MAC 110 — Introdução à Ciência da Computação

Aula 9

Nelson Lago

BMAC – 2024


E a prova, hein?

Prova — questão 1

Simule o código abaixo e selecione as opções correspondentes à saída impressa do programa.

```
n = 20
i = 4
a = 1
b = 0
while b >= 0:
    if i % 2 == 0:
        i = i +1
        n = n -3
    else:
       i = i +3
       n = n -1
    b = a
    a = a + (n - 2*i)
    print((n % 7 + 3) * 7)
```

Prova — questão 2

Cartões de crédito geralmente têm 16 dígitos. Os primeiros dígitos à esquerda indicam a bandeira do cartão (Visa começa com dígito 4, Mastercard varia de 51 a 55). O último dígito (mais à direita) é um dígito verificador calculado a partir dos demais dígitos pelo algoritmo de Luhn (explicado no quadro abaixo) como medida de segurança, a fim de evitar falsificações e erros de digitação.

Preencha as lacunas no código abaixo de forma a obter um programa que, dado um inteiro com os 16 dígitos de um cartão, checa se seu dígito verificador está correto. Em caso afirmativo, o programa ainda informa se a bandeira é Mastercard. Não tente montar o programa testando as combinações possíveis, pois não vai dar tempo; escreva primeiro seu programa e depois procure analisar as opções. A cada opção errada que for selecionada, poderá ser descontada nota do exercício.

Prova — questão 2 (algoritmo de Luhn)

Exemplo: vamos conferir se o dígito verificador 7 está correto no cartão de crédito número $\underline{4}\,9\,\underline{8}\,4\,\underline{2}\,3\,\underline{2}\,0\,\underline{5}\;\;0\,\underline{4}\,8\,\underline{6}\,4\,\underline{2}\,7$.

Numerando os dígitos da direita para esquerda, iniciando em 1, tomamos os dígitos em posições pares (grifadas) multiplicados por dois. Caso o produto resulte em algum número de 2 dígitos, estes devem ser somados. Logo, para o exemplo temos: 2×2, 6×2, 4×2, 5×2, 2×2, 2×2, 8×2, $4 \times 2 \Rightarrow 4$, 12, 8, 10, 4, 4, 16, $8 \Rightarrow 4$, 3, 8, 1, 4, 4, 7, 8. Calculamos então a soma dos valores obtidos: $4 + 3 + 8 + 1 + 4 + 4 + 7 + 8 = 39$. A seguir, adicionamos ao somatório todos os dígitos em posições ímpares que não foram usados na etapa anterior, com exceção do primeiro à direita. Assim, temos $39 + 9 + 4 + 3 + 0 + 0 + 8 + 4 = 39 + 28 = 67$.

Para 67 ser um múltiplo de 10, faltam 3 unidades, então 3 é o nosso dígito verificador e, portanto, o valor 7 à direita está incorreto; o cartão 4984232050486427 é inválido.

Prova — questão 2 (código)

```
cartão = int(input("Digite o número do cartão: "))
dv = cartão % 10
falta = cartão // 10
par, soma = True, 0
L5
    atual = falta % 10
    if par:
        val = 2 * atual
        L9
    else:
        val = atual
    soma += val
    L13
    L14
L15
if dv == calculado:
    L17
    L18
        L19
    print("cartão válido", mastercard)
else:
    print("cartão inválido:", end=" ")
    print("dígito verificador deveria ser", calculado)
```

Prova — questão 2 (alternativas)

L5:

- **`while falta > 0:`**
- `while falta >= 0:`
- `while par:`
- `while falta // 10 > 0:`

L9:

- **`val = val // 10 + val % 10`**
- `val = val // 10 + val`
- `val = val +1`
- `val = val + (val % 10)`

L13:

- **`falta //= 10`**
- `falta %= 10`
- `falta = falta -1`
- `falta = falta - soma`

L14:

- **`par = not par`**
- `par = False or par`
- `par = True or par`
- `par = val > 0`

Prova — questão 2 (alternativas)

L15:

- `calculado = 10 - (soma % 10)`
- `calculado = soma % 10`
- `calculado = (soma +10) % 10`
- `calculado = soma // 10**15`

L18:

- `if cartão // 10**14 >= 51 and cartão // 10**14 <= 55:`
- `if cartão // 10**14 == 51 or cartão // 10**14 == 55:`
- `if not (cartão // 10**14 <= 51 or cartão // 10**14 >= 55):`
- `if not (cartão // 10**14 == 4):`

L17:

- `mastercard = ""`
- `mastercard = False`
- `mastercard = True`
- `mastercard = 0`

L19:

- `mastercard = "(Mastercard)"`
- `mastercard = False`
- `mastercard = True`
- `mastercard = 0`

Prova — questão 3

Considere as seguintes definições:

- **Números amigos:** dizemos que dois naturais a e b são amigos se cada um deles é igual à soma dos divisores próprios do outro. Um exemplo é o par 220 e 284, uma vez que os divisores próprios de 220 são 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 e 110, que somam 284, e os divisores próprios de 284 são 1, 2, 4, 71 e 142, que somam 220.
- **Números mutuamente amigos:** dois números a e b são mutuamente amigos se a razão entre a soma de todos os divisores do número a e o número a é igual à razão entre a soma de todos os divisores do número b e o número b. Por exemplo, os números 30 e 140 são mutuamente amigos, pois:
  - para 30 temos $(1 + 2 + 3 + 5 + 6 + 10 + 15 + 30)/30 = 72/30 = 12/5$;
  - para 140 temos $(1 + 2 + 4 + 5 + 7 + 10 + 14 + 20 + 28 + 35 + 70 + 140)/140 = 336/140 = 12/5$.
- **Números primos entre si:** dois inteiros a e b são primos entre si se o único divisor comum a ambos é 1.

Prova — questão 3

Preencha as lacunas no código abaixo de forma a obter um programa que, dados dois inteiros positivos distintos a e b ($a > 0, b > 0$ e $a \neq b$), verifica se eles satisfazem as definições anteriores. Não tente montar o programa testando as combinações possíveis, pois não vai dar tempo; escreva primeiro seu programa e depois procure analisar as opções. A cada opção errada que for selecionada, poderá ser descontada nota do exercício.

Prova — questão 3 (código)

```
a = int(input("Digite a: "))
b = int(input("Digite b: "))
i, sa = 1, 0
L4
    L5
        L6
    i = i +1
print(a, "gera soma dos divisores próprios", sa)
i, sb = 1, 0
L10
    L11
        L12
    i = i +1
print(b, "gera soma dos divisores próprios", sb)
mdc = a
L16
    mdc = mdc -1
L18
    print("primos entre si")
if sa == b and sb == a:
    print("números amigos")
L22
    print("números mutuamente amigos")
```

Prova — questão 3 (alternativas)

L4:

- `while i < a:`
- `while i <= a:`
- `while i > 0:`
- `while sa < a:`

L5:

- `if a % i == 0:`
- `if a % i != 0:`
- `if a // i > 0:`
- `if sa == i:`

L6:

- `sa = sa + i`
- `sa = sa +1`
- `sa = i`
- `sa = i * a`

L10:

- `while i < b:`
- `while i <= b:`
- `while i > 0:`
- `while sb < b:`

Prova — questão 3 (alternativas)

L11:

- **`if b % i == 0:`**
- `if b % i != 0:`
- `if b // i > 0:`
- `if sb == i:`

L12:

- **`sb = sb + i`**
- `sb = sb +1`
- `sb = i`
- `sb = i * b`

L16:

- **`while a % mdc != 0 or b % mdc != 0:`**
- `while a % mdc != 0 or b % mdc == 0:`
- `while a % mdc != 0 and b % mdc != 0:`
- `while a % mcd == 0 or b % mdc != 0:`

Prova — questão 3 (alternativas)

L18:

- `if mdc == 1:`
- `if mdc > 1:`
- `if mdc >= 1:`
- `if mdc > 0:`

L22:

- `if (sa + a)*b == (sb + b)*a:`
- `if (sa + a)*a == (sb + b)*b:`
- `if (sa + a)//b == (sb + b)//a:`
- `if (sa + a)//a == (sb + b)//b:`

Prova — questão 4

Preencha as lacunas no código abaixo de forma a obter um programa que lê uma sequência de números inteiros terminada por zero em que cada número representa os dados de GPS de um participante de uma corrida de rua. Com base nos valores lidos, o programa calcula qual o participante mais rápido e sua velocidade. Os dados do GPS têm 4 dígitos, sendo os dois últimos um intervalo de tempo (em segundos) e os dois primeiros, a distância percorrida (em metros) pelo participante nesse intervalo. Não tente montar o programa testando as combinações possíveis, pois não vai dar tempo; escreva primeiro seu programa e depois procure analisar as opções. A cada opção errada que for selecionada, poderá ser descontada nota do exercício.

Prova — questão 4 (código)

```
i, vmax, imax = 1, 0, 0
gps = int(input("Digite os dados para o participante num {}".format(i)))
while gps > 0:
    L4
    L5
    v = d/t
    L7
        L8
        L9
    i = i +1
    gps = int(input("Digite os dados para o participante num {}".format(i)))
print("O participante {} é o mais rápido (velocidade {}m/s)".format(imax, vmax))
```

L4:

- **`d = gps // 100`**
- `d = gps % 100`
- `d = gps / 100`
- `d = i + (gps % 100)`

L5:

- **`t = gps % 100`**
- `t = gps // 100`
- `t = gps / 100`
- `t = i + (gps % 100)`

L7:

- **`if v > vmax:`**
- `if vmax > v:`
- `if v > 100:`
- `if d > 100:`

L8:

- **`vmax = v`**
- `vmax = vmax +1`
- `vmax = vmax + i`
- `vmax = i`

L9:

- **`imax = i`**
- `imax = imax +1`
- `imax = imax + i`
- `imax = v`

and now for something slightly different

Por que computação?

O computador é extremamente rápido, mas é uma ferramenta com o mesmo nível de "inteligência" que um martelo

Não é mais fácil fazer manualmente?

Repetições e condições

- **"Algo" precisa acontecer para indicar que as repetições chegaram ao fim**

  (ok, às vezes queremos repetir indefinidamente, mas vamos ignorar isso por enquanto)

- **As repetições são controladas por algum tipo de *condição* baseada no estado de uma *variável***

Partes mínimas de um laço

- **Um laço correto precisa**
  - Inicializar a variável de controle antes do início do laço
  - Verificar a condição adequada a cada iteração para que as repetições aconteçam o número correto de vezes
  - Alterar o valor da variável de acordo com a lógica do programa (no mínimo, na última iteração) para garantir que o laço termine

```
chute = input("Adivinhe qual é minha cor favorita: ")
while chute != "rosa choque":
    chute = input("Errou! Tente novamente: ")
print("Acertou!")
```

Tipos de repetição

- **Dois tipos fundamentais de repetição**
  1. Repetições até atingir um resultado
     - » *Encontrar o próximo primo*
     - » *Reiniciar o jogo até o usuário escolher "sair"*
     - » *...*
  2. Repetições sobre os elementos de um conjunto
     - » *Apresentar todos os pixels de uma foto na tela*
     - » *Trocar todas as letras de um texto para maiúsculas*
     - » *...*

podemos usar mais variáveis no laço além da variável de controle

Indicadores de passagem

Leia uma série de números terminada por zero fornecida pelo usuário e calcule sua soma, informando se os números estavam em ordem crescente

```
ordenado = True
n = int(input("Digite um número (zero para sair): "))
anterior = n
soma = 0
while n != 0:
    if n < anterior:
        ordenado = False
    anterior = n
    soma = soma + n
    n = int(input("Digite um número (zero para sair): "))
print("A soma dos números é", soma , end="")
if ordenado:
    print(" e eles estão em ordem" , end="")
print()
```

Indicadores de passagem

- **Chamamos uma variável similar a `ordenado` do exemplo anterior de indicador de passagem**
- **O uso de indicadores de passagem é um padrão de programação bastante comum**
  - Padrões de programação são técnicas (ou truques!) úteis e, portanto, comuns, porém não óbvios
- **Um indicador de passagem é usado para indicar se "alguma coisa" aconteceu durante o laço**
  - Quando usamos um indicador de passagem, não estamos preocupados com **qual** elemento do laço causou o evento
  - Na verdade, pode haver um ou mais elementos responsáveis por essa mudança; o valor do indicador de passagem é alterado no máximo uma vez

Indicadores de passagem

- **O indicador de passagem é diferente da variável de controle do laço**
  - A variável de controle **sempre** muda de valor **no mínimo** uma vez (na última repetição) para indicar o fim do laço
  - O indicador de passagem muda de valor **zero ou uma vezes** e a mudança pode acontecer em qualquer uma das iterações
    - » *(o comando que altera o valor do indicador de passagem pode ser executado mais de uma vez, mas o valor da variável só é efetivamente alterado na primeira)*
- **MAS...**
- **Às vezes podemos usar o indicador de passagem para terminar o laço antecipadamente**

Indicadores de passagem

Leia uma série de números terminada por zero fornecida pelo usuário e informe se os números estão em ordem crescente

```
ordenado = True
n = int(input("Digite um número (zero para sair): "))
anterior = n
while n != 0 and ordenado:
    if n < anterior:
        ordenado = False
    anterior = n
    n = int(input("Digite um número (zero para sair): "))
if ordenado:
    print("Os números estão em ordem")
else:
    print("Os números não estão em ordem")
```

Exercício

Dado um número $n > 0$, informe se esse número possui ao menos dois dígitos adjacentes que sejam iguais

```
n = int(input("Digite um inteiro positivo: "))
repetidos = False
tmp = n
anterior = tmp % 10
tmp //= 10
while tmp > 0 and not repetidos:
    atual = tmp % 10
    if anterior == atual:
        repetidos = True
    anterior = atual
    tmp //= 10
if repetidos:
    print("O número {} tem dígitos repetidos".format(n))
else:
    print("O número {} não tem dígitos repetidos".format(n))
```

Repetições encaixadas

- **Repetições usam uma variável de controle**
- **Quando essa variável corresponde a um conjunto, as repetições permitem trabalhar sobre os elementos desse conjunto**
  - De maneira unidimensional!
  - Números naturais, nomes de pessoas, notas de alunos...
- **Como trabalhar com conjuntos de maneira bidimensional?**
  - Pontos de um plano, tabelas...
- **Repetições encaixadas**

Repetições encaixadas

```
while condição1:
    while condição2:
        …
```

- A cada "rodada" do laço mais externo, o laço interno é executado várias vezes (até sua condição deixar de ser verdadeira)
- Para isso funcionar, `condição2` deve voltar a ser verdadeira a cada nova rodada do laço mais externo

Exemplo — tabuada

```
tabuada_do = 1
while tabuada_do <= 10:
    n = 1
    while n <= 10:
        print(tabuada_do * n , end=" ")
        n += 1
    tabuada_do += 1
    print()
```

Exemplo — tabuada

```
tabuada_do = 1
while tabuada_do <= 10:
    n = 1
    while n <= 10:
        print(tabuada_do * n, end="\t")
        n += 1
    tabuada_do += 1
    print()
```

Exemplo — tabuada

```
tabuada_do = 1
while tabuada_do <= 10:
    n = 1
    while n <= 10:
        print("{:3}".format(tabuada_do * n), end=" ")
        n += 1
    tabuada_do += 1
    print()
```

Exercício

Leia um número natural e faça a fatoração desse número em primos, indicando os fatores repetidos

```
n = int(input("Digite um inteiro positivo: "))
divisor = 2
while n > 1:
    while n % divisor > 0:
        divisor += 1
    print(divisor, end = " ")
    n /= divisor
print()
```

Exercício

Leia uma sequência de números naturais estritamente positivos terminada por zero e, para cada número, imprima seu fatorial

```
n = int(input("Digite o número: "))
while n != 0:
    fat = n
    i = 2 # número mágico
    while i <  n:
        fat *= i
        i += 1
    print("O fatorial de {} é {}".format(n, fat))
    n = int(input("Digite o número: "))
```