TÉCNICO LISBOA

Fundamentos da Programação

Solução do Primeiro Teste

29 de Março de 2014

09:00–10:30

1. (1.0) O que é um processo computacional? Qual a relação entre um programa e um processo computacional?

   Resposta: Um processo computacional é um ente imaterial que existe dentro de um computador durante a execução de um programa, e cuja evolução ao longo do tempo é ditada pelo programa.

2. Considere a seguinte gramática em notação BNF:

```
<operação> ::= (<argumento> <operador> <argumento>)
<operador> ::= + | - | * | /
<argumento> ::= <dígito>+
<dígito> ::= 2 | 4 | 6 | 8 | 0
```

   a. (0.5) Indique os símbolos terminais e os símbolos não terminais da gramática.

   Resposta: ST: (,), +, -, *, /, 2, 4, 6, 8, 0. SNT: operação, operador, argumento, dígito.

   b. (0.5) Indique, justificando, quais das expressões seguintes pertencem ou não pertencem ao conjunto de operações da linguagem definida pela gramática.

```
(1 + 2)
(2 + -)
(24 * 06)
2 * 0
(84 + )
(0 / 0)
```

   Resposta:

```
(1 + 2)         sim
(2 + -)         não, tem dois operadores seguidos
(24 * 06)       sim
2 * 0           não, falta ( no inicio e ) no fim
(84 + )         não, falta um argumento entre o operador + e o )
(0 / 0)         sim
```

3. A abstracção procedimental foi descrita como um mecanismo utilizado para dominar a complexidade de programas.

   a. (0.7) Diga em que consiste a abstracção procedimental.

   Resposta: A abstracção procedimental corresponde a abstrair o modo como uma função realiza o seu trabalho, considerando apenas o que ela faz.

   b. (0.8) Explique como esta pode ser usada no controle da complexidade de programas.

   Resposta: Ao desenvolver um programa, identificam-se os principais problemas que este temque resolver, especificando-se funções que realizam esse trabalho e sem entrar nos detalhes do modo como elas realizam o seu trabalho. Depois de escrita uma primeira versão do programa recorrendo à abstracção

procedimental, aborda-se o desenvolvimento de cada uma das funções especificadas utilizando o mesmo método.

c. (0.5) Como é que a abstracção procedimental é realizada em Python?

Resposta: Através da definição de funções que recebem os argumentos apropriados.

4. Os métodos de passagem de parâmetros correspondem a modos de associar os parâmetros concretos com os parâmetros formais.

   a. (0.6) Diga o que são os parâmetros formais e o que são os parâmetros concretos.

   Resposta: Os parâmetros formais são os argumentos especificados na definição de uma função e os parâmetros concretos são os valores que são usados na invocação de uma função.

   b. (0.7) Explique o funcionamento da passagem por valor.

   Resposta: Na passagem por valor, o parâmetro concreto é avaliado e o seu valor é associado com o respectivo parâmetro formal. A passagem por valor é um mecanismo unidireccional, do ponto de chamada para a função.

   c. (0.7) Explique o funcionamento da passagem por referência.

   Resposta: Na passagem por referência a localização de memória da entidade correspondente ao parâmetro concreto é fornecida ao parâmetro formal. Na passagem por referência o parâmetro concreto e o parâmetro formal partilham a mesma entidade na memória do computador.

5. (1.0) Considere o seguinte programa em Python:

```
numero_1 = 5
numero_2 = 10
while numero_1 > 0:
   numero_2 = numero_2 - numero_1
```

   Será que este programa pode ser considerado um algoritmo? Justifique a sua resposta.

   Resposta: Este programa não pode ser considerado um algoritmo porque nunca termina. Na reali dade a condição numero_1 > 0 é sempre verdadeira, dado que numero_1 tem o valor 5 e o corpo do ciclo while não altera este valor.

6. (1.0) Escreva um programa em Python que pede ao utilizador que lhe forneça um inteiro correspondente a um certo número de dias e que escreve um número real que traduz o número de anos correspondentes ao inteiro lido. Considere que cada ano tem 365.25 dias. O seu programa deve gerar uma interacção como a seguinte:

```
Eu converto dias para anos
Escreva os dias
Dias? 2568
2568 dias correspondem a 7.030800821355236 anos
```

   Resposta:

```
print('Eu converto dias para anos')
dias = eval(input('Escreva os dias\nDias? '))
print(dias, ' dias correspondem a ', dias/365.25, ' anos')
```

7. (1.0) Diga o que é escrito pela seguinte instrução:

```
for i in range(2):
```

```
for j in range(3, 5):
   for k in range(4, 1, -1):
      if (i + j) % 2 == 0:
         print(i, j, k)
```

Resposta:

```
0 4 4
0 4 3
0 4 2
1 3 4
1 3 3
1 3 2
```

8. (1.5) Escreva uma função em Python com o nome `soma_digitos_pares` que recebe um inteiro positivo, n, e devolve a soma dos dígitos pares de n. Por exemplo:

```
>>> soma_digitos_pares (135)
0
>>> soma_digitos_pares (1249)
6
```

Resposta:

```
def soma_digitos_pares (n):
   soma = 0
   while n > 0:
      if n%2==0:
         soma = soma + n%10
      n = n // 10
   return soma
```

9. (1.5) Escreva uma função `numero_ocorrencias_tuplo_numeros` que recebe um tuplo de números e um número, e devolve o número de vezes que o número ocorre no tuplo.

```
>>> numero_ocorrencias_tuplo_numeros((4, 5, 6), 5)
1
>>> numero_ocorrencias_tuplo_numeros((3, 5, 7), 2)
0
>>> numero_ocorrencias_tuplo_numeros((3, 5, 3), 3)
2
```

Resposta:

```
def numero_ocorrencias_tuplo_numeros(tuplo, num):
   count = 0
   for el in tuplo:
      if el == num:
         count = count + 1
   return count
```

10. (1.5) Escreva uma função `obtem_ultima_posicao_n` que recebe um tuplo de números e um número, e devolve a última posição em que o número ocorre no tuplo. Se o número não existir no tuplo deve devolver `False`.

```
>>> obtem_ultima_posicao_n ((4, 5, 6, 5), 22)
False
>>> obtem_ultima_posicao_n ((4, 5, 6, 5), 5)
3
```

Resposta:

```
def obtem_ultima_posicao_n (t, n):
   for i in range (-1,-len(t),-1):
      if t[i] == n:
         return i + len(t)
   return False
```

11. (1.5) Escreva uma função `concatena_n_strings_lista` que recebe uma lista de cadeias de caracteres e um inteiro n. Cada n cadeias de caracteres devem ser agrupadas numa nova cadeia de caracteres. A função deve devolver uma nova lista com as novas cadeias de caracteres. A sua função não tem que validar os argumentos.

```
>>> concatena_n_strings_lista (['a','b','c','d','e','f'],2)
['ab', 'cd', 'ef']
>>> concatena_n_strings_lista (['a','b','c','d','e','f'],4)
['abcd', 'ef']
```

Resposta:

```
def concatena_n_strings_lista (lst, n):
    res = []
    cc = ''
    i = 0
    for ele in lst:
        cc = cc + ele
        i = i + 1
        if i == n:
            res = res + [cc]
            cc = ''
            i = 0
    if cc != '':
        res = res + [cc]
    return res
```

12. Dada uma lista:

    a. (1.5) Escreva uma função em Python chamada `e_lista_numeros` que devolve `True` se a lista for uma lista não vazia de números ou `False` no caso contrário.

Resposta:

```
def e_lista_numeros(lst):
    if not(isinstance(lst, list)) or lst == []:
        return False
    else:
        for el in lst:
            if not(isinstance(el, (int, float))):
                return False
        return True
```

    b. (1.5) Escreva uma função em Python chamada `altera_pares` que recebe uma lista de números e modifica os números pares dessa lista adicionando-lhes uma unidade. A sua função deve testar se o argumento é uma lista não vazia de números e produzir uma mensagem de erro adequada em caso contrário. Por exemplo,

```
>>> lst = [3, 5, 6, 2, 7, 10, 2]
>>> altera_pares(lst)
>>> lst
[3, 5, 7, 3, 7, 11, 3]
>>> altera_pares ([])
builtins.ValueError: altera_pares: arg não é lista de números
```

Resposta:

```
def altera_pares (lst):
    if not e_lista_numeros(lst):
        raise ValueError ('altera_pares: arg não é lista de números')
    else:
        for i in range(len(lst)):
            if lst[i]%2==0:
                lst[i] = lst[i] + 1
```

13. (2.0) Escreva uma uma função em Python chamada `ordena_borbulhamento` que recebe uma lista de números e a ordena usando o algoritmo de ordenação por borbulhamento. Assuma que a lista passada à função está correcta. Por exemplo,

```
>>> lst = [23, 5, 6, 12, 7, 10, 2]
>>> ordena_borbulhamento (lst)
>>> lst
[2, 5, 6, 7, 10, 12, 23]
```

Resposta:

```
def ordena_borbulhamento (lst) :
    maior_indice = len(lst) - 1
    nenhuma_troca = False
    while not nenhuma_troca :
        nenhuma_troca = True
        for i in range(maior_indice) :
            if lst[i] > lst[i+1] :
                lst[i], lst[i+1] = lst[i+1], lst[i]
                nenhuma_troca = False
        maior_indice = maior_indice - 1
```