Universidade de São Paulo
Brasil

Prova 1

Introdução à Computação

Professor: Paulo Meirelles (paulormm@ime.usp.br)

Departamento de Ciência da Computação
Instituto de Matemática e Estatística

Q1. Simulação

```
n = 42
a = ??
while n >= a:
  b = n // 2
  if b * 2 == n:
   a = a + 3
  else:
   a = a + 2
  n = n - 7
  x = a % 9
  print(x * 7 + ??)
```

```
n = 42
a = 0
while n >= a:
  b = n // 2
  if b * 2 == n:
   a = a + 3
  else:
   a = a + 2
  n = n - 7
  x = a % 9
  print(x * 7 + 14)
```

Saídas

35 49 70 21 42

```
n = 42
a = 1
while n >= a:
  b = n // 2
  if b * 2 == n:
   a = a + 3
  else:
   a = a + 2
  n = n - 7
  x = a % 9
  print(x * 7 + 15)
```

Saídas

43 57 15 29 50

```
n = 42
a = 2
while n >= a:
  b = n // 2
  if b * 2 == n:
   a = a + 3
  else:
   a = a + 2
  n = n - 7
  x = a % 9
  print(x * 7 + 12)
```

Saídas

47 61 19 33 54

Q2.EP1

...

Este programa em Python simula o funcionamento de um caixa eletrônico que fornece um conjunto de notas correspondentes a um valor de saque desejado pelo usuário. O programa impõe algumas regras, como notas mínimas de R$10, um limite máximo de saque de R$1.000 e a restrição de notas de R$100 que só podem ser usadas para evitar mais de 4 notas de R$50.

Função main()

Inicializa a variável `válido` como `False`. Essa variável será usada para determinar se o valor de saque é válido de acordo com as regras estabelecidas.

Lê o valor de saque desejado (`solicitado`) fornecido pelo usuário.

O programa verifica algumas condições:

Se o valor de saque for maior que R$1.000, imprime "O maior saque permitido é R$1.000".

Se o valor de saque não for divisível por 10 (ou seja, não é um múltiplo de 10), imprime "A menor nota disponível é R$10".

Se nenhuma das condições acima for atendida, a variável `válido` é definida como `True`, indicando que o valor de saque é válido de acordo com as regras. Caso contrário, lê o valor desejado novamente.

```
def main():
  valido = False
  while not valido:
    solicitado = int(input("Digite o valor do saque: "))
    if solicitado > 1000:
      print("O maior saque permitido é R$1.000")
    elif solicitado % 10 != 0:
      print("A menor nota disponível é R$10")
    else:
      valido = True
```

Função main()

A variável `falta` é inicializada com o valor do saque `solicitado`. Ela rastreará o valor que ainda precisamos fornecer em notas.

Calcula o número de notas de R$50 (`notasCinquenta`) que devem ser usadas. Divide o valor restante `falta` por 50 e armazena o quociente em `notasCinquenta`. Em seguida, atualiza o valor de `falta` com o restante.

Inicializa a variável `notasCem` como 0. Essa variável rastreará o número de notas de R$100 que devem ser usadas.

```
def main():
  valido = False
  while not valido:
    solicitado = int(input("Digite o valor do saque: "))
    if solicitado > 1000:
      print("O maior saque permitido é R$1.000")
    elif solicitado % 10 != 0:
      print("A menor nota disponível é R$10")
    else:
      valido = True
  falta = solicitado
  notasCinquenta = falta // 50
  falta %= 50
  notasCem = 0
```

Função main()

Entra em um loop `while` para verificar se há mais de 4 notas de R$50 (`notasCinquenta > 4`). Se isso for verdade, o programa incrementa `notasCem` em 1 e decrementa `notasCinquenta` em 2. Isso garante que notas de R$100 só sejam usadas se houver pelo menos 5 notas de R$50 incluídas na saída. O loop continua até que essa condição não seja mais atendida.

Calcula o número de notas de R$10 (`notasDez`) que devem ser usadas para o valor restante `falta` dividindo-o por 10.

Imprime o resultado, informando ao usuário quantas notas de R$100, R$50 e R$10 serão fornecidas para o saque.

```
def main():
  válido = False
  while not válido:
    solicitado = int(input("Digite o valor do saque: "))
    if solicitado > 1000:
      print("O maior saque permitido é R$1.000")
    elif solicitado % 10 != 0:
      print("A menor nota disponível é R$10")
    else:
      válido = True
  falta = solicitado
  notasCinquenta = falta // 50
  falta %= 50
  notasCem = 0
  while notasCinquenta > 4:
     notasCem += 1
     notasCinquenta -= 2
  notasDez = falta // 10
  print("Você vai receber R${} com {} notas de 100, {} notas de 50 e {} notas
de dez".format(solicitado, notasCem, notasCinquenta, notasDez))
```

Q3. Horas de vida

Função bissexto(a)

Esta função é usada para determinar se um ano é bissexto ou não, com 366 ou 365 dias.

- A função recebe um argumento "a", que representa o ano que queremos verificar.

- A expressão "a % 4 == 0" verifica se o ano é divisível por 4, que é a condição "usual" para ser bissexto.

- A expressão "a % 100 != 0" verifica se o ano não é divisível por 100. Anos divisíveis por 100 não são bissextos, a menos que sejam divisíveis por 400 (próxima condição).

- A expressão "a % 400 == 0" verifica se o ano é divisível por 400, que é outra condição para ser bissexto.

Se a combinação dessas condições for verdadeira, a função devolve "True", indicando que o ano é bissexto; caso contrário, a função devolve "False".

Função TotalDiasPorMes(m, a)

Esta função é usada para calcular o número total de dias em um mês "m" de um ano "a". A quantidade de dias em um mês varia, e essa função leva em consideração se o ano é bissexto ou não. Aqui está como a função funciona:

- A função recebe dois argumentos: "m" representa o mês que queremos calcular e "a" representa o ano.

- A função começa verificando se o mês é fevereiro (representado por "m == 2") e se o ano é bissexto (usando a função "bissexto(a)"). Se for fevereiro e bissexto, o mês terá 29 dias. Se for fevereiro e não bissexto, o mês terá 28 dias.

Função TotalDiasPorMes(m, a)

- Se o mês não for fevereiro, a função verifica se o mês é par e está antes de julho ("(m % 2 == 0 and m < 7)") ou se o mês é ímpar e está depois de agosto ("(m % 2 != 0 and m > 8)"). Nesses casos, o mês terá 30 dias.

- Se nenhuma das condições acima for atendida, isso significa que o mês tem 31 dias.

- A função retorna o número de dias no mês calculado de acordo com as condições acima.

Função main()

Aqui começa o programa principal, onde o usuário fornecerá a data de nascimento e a data final para calcular o número de horas vividas.

- As variáveis "d1, m1, a1, h1" são usadas para armazenar o dia, mês, ano e hora do nascimento do usuário, respectivamente. O usuário é solicitado a inserir esses valores.

- As variáveis "d2, m2, a2, h2" são usadas para armazenar o dia, mês, ano e hora da data final fornecida pelo usuário.

- Inicializamos a variável "horas" como zero. Ela será usada para rastrear o total de horas vividas.

Função main()

O loop "while" é usado para calcular o número de horas vividas entre as datas de nascimento e a data final. O loop continuará enquanto a data de nascimento seja menor que a data final, o que significa que ainda não chegamos à data final.

Dentro do loop:

- Adicionamos 24 horas a "horas" a cada iteração, pois cada dia tem 24 horas.

- Incrementamos o valor do dia "d" em 1. Se o dia "d" ultrapassar o número total de dias em um mês, o mês "m" é incrementado em 1 e o dia é resetado para 1. Se o mês "m" ultrapassar 12, o ano "a" é incrementado em 1 e o mês é resetado para 1.

- O loop continuará até que a data “atual” ("d", "m", "a") seja igual à data final ("d2", "m2", "a2").

Função main()

Após o loop, a diferença nas horas ("h2 - h1") entre a hora de nascimento e a hora da data final é adicionada a "horas". Isso representa as horas vividas no último dia.

Por fim, o programa imprime a quantidade total de horas vividas, que está armazenada em "horas".

Q4. Arctan

...

Este programa em Python tem como objetivo calcular o valor do arco-tangente de vários ângulos x usando a aproximação da série de Taylor, além de calcular o erro absoluto da aproximação em relação ao valor real do arco-tangente.

A série de Taylor é uma representação matemática que descreve uma função como uma soma infinita de termos. Ela é uma ferramenta importante na matemática e na análise, usada para aproximar funções complicadas por meio de funções mais simples, especialmente em cálculo.

O código começa importando o módulo math, que será usado para calcular o valor real do arco-tangente com a função math.atan(x).

Função main()

Lê o número de testes `m` (quantos valores de `x` serão testados) e o número de termos `n` da série de Taylor a serem usados na aproximação.

Inicializa uma variável `i` para rastrear quantos testes já foram realizados.

Entra em um loop `while` que permite realizar múltiplos testes com valores diferentes de `x`.

Inicializa a variável `estimado` para armazenar a estimativa do arco-tangente.

Lê o valor de `x` do usuário, que deve estar no intervalo (`-1 < x < 1`), ou seja, valores válidos para o cálculo do arco-tangente.

Função main()

Inicializa a variável `sinal` como `-1`. Isso será usado para alternar o sinal dos termos na série de Taylor.

Inicializa a variável `j` para rastrear quantos termos da série de Taylor já foram usados.

Entra em um segundo loop `while` para calcular a série de Taylor com os `n` primeiros termos.

Alternadamente, inverte o sinal `sinal` de `-1` para `1` e vice-versa.

Função main()

Ainda dentro do loop, calcula o termo atual da série de Taylor usando a fórmula dada e armazena em `termo`.

Adiciona o `termo` à estimativa `estimado`.

Incrementa `j` para passar para o próximo termo.

Função main()

Depois de calcular a estimativa da série de Taylor, calcula o valor real do arco-tangente usando `math.atan()` e armazena em `esperado`.

Calcula o erro absoluto da estimativa em relação ao valor real e armazena em `erro`.

Incrementa `i` para passar para o próximo teste.

Imprime os valores `esperado`, `estimado` e `erro` para cada teste realizado.

O loop continua até que todos os `m` testes sejam concluídos

MAC
Introdução à Computação

- Professor: Paulo Meirelles
  E-mail: paulormm@ime.usp.br