Instituto Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

Orientação Tutorial
Algoritmos e Programação em Python

9 a 13 de Outubro

Entrada e validação de dados
Estruturas condicionais
Estruturas de repetição
Traçagem de algoritmos
Desafios

Curso: Engenharia Informática
Unidade Curricular: Algoritmos e Estruturas de Dados
Ano Letivo: 2017/2018
Docente: Paulo Jorge Costa Nunes
Coordenador da área disciplinar: Noel Lopes

Conteúdo

Capítulo 1

Traçagem de algoritmos

Provar ou afirmar que um algoritmo está correto é um dos passos mais difícil, e muitas vezes tedioso, do desenvolvimento de algoritmos [1].

Em muitos casos o procedimento seguido para afirmar que um algoritmo está correto consiste em executar o algoritmo para uma variedade de casos de teste que se considera que cobre todas os valores e combinações possíveis de dados. Os resultados obtidos são então comparados com resultados conhecidos ou calculados manualmente. Quando este são iguais somos tentados a dizer que o algoritmo funciona. Contudo, esta técnica apenas para algoritmos extremamente simples permite retirar todas as dúvidas de que possam existir casos de teste em que o algoritmo falha. Para algoritmos mais complexos, raramente se pode afirmar que o algoritmo está correto. Pode-se apenas dizer que funciona para os casos de teste.

1.0.1 Traçagem do algoritmo

Para se obterem resultados para casos de teste pode-se proceder a uma traçagem do algoritmo que consiste em simular a execução do algoritmo para cada caso de teste e registar a alteração de cada uma das variáveis do algoritmo. No final de cada uma das simulações podemos obter os resultados do algoritmo lendo o valor das suas variáveis de saída. A traçagem de algoritmos permite descobrir erros. Contudo, não permite comprovar a sua ausência.

O processo de simulação é iterativo, isto é, linha a linha de código, desta forma torna-se útil atribuir um número a cada linha de código. Muitas vezes para reduzir o número de passos atribui-se um número a um grupo de linhas de código.

Para facilitar o registo da evolução dos valores das variáveis e saída recorre-se a uma matriz onde se regista a evolução das variáveis e todas as operações auxiliares necessárias ao seu cálculo, tais como operações aritméticas, condições e repetições.

Nas colunas da matriz são colocadas todas as variáveis todas as condições e saídas. Nas linhas vão-se registando os passos do algoritmo que são executados durante a simulação.

1.0.2 Exemplo

1.0.2.1 Passos

Número	Instruções
P00	BEGIN
P01	Boolean B1, B2, B3;
P02	If(B1) Then
P03	C1;
P04	Else
P05	If(B2) Then
P06	If(B3) Then
P07	C2;
P08	Else
P09	C3;
P10	C4;
P11	End IF
P12	End IF
P13	C5;
P14	End IF
P15	END

Figura 1.1

1.0.2.2 Tabela de traçagem

Passo	B1(False)	B2(True)	B3(True)	B1	B2	B3	Passo Seg.	Saída (Ecrã)
P00								
P01	False	True	True					
P02	-	-	-	**True**			**P05**	
P05	-	-	-		**True**		**P06**	
P06	-	-	-			**True**	**P07**	
P15	-	-	-					Resultado da execução de **C2**

Capítulo 2

Python

1. if — https://docs.python.org/3/tutorial/controlflow.html#if-statements
2. else
3. elif
4. while — https://docs.python.org/3/tutorial/controlflow.html#if-statements
5. range — https://docs.python.org/3/tutorial/controlflow.html#the-range-function
6. in –
7. continue — https://docs.python.org/3/reference/simple_stmts.html#continue — https://docs.python.org/3/tutorial/controlflow.html#if-statements
8. break — https://docs.python.org/3/reference/simple_stmts.html#break
9. True
10. not
11. or
12. and
13. try — https://docs.python.org/3/reference/compound_stmts.html#try
14. except — https://docs.python.org/3/reference/compound_stmts.html#except

Capítulo 3

Implementar em Python os exercícios da aula TP Nº3

4. Write an algorithm to check whether a number is even or odd.
5. Write an algorithm to make the sum of several positive integers entered via the keyboard. The input of values ends when a negative value is entered.
6. Write an algorithm to calculate the arithmetic mean of the grades of 5 subjects.
7. Write an algorithm to calculate the arithmetic mean of the grades of 5 subjects.

Capítulo 4

Entrada e validação de dados

4.1 Entrada de número inteiros

4.1.1 Número maior/menor do que um dados valor

```
number = -1        # start with an illegal value
while number < 0: # to get into the loop
  number = float(input("Enter a positive number: "))
print (number)
```

Listing 4.1: Post-Test Loop

```
while True:
  number = float(input("Enter a positive number: "))
  if number >= 0:
    break # Exit loop if number is valid
print (number)
```

Listing 4.2: Post-Test Loop: Python break

```
# In the while loop version, this is awkward:
number = -1
while number < 0:
  number = float(input("Enter a positive number: "))
  if number < 0: # We?re doing the validity check in two places!
    print("The number you entered was not positive")
print (number)
```

Listing 4.3: Post-Test Loop: Python break, warning

```
while True:
  number = int(input("Enter a positive number: "))
  if number >= 0:
    break # Exit loop if number is valid
  else:
    print("The number you entered was not positive.")
print (number)
```

Listing 4.4: Post-Test Loop: Python break, else, warning

```
while True:
  number = int(input("Enter a positive number: "))
  if number >= 0:
    break # Loop exit
  print("The number you entered was not positive")
print (number)
```

Listing 4.5: Post-Test Loop: Python break, warning

4.1.2 Números reais num intervalo

```
while True:
  number = float(input("Enter a number in the range [10, 20]: "))
  if number >= 10 and number <= 20:
    break # Exit loop if number is valid
  print("The number you entered was not in the range.")
print (number)
```

Listing 4.6: Post-Test Loop: Python break, warning

4.1.3 Números inteiro num intervalo

```
while True:
  number = int(input("Enter a number in the range [10, 20]: "))
  if number in range(10,21):
    break # Exit loop if number is valid
  print("The number you entered was not in the range.")
print (number)
```

Listing 4.7: Post-Test Loop: Python break, range, warning

4.1.4 Números inteiro de uma lista

```
options = [1, 5, 7, 8]
while True:
  number = int(input("Enter a number in the set {1, 5, 7, 8}: "))
  if number in options:
    break # Exit loop if number is valid
  print("The number you entered was not in the set.")
print (number)
```

Listing 4.8: Post-Test Loop: Python break, in warning

4.1.5 Números inteiro num intervalo com tratamento de erros

```
while True:
    try:
        number = int(input("Enter a number in the range [10, 20]: "))
    except ValueError:
        print("Sorry, I didn't understand that.")
    if number not in range(10,21):
      print("Sorry, your response is not be in the range.")
      continue
    else:
        break
print (number)
```

Listing 4.9: Post-Test Loop: Python break, range, except, else, continue, warning

4.2 Desafio / Challenge

Desenvolva programas para validar os seguintes tipos de dados:

1. Peso de uma pessoa
2. Idade de uma pessoa
3. Género de uma pessoa (masculino, feminino)
4. Número de telemóvel
5. Numero de telefone
6. Endereço de correio eletrônico
7. Data de nascimento
8. Data de validade
9. Hora de partida

Bibliografia

[1] S.E. Goodman and S.T. Hedetniemi. *Introduction to the design and analysis of algorithms.* McGraw-Hill computer science series. McGraw-Hill, 1977.