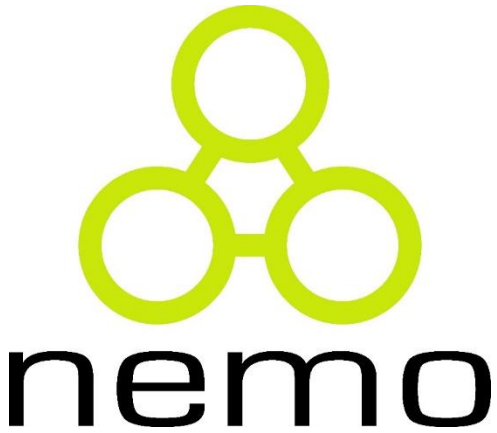


Programação I

Jordana S. Salamon

jssalamon@inf.ufes.br

jordanasalamon@gmail.com

DEPARTAMENTO DE INFORMÁTICA

CENTRO TECNOLÓGICO

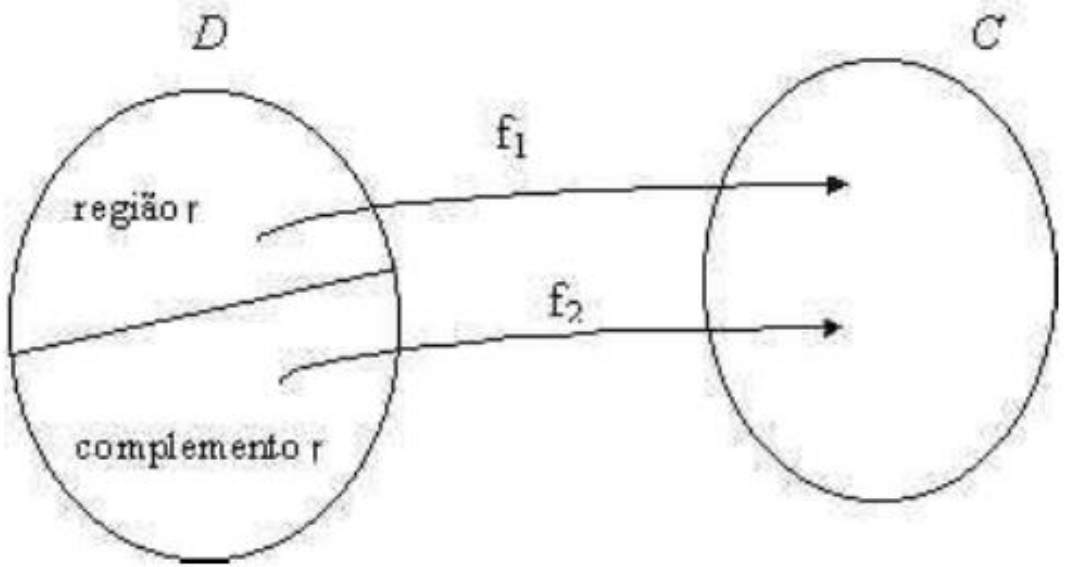
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Definições Condicionais

REVISÃO

- Sabemos de nosso conhecimento matemático que algumas funções não são contínuas em um domínio e que, portanto, possuem várias definições.

- Em muitos casos, o domínio D de uma função está dividido em regiões disjuntas que se complementam e, para cada uma dessas regiões, existe uma expressão que define o seu mapeamento no contra-domínio. Podemos representar esta situação pela figura abaixo:



nemo

Definições Condicionais

REVISÃO

- Vamos estudar inicialmente a construção mais simples de definição condicional, construída com a estrutura **if-else** segundo a seguinte sintaxe:

```
if <expressão lógica>: <expressão 1>
else: <expressão 2>
```

- Onde:

<expressão lógica>	Uma expressão descrevendo uma condição a ser satisfeita, envolvendo operadores relacionais e operadores lógicos.
<expressão1> e <expressão2>	1. Expressões descrevendo um valor a ser produzido como resposta à entrada fornecida e, como a expressão total deve ser de um único tipo, as duas expressões devem ser do mesmo tipo. 2. Cada uma destas expressões pode ser inclusive outra condicional, dentro da qual pode haver outras e assim sucessivamente. 3. Quando a <expressão lógica> é avaliada para ***True*** o valor resultante será o que for obtido pela avaliação da <expressão 1> caso contrário será o obtido pela avaliação da <expressão 2>

nemo

Definições Condicionais

REVISÃO

- Para a função que calcula o valor da passagem aérea podemos então construir a seguinte definição:

```
>>> def valorPassagem(x):
        if x < 60: return 600
        else: return 360
```

nemo

Definições Condicionais

REVISÃO

- Contudo, alguns problemas apresentam uma definição condicional mais complexa, de forma que o problema se divide em mais de dois subdomínios.

- Nestes casos, a estrutura **if-elif-else**, complementar à recém-apresenta, deve ser usada segundo a seguinte sintaxe:

```
if <expressão lógica 1>: <expressão 1>
elif <expressão lógica 2>: <expressão 2>
...
elif <expressão lógica n-1>: <expressão n-1>
else: <expressão n>
```

nemo

Definições Condicionais

REVISÃO

- Este é o caso do exemplo 2 que define a função para cálculo do Imposto de Renda.

- O domínio neste caso deve ser quebrado em quatro subdomínios e para cada um deles construiremos uma expressão.

Domínio	Expressão
$s \leq 10800$	0
$10800 < s \leq 20000$	s * 0.1 - 1000
$20000 < s \leq 30000$	s * 0.2 - 1500
$s > 30000$	s * 0.25 - 1800

nemo

Definições Condicionais

REVISÃO

- Para a codificação, precisaremos quebrar sucessivamente o domínio da função em intervalos.
- A codificação final pode ser:

```
>>> def imposto(s):
        if s <= 10800: return 0
        elif s <= 20000: return s * 0.1 - 1000
        elif s <= 30000: return s * 0.2 - 1500
        else: return s * 0.25 - 1800
```

nemo

Teste de Programas

- Não basta desenvolver um programa para resolver um dado problema. É preciso garantir que a solução esteja correta.
- Muitos erros podem ocorrer durante o desenvolvimento de um programa e, portanto temos que garantir que o programa que irá ser executado está livre de todos eles.
- Esses erros podem ocorrer por um mau entendimento dos elementos da linguagem ou até mesmo por descuido, o certo é que eles ocorrem. Uma estratégia muito útil, mas não infalível, é o teste de programa.
- Em sua essência, o teste de programa consiste em submeter um programa ao exercício de algumas instâncias do problema e comparar os resultados esperados com os resultados obtidos.

nemo

O Processo de Teste

- Em primeiro lugar devemos escolher as instâncias apropriadas, não basta escolhê-las aleatoriamente.
- A seguir devemos determinar, sem o uso do programa, qual o valor que deveria resultar quando o programa for alimentado com essas instâncias.
- O passo seguinte consiste em submeter cada instância ao programa e anotar o resultado produzido por ele.
- Finalmente devemos comparar cada valor esperado com o valor produzido e descrever qual o tipo de ocorrência.

nemo

O Processo de Teste

- Exemplo:

- Considere o problema de identificar se um dado ponto está ou não localizado no primeiro quadrante do espaço cartesiano. Considere ainda a seguinte definição:

```
>>> def primQuad(x, y): return (x >= 0) and (y >= 0)
```

- Precisamos agora verificar se ela atende nossa intenção. Para tanto devemos escolher algumas instâncias, prever o resultado esperado e em seguida submeter ao interpretador, para ver o que acontece.

- Que pares de valores devemos escolher?

O Processo de Teste

- Podemos escolher uns pares usando a seguinte estratégia: um par onde x é maior que y, outro onde y seja maior que x e um terceiro em que os dois sejam iguais.

x	y	resultado esperado	resultado obtido	diagnóstico
-5	-2	False		
-2	5	False		
5	5	True		

nemo

O Processo de Teste

- Podemos agora submeter as instâncias à avaliação do sistema, obtendo a seguinte interação:

- >>> primQuad(-5, -2)
- False
- >>> primQuad(-2, 5)
- False
- >>> primQuad(5, 2)
- True

O Processo de Teste

- Completando a tabela:

x	y	resultado esperado	resultado obtido	diagnóstico
-5	-2	False	False	sucesso
-2	5	False	False	sucesso
5	2	True	True	sucesso

- Será que podemos dizer que nosso programa está correto?

O Processo de Teste

- Apesar de passar em todos os testes a que foi submetido, ele não funciona corretamente.

- Tudo que podemos afirmar neste instante é que para os valores usados, o programa funciona corretamente. E para os outros valores, será que funciona corretamente? Outros valores? Quais?

- Antes de mais nada, devemos identificar as classes de valores que serão relevantes para o teste, e em um segundo instante podemos então escolher os representantes destas classes.

O Processo de Teste

- Quando temos um parâmetro, os possíveis valores a serem usados são todas as constantes do domínio.
- Para o caso de um parâmetro do tipo int, existem 65536 valores diferentes. Será que precisamos testar nosso programa para todos esses valores?
- Felizmente não precisamos de todos eles, basta identificar as classes distintas que importam para o problema, ou seja, as classes de equivalência relevantes.

O Teste de Programas

- Isto pode ser obtido analisando o problema:

x	**y**	**quadrante**
positivo	positivo	primeiro
negativo	positivo	segundo
negativo	negativo	terceiro
positivo	negativo	quarto
nulo	qualquer não nulo	eixo das ordenadas
qualquer não nulo	nulo	eixo das abscissas
nulo	nulo	origem

O Teste de Programas

- Voltando ao nosso exemplo, podemos agora elaborar a nossa planilha de teste considerando as classes de equivalência a serem definidas. Uma questão que surge é como escolhemos o representante de uma classe? Existem melhores e piores?

x	y	resultado esperado	resultado obtido	diagnóstico
2	3	True		
-2	3	False		
-2	-3	False		
2	-3	False		
0	3	False		
0	-3	False		
2	0	False		
-2	0	False		
0	0	False		

nemo

O Teste de Programas

- >>> primQuad(2, 3)
- True
- >>> primQuad(-2, 3)
- False
- >>> primQuad(-2, -3)
- False
- >>> primQuad(2, -3)
- False
- >>> primQuad(0, 3)
- True
- >>> primQuad(0, -3)
- False
- >>> primQuad(2, 0)
- True
- >>> primQuad(-2, 0)
- False
- >>> primQuad(0, 0)
- True

O Teste de Programas

x	y	resultado esperado	resultado obtido	diagnóstico
2	3	True	True	sucesso
-2	3	False	False	sucesso
-2	-3	False	False	sucesso
2	-3	False	False	sucesso
0	3	False	True	falha
0	-3	False	False	sucesso
2	0	False	True	falha
-2	0	False	False	sucesso
0	0	False	True	falha

O Teste de Programas

- Depurando nossa solução - Podemos concluir por simples inspeção da nossa última planilha que nossa solução está incorreta.

- Uma interpretação dos resultados nos leva à hipótese de que a nossa solução considera que quando o ponto se localiza na origem ou em um dos eixos positivos, a nossa definição está considerando que eles estão no primeiro quadrante.

- Passo seguinte, verificar se de fato nossa definição incorre neste erro. Em caso afirmativo, corrigi-la e a seguir, submetê-la novamente aos testes.

nemo

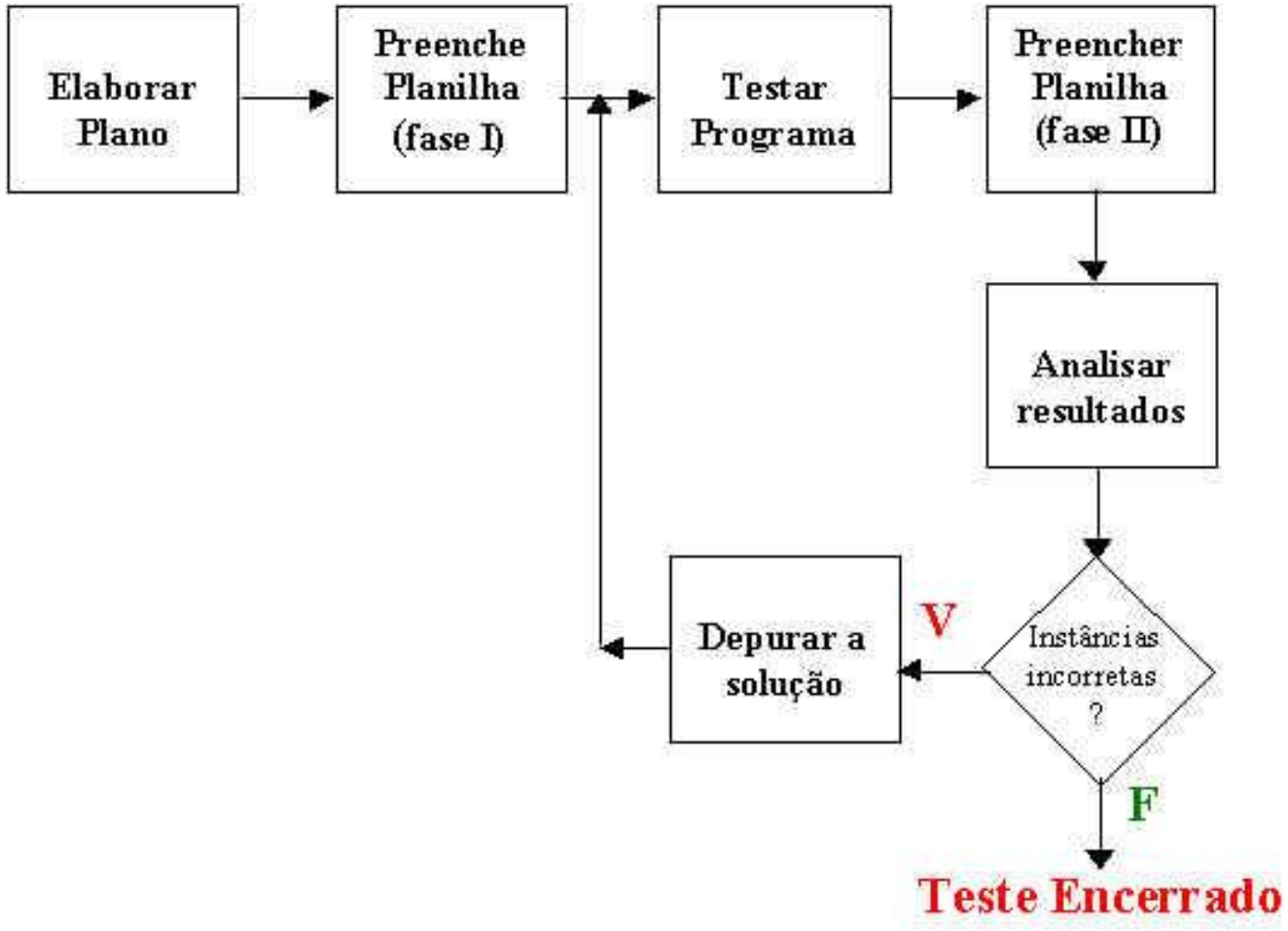
O Teste de Programas

- Observando a nossa definição inicial, podemos concluir que de fato nossa hipótese se confirma.
- >>> **def** primQuad(x, y): **return** (x >= 0) and (y >= 0)

- Podemos então modificá-la para obter uma nova definição, que esperamos que esteja correta.

- >>> **def** primQuad(x, y): **return** (x > 0) and (y > 0)

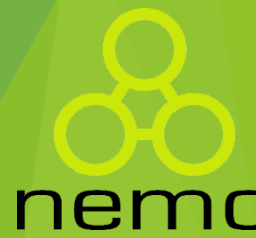
O Teste de Programas



Exercícios

- Para todos os exercícios abaixo, além de desenvolver a solução, monte um plano de testes.

- 1) Dado um número inteiro, verifique se ele pertence ao intervalo (0,100) e é divisível por 3 e por 5.
- 2) Dados três comprimentos de lados l1, l2 e l3, verifique se podem formar um triângulo.
- 3) Dados três números inteiros distintos, calcule o quadrado do sucessor do maior número.



Exercícios

- 4) Uma empresa decidiu dar a seus funcionários um abono de natal. A gratificação será baseada em dois critérios: o número de horas extras trabalhadas e o número de horas que o empregado faltou ao trabalho. O critério estabelecido para calcular o prêmio é: subtrair dois terços das horas que o empregado faltou de suas horas extras, obtendo um valor que determina o número de pontos do funcionário. Faça uma função para calcular o abono de natal para cada funcionário. A distribuição do prêmio é feita de acordo com a tabela abaixo:

Pontos obtidos	Prêmio em R$
1 a 10	100,00
11 a 20	200,00
21 a 30	300,00
31 a 40	400,00
A partir de 41	500,00

nemo

Exercícios

- 5) Considere que o preço de uma passagem de avião em um trecho pode variar dependendo da idade do passageiro. Pessoas com 60 anos ou mais pagam apenas 60% do preço total. Crianças até 10 anos pagam 50% e bebês (abaixo de 2 anos) pagam apenas 10%. Faça uma função que tenha como entrada o valor total da passagem e a idade do passageiro e produz o valor a ser pago.

Exercícios

- 6) Dada a idade de uma pessoa em segundos e um planeta do sistema solar, calcule qual seria a idade relativa dessa pessoa no planeta informado, sabendo que o Período Orbital é o intervalo de tempo que o planeta leva para executar uma órbita em torno do Sol (o que é denominado de ano, que na Terra tem aproximadamente 365,25 dias). Considere então as informações abaixo para outros planetas do sistema solar:

- Terra-> período orbital: 365.25 dias na Terra, ou 31557600 segundos
- Mercúrio-> período orbital: 0.2408467 anos na Terra
- Vênus-> período orbital: 0.61519726 anos na Terra
- Marte-> período orbital: 1.8808158 anos na Terra
- Júpiter-> período orbital: 11.862615 anos na Terra
- Saturno-> período orbital: 29.447498 anos na Terra
- Urano-> período orbital: 84.016846 anos na Terra
- Netuno-> período orbital: 164.79132 anos na Terra

nemo

That’s all Folks!