BCM0505-22 – Processamento da Informação

Laços - Parte 3

Maycon Sambinelli
m.sambinelli@ufabc.edu.br
http://professor.ufabc.edu.br/~m.sambinelli/

Outline

Laços Aninhados

Break

Continue

Exercícios

Laços Aninhados

ASCII ART

```
*
**
* *
*   *
*    *
*     *
*      *
*       *
*        *
*         *
*          *
*           *
*            *
*             *
*              *
*               *
*                *
*                 *
*                  *
*                   *
******************
```

Vamos desenhar um triângulo retângulo com ASCII Art

- **ASCII Art**: A ASCII art é uma forma de expressão artística usando apenas 95 caracteres imprimíveis (de um total de 128) definidos pelo padrão ASCII de 1963 (ASCII, abreviação do inglês para *American Standard Code for Information Interchange*).

ASCII ART

```
*
**
* *
*   *
*    *
*     *
*      *
*       *
*        *
*         *
*          *
*           *
*            *
*             *
*              *
*               *
*                *
*****************
```

Vamos desenhar um triângulo retângulo com ASCII Art

- **ASCII Art**: A ASCII art é uma forma de expressão artística usando apenas 95 caracteres imprimíveis (de um total de 128) definidos pelo padrão ASCII de 1963 (ASCII, abreviação do inglês para *American Standard Code for Information Interchange*).

```
  _       _          _
_{v}_    (v)      ('>     ( )
 /-\    //-\\     /V\    // \\
(\_/)   (\_/)    <(_)    (\=/)
 ^ ^     ^ ^       ~~     ~ ~
```

ASCII Art de Triângulo em Python

```python
altura = int(input())

for lin in range(1, altura + 1):
    for i in range(1, lin + 1):
        if i == 1 or i == lin or lin == altura:
            print("*", end="")
        else:
            print(" ", end="")
    print()
```

Lembre-se: o parâmetro end="" instrui o comando `print` a não quebrar a linha

ASCII Art de Triângulo em Python

```python
altura = int(input())

print("*") #primeira linha

for lin in range(2, altura):
    print("*", end="")
    for i in range(2, lin):
        print(" ", end="")
    print("*")

#última linha
for i in range(altura):
    print("*", end="")
print()
```

Listagem de Números Primos

Dado um número *n*, liste os *n* primeiros números primos.

```
1  from math import sqrt
2
3  qtd_primos_para_imprimir = int(input())
4
5  candidato_primo = 2
6  qtd_primos_impressos = 0
7  while qtd_primos_impressos < qtd_primos_para_imprimir:
8      divisor = 2
9      encontrei_divisor = False
10     while not encontrei_divisor and divisor <= sqrt(candidato_primo):
11         if candidato_primo % divisor == 0:
12             encontrei_divisor = True
13         divisor += 1
14     if not encontrei_divisor:
15         print(candidato_primo)
16         qtd_primos_impressos += 1
17     candidato_primo += 1
```

Listagem de Números Primos

- Podemos melhorar o código anterior: sabemos que 2 é o único número primo par. Assim, podemos modificar a forma de atualização da variável `candidato_primo` para que essa só assuma valores ímpares (isso reduz pela metade o número de iterações do laço externo).
- A mesma melhoria pode ser aplicada a `divisor`

Exercício: Faça essas melhorias!

Listagem de Números Primos - Versão 2

Dado um número *n*, lista os *n* primeiros números primos.

```python
from math import sqrt

def eh_primo(num):
    if num == 0 or num == 1:
        return False
    divisor = 2
    encontrei_divisor = False
    while not encontrei_divisor and divisor <= sqrt(num):
        if num % divisor == 0:
            encontrei_divisor = True
        divisor += 1
    return not encontrei_divisor
```

Listagem de Números Primos - Versão 2

```
1  qtd_primos_para_imprimir = int(input())
2
3  candidato_primo = 2
4  qtd_primos_impressos = 0
5  while qtd_primos_impressos < qtd_primos_para_imprimir:
6      if eh_primo(candidato_primo):
7          print(candidato_primo)
8          qtd_primos_impressos += 1
9      candidato_primo += 1
```

- Note como podemos esconder a complexidade dos laços aninhados usando funções

Equações Diofantinas

Uma *Equação Diofantina* é uma equação, tipicamente polinomial, de duas o mais variáveis e de coeficientes inteiros, na qual apenas as soluções inteiras são de interesse.

$$37x + 23y = 101$$

Equações Diofantinas

Uma *Equação Diofantina* é uma equação, tipicamente polinomial, de duas o mais variáveis e de coeficientes inteiros, na qual apenas as soluções inteiras são de interesse.

$$37x + 23y = 101$$

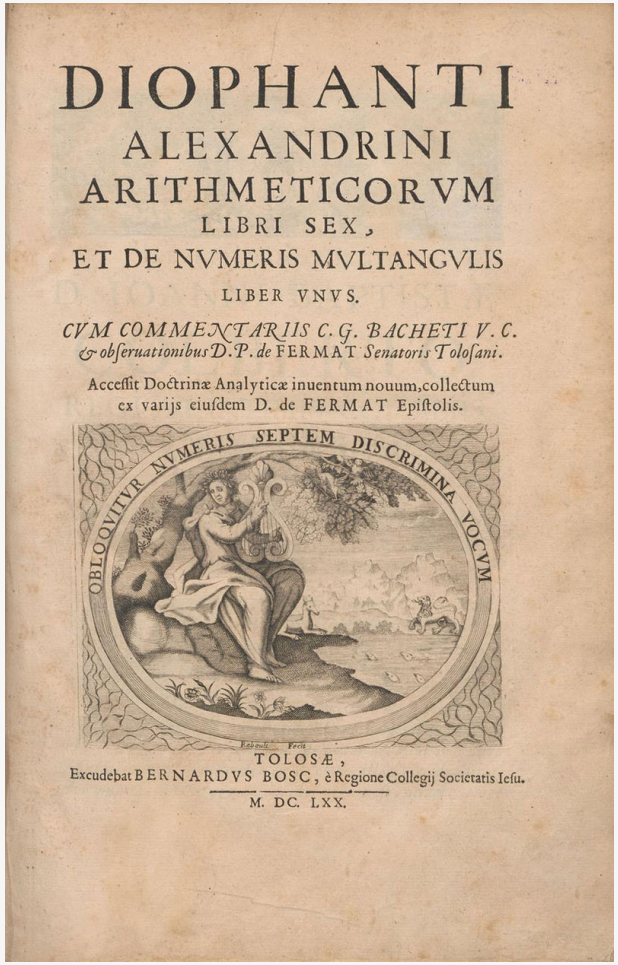
Soluções:

x	y
-47	80
-24	43
-1	6
22	-31
45	-68

Equações Diofantinas

Quando lidamos com (sistemas de) equações Diofinatinas, temos mais incógnitas do que equações e estamos interessados em encontrar inteiros que resolvem simultaneamente todas as equações.

O nome *Diofantina* se refere ao matemático grego do século III Diofanto de Alexandria, considerado o pai da Álgebra.


DIOPHANTI
ALEXANDRINI
ARITHMETICORVM
LIBRI SEX,
ET DE NVMERIS MVLTANGVLIS
LIBER VNVS.
CVM COMMENTARIIS C. G. BACHETI V. C.
& obſeruationibus D. P. de FERMAT Senatoris Toloſani.
Acceſſit Doctrinæ Analyticæ inuentum nouum, collectum
ex varijs eiuſdem D. de FERMAT Epiſtolis.
OBLOQVITVR NVMERIS SEPTEM DISCRIMINA VOCVM
TOLOSÆ,
Excudebat BERNARDVS BOSC, è Regione Collegij Societatis Ieſu.
M. DC. LXX.

Equações Diofantinas Lineares em Python

Faça um programa que lista as soluções para a seguinte equação diofantina linear:

$$ax + by + cz = 0$$

Equações Diofantinas Lineares em Python

Faça um programa que lista as soluções para a seguinte equação diofantina linear:

$$ax + by + cz = 0$$

```python
a = int(input())
b = int(input())
c = int(input())

inf = int(input())
sup = int(input())

for x in range(inf, sup + 1):
    for y in range(inf, sup + 1):
        for z in range(inf, sup + 1):
            valor = a*x + b*y + c*z
            if valor == 0:
                print(f"solução: x = {x}, y = {y}, z = {z}")
```

Equações Diofantinas Lineares em Python

Pequena melhoria:

$$ax + by + cz = 0 \quad \Rightarrow \quad ax + by = -cz \quad \Rightarrow \quad \frac{ax + by}{-c} = z$$

Equações Diofantinas Lineares em Python

Pequena melhoria:

$$ax + by + cz = 0 \quad \Rightarrow \quad ax + by = -cz \quad \Rightarrow \quad \frac{ax + by}{-c} = z$$

```
a = int(input())
b = int(input())
c = int(input())

inf = int(input())
sup = int(input())

for x in range(inf, sup + 1):
    for y in range(inf, sup + 1):
        if (a*x + b*y) % (-c) == 0: # divisão dá número inteiro
            z = (a*x + b*y) // (-c)
            print(f"solução: x = {x}, y = {y}, z = {z}")
```

Break

Break

O comando `break` serve para finalizar imediatamente o laço mais aninhado

- Funciona com `while` e `for`

Break

```
for i in range(1, 8):
    if i == 4:
        break
    print(i)
```

Break

```
for i in range(1, 8):
    if i == 4:
        break
    print(i)
```

```
for i in range(1, 7):
    for j in range(i+1, 8):
        if j == 4:
            break
        print(i, j)
```

Como calcular a raiz quadrada de um número

Dado um x, queremos computar $y = \sqrt{x}$

Como calcular a raiz quadrada de um número

Dado um x, queremos computar $y = \sqrt{x}$

Método Babilônico (ou de Heron)

1. Seja y_1 uma estimativa para $y = \sqrt{x}$
   - Por exemplo, $y_1 = x$
   - Quanto melhor a estimativa mais rapidamente o algoritmo converge para a solução
2. Faça $y_n = \frac{1}{2}(y_{n-1} + \frac{x}{y_{n-1}})$
3. Se $|y_n - y_{n-1}|$ for "grande", volte para 2
4. Devolva y_n

Método Babilônico em Python

```python
ERRO = 1e-12

def mysqrt(num):
    est_atual = num
    while True:
        est_anterior = est_atual
        est_atual = (est_atual + num/est_atual) / 2
        if abs(est_anterior - est_atual) <= ERRO:
            break
    return est_atual

valor = float(input())
raiz = mysqrt(valor)
print(f"raiz de {valor:.4f}: {raiz:.4f}")
```

Método Babilônico em Python

O `break` não é um comando necessário: podemos reescrever o código para eliminá-lo, mas ele é um comando conveniente que geralmente resulta em um código mais simples.

```
1  ERRO = 1e-12
2
3  def mysqrt(num):
4      est_atual = num
5      erro_pequeno = False
6      while not erro_pequeno:
7          est_anterior = est_atual
8          est_atual = (est_atual + num/est_atual) / 2
9          if abs(est_anterior - est_atual) <= ERRO:
10             erro_pequeno = True
11     return est_atual
12
13 valor = float(input())
14 raiz = mysqrt(valor)
15 print(f"raiz de {valor:.4f}: {raiz:.4f}")
```

Método Babilônico em Python

Outra possibilidade, sem `while` e sem variável controladora.

```
1  ERRO = 1e-12
2
3  def mysqrt(num):
4      est_anterior = num + ERRO
5      est_atual = num
6      while abs(est_anterior - est_atual) > ERRO:
7          est_anterior = est_atual
8          est_atual = (est_atual + num/est_atual) / 2
9      return est_atual
10
11 valor = float(input())
12 raiz = mysqrt(valor)
13 print(f"raiz de {valor:.4f}: {raiz:.4f}")
```

Continue

Continue

O comando `continue` serve para ignorar o restante do corpo do laço (while ou for) na iteração atual

Continue

```
for i in range(1, 8):
    if i == 4:
        continue
    print(i)
```

Continue

```python
for i in range(1, 8):
    if i == 4:
        continue
    print(i)
```

```python
for i in range(1, 7):
    for j in range(i+1, 8):
        if j == 4:
            continue
        print(i, j)
```

Um velho conhecido

Dado um *n*, calcule a soma dos *n* primeiros números ímpares.

```python
n = int(input())
numero_atual = 0
soma = 0
qtd_impares_somados = 0
while qtd_impares_somados < n:
    numero_atual += 1
    if numero_atual % 2 == 0:
        continue
    soma += numero_atual
    qtd_impares_somados += 1
print(f"soma dos {n} primeiros ímpares: {soma}")
```

Exercícios

Cada um com o seu quadrado

Dado um inteiro n, escreva um programa que imprima um quadrado de n asteriscos por n asteriscos. Abaixo é possível ver a saída esperada para o caso no qual $n = 5$.

```
*****
*****
*****
*****
*****
```

Tabuada

Escreva um programa que imprima a tabuada do 1 ao 10.

Sugestão de Saída:

```
1 x 1 = 1
1 x 2 = 2
...
2 x 1 = 2
2 x 2 = 4
...
10 x 1 = 1
10 x 2 = 20
...
10 x 10 = 100
```

Triângulo Numérico

Escreva um programa que, dado um número *n*, imprima um triângulo de *n* linhas onde a i-ésima linha do triângulo é composta pelos *i* primeiros naturais. Abaixo é possível ver o padrão de impressão qndo $n = 6$.

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

Triângulo de Pascal

O Triângulo de Pascal é um triângulo numérico infinito formado por números binomiais.
A n-ésima linha do triângulo contém n números, cada um sendo um coeficiente binomial da forma $\binom{n-1}{k}$, para k variando de 0 até $n-1$:

	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$\dots$
$n=1$	$\binom{0}{0}$						
$n=2$	$\binom{1}{0}$	$\binom{1}{1}$					
$n=3$	$\binom{2}{0}$	$\binom{2}{1}$	$\binom{2}{2}$				
$n=4$	$\binom{3}{0}$	$\binom{3}{1}$	$\binom{3}{2}$	$\binom{3}{3}$			
$n=5$	$\binom{4}{0}$	$\binom{4}{1}$	$\binom{4}{2}$	$\binom{4}{3}$	$\binom{4}{4}$		
$n=6$	$\binom{5}{0}$	$\binom{5}{1}$	$\binom{5}{2}$	$\binom{5}{3}$	$\binom{5}{4}$	$\binom{5}{5}$	
$\vdots$							

Lembre-se que $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.
Faça um programa que imprima as n primeiras linhas do Triângulo de Pascal para um dado n.

Meus Primos

Escreva um programa que, dado um número inteiro n, imprima a decomposição de n em fatores primos.

Exemplo:

- Se $n = 17640$, então o seu programa deve imprimir:

```
17640 = 2 x 2 x 2 x 3 x 3 x 5 x 7 x 7
```