cada possibilidade do primeiro e segundo dado, devemos imprimir todas as possibilidades dos 2 dados restantes....
- Você consegue pensar em uma solução com laços aninhados?

Laços Encaixados: Dados

```
print("D1 D2 D3 D4")
for d1 in range(1, 7):
    for d2 in range(1, 7):
        for d3 in range(1, 7):
            for d4 in range(1, 7):
                print(d1,d2,d3,d4)
```

Laços Encaixados: Mega-Sena

- Na Mega-Sena, um jogo consiste de 6 números distintos com valores entre 1 e 60.

Problema

Imprimir todos os jogos possíveis da Mega-Sena.

Laços Encaixados: Mega-Sena

- Partimos da mesma idéia dos dados: gerar todos os possíveis valores para cada um dos 6 números do jogo.

```
for d1 in range(1, 61):
    for d2 in range(1, 61):
        for d3 in range(1, 61):
            for d4 in range(1, 61):
                for d5 in range(1, 61):
                    for d6 in range(1, 61):
                        print(d1,d2,d3,d4,d5,d6)
```

- Qual a saída deste programa? Ele está correto?

Laços Encaixados: Mega-Sena

```
for d1 in range(1, 61):
    for d2 in range(1, 61):
        for d3 in range(1, 61):
            for d4 in range(1, 61):
                for d5 in range(1, 61):
                    for d6 in range(1, 61):
                        print(d1,d2,d3,d4,d5,d6)
```

- As primeiras linhas impressas por este programa serão:

```
1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 2
1, 1, 1, 1, 1, 3
1, 1, 1, 1, 1, 4
1, 1, 1, 1, 1, 5
1, 1, 1, 1, 1, 6
1, 1, 1, 1, 1, 7
1, 1, 1, 1, 1, 8
1, 1, 1, 1, 1, 9
```

Laços Encaixados: Mega-Sena

- O programa anterior repete números, portanto devemos remover repetições.

```
for d1 in range(1, 61):
    for d2 in range(1, 61):
        for d3 in range(1, 61):
            for d4 in range(1, 61):
                for d5 in range(1, 61):
                    for d6 in range(1, 61):
                        if d1 != d2 and d2 != d3 ... and d5!=d6:
                            print(d1,d2,d3,d4,d5,d6)
```

- Após incluir todos os testes para garantir que os números são distintos, temos a solução?

Laços Encaixados: Mega-Sena

- Não temos uma solução válida, pois o programa irá imprimir jogos como:

```
12, 34, 8, 19, 4, 45
34, 12, 8, 19, 4, 45
34, 12, 19, 8, 4, 45
```

- Todos estes jogos são um único jogo: 4, 8, 12, 19, 34, 45.
- Podemos assumir que um jogo é sempre apresentado com os números em ordem crescente.
- Dado que fixamos o valor de **d1**, **d2** necessariamente é maior que **d1**. Após fixar **d1** e **d2**, **d3** deve ser maior que **d2** etc.

Laços Encaixados: Mega-Sena

Solução correta:

```
for d1 in range(1, 61):
    for d2 in range(d1+1, 61):
        for d3 in range(d2+1, 61):
            for d4 in range(d3+1, 61):
                for d5 in range(d4+1, 61):
                    for d6 in range(d5+1, 61):
                        print(d1,d2,d3,d4,d5,d6)
```

Exercício

- Faça um programa que leia um número n e imprima n linhas na tela com o seguinte formato (exemplo se $n = 6$):

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
```

Exercício

- Faça um programa que leia um número n e imprima n linhas na tela com o seguinte formato (exemplo se $n = 6$):

```
+ * * * * *
* + * * * *
* * + * * *
* * * + * *
* * * * + *
* * * * * +
```

Exercício

- Um jogador da Mega-Sena é supersticioso, e só faz jogos em que o primeiro número do jogo é par, o segundo é ímpar, o terceiro é par, o quarto é ímpar, o quinto é par e o sexto é ímpar. Faça um programa que imprima todas as possibilidades de jogos que este jogador supersticioso pode jogar.