Tópicos especiais - Trabalho Prático 1

Aluno: Helder Melik Schramm

Programa utilizado

O programa utilizado foi baseado no exercício 1160 do URI Online Judge. A função “crescimento_populacional()” serve para calcular quantos anos demorará para que a população da cidade A ultrapasse a da cidade B, usando como base quatro entradas: dois inteiros PA e PB indicando respectivamente a população de A e B, e dois valores G1 e G2 com um dígito após o ponto decimal cada, indicando respectivamente o crescimento populacional de A e B (em percentual).

Algumas regras devem ser seguidas para que a função seja executada conforme o esperado:

- A população de A (PA) tem tamanho mínimo de 100 cidadãos e não pode exceder a quantidade de 1000000.
- A população de B (PB) também não pode exceder 1000000 de cidadãos, mas deve ser maior que PA.
- O crescimento populacional de A (G1) deve ser no mínimo 0.1 e não deve exceder 10.0.
- O crescimento populacional de B (G2) deve ser no mínimo 0.0 e não deve exceder 10.0.
- O valor de G2 deve ser menor que o valor de G1.

O programa retorna o número de anos que levará para que a cidade A ultrapasse a cidade B em número de habitantes, seguido da string “ anos”. Entretanto, caso o tempo for mais do que 100 anos, o programa deve retornar a mensagem “Mais de 1 século”.

Testes funcionais

Partição de equivalência

- População da cidade A (PA) -> PA < 100: inválido; 100 ≤ PA ≤ 1000000: válido; PA > 1000000: inválido
- População da cidade B (PB) -> PB ≤ PA: inválido; PA < PB ≤ 1000000: válido; PB > 1000000: inválido
- Crescimento populacional da cidade A (G1) -> G1 < 0.1: inválido; 0.1 ≤ G1 ≤ 10.0: válido; G1 > 10.0: inválido
- Crescimento populacional da cidade B (G2) -> G2 < 0.0: inválido; 0.0 ≤ G2 ≤ 10.0: válido; G2 > 10.0: inválido

Roteiro de testes (partição de equivalência)					
Caso de teste	PA	PB	G1	G2	Saídas esperadas
CT1	**50**	500	10.0	5.0	Inválido
CT2	**1000500**	**1000555**	10.0	5.0	Inválido
CT3	500	**200**	10.0	5.0	Inválido
CT4	500	**1000500**	10.0	5.0	Inválido
CT5	100	300	**-2.5**	**-5.0**	Inválido
CT6	100	300	**15.5**	5.0	Inválido
CT7	100	300	2.5	**-2.5**	Inválido
CT8	100	300	**16.0**	**15.5**	Inválido
CT9	100	300	5.0	**10.0**	Inválido

Anotações:

- A cor vermelha indica entradas incorretas que levam à saída “Inválido”;
- CT2 retorna “inválido”, pois PA é maior que 1000000. Como PB deve ser maior que PA, PB acaba ultrapassando 1000000 também;
- CT5 retorna “inválido”, pois G1 é menor que 0.1. Como G2 deve ser menor que G1, G2 é obrigado a ser menor que o seu limite mínimo de 0.0;
- CT8 retorna “inválido”, pois G2 é maior que 10.0. Como G2 deve ser menor que G1, G1 também é obrigado a ultrapassar o seu limite máximo de 10.0.

Análise de limite

- População da cidade A (PA) -> PA < 100: inválido; 100 ≤ PA ≤ 1000000: válido; PA > 1000000: inválido
- População da cidade B (PB) -> PB ≤ PA: inválido; PA < PB ≤ 1000000: válido; PB > 1000000: inválido
- Crescimento populacional da cidade A (G1) -> G1 < 0.1: inválido; 0.1 ≤ G1 ≤ 10.0: válido; G1 > 10.0: inválido
- Crescimento populacional da cidade B (G2) -> G2 < 0.0: inválido; 0.0 ≤ G2 ≤ 10.0: válido; G2 > 10.0: inválido

Roteiro de testes (análise do valor limite)					
Casos de Teste	Entradas				Saídas esperadas
	PA	PB	G1	G2	
CT1	**99**	600	5.0	2.5	Inválido
CT2	**100**	**101**	5.0	2.5	1 anos
CT3	**1000000**	1000001	5.0	2.5	Inválido
CT4	**1000001**	1000002	5.0	2.5	Inválido
CT5	500	**500**	5.0	2.5	Inválido
CT6	500	**1000000**	5.0	2.5	Mais de um século
CT7	500	**1000001**	5.0	2.5	Inválido
CT8	500	600	**0.0**	**-0.1**	Inválido
CT9	500	600	**0.1**	**0.0**	Mais de um século
CT10	500	600	**10.0**	**9.9**	Mais de um século
CT11	500	600	**10.1**	**10.0**	Inválido
CT12	500	600	5.0	**-0.1**	Inválido
CT13	500	600	9.0	**10.1**	Inválido
CT14	500	600	5.0	**5.0**	Inválido

Anotações:

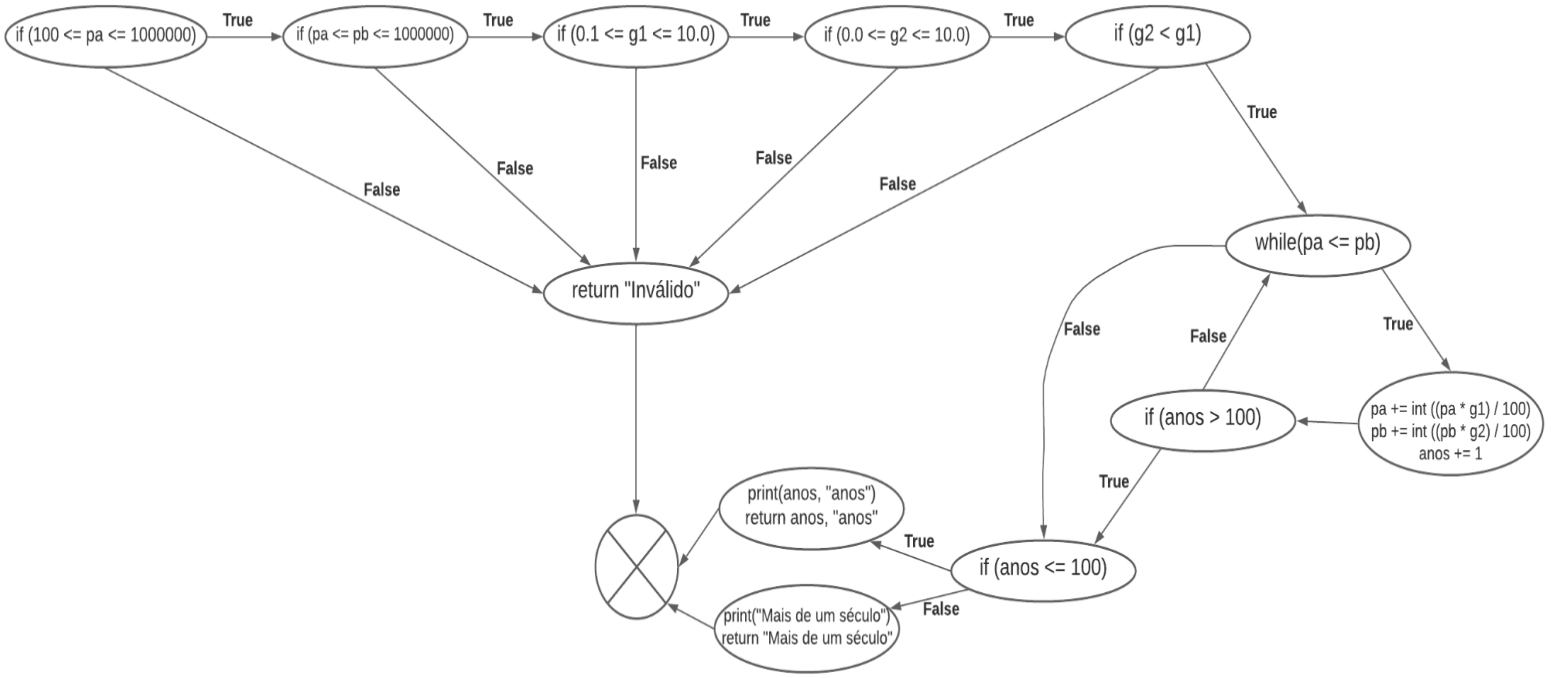
- As entradas em negrito e sublinhadas indicam qual entrada está sendo verificada no caso de teste;
- A cor vermelha indica entradas incorretas que levam à saída “Inválido”;
- CT2 testa tanto PA ≥ 100 quanto PA < PB;
- CT8 e CT9 testam G1 ≥ 0.1 e G2 ≥ 0.0;
- CT10 tanto G1 ≤ 10.0 quanto G2 < G1 estão sendo testados;
- CT11 testa G1 ≤ 10.0 e G2 ≤ 10.0.

Testes estruturais

Este é o código da função "crescimento_populacional()", que foi implementada em Python:

```python
def crescimento_populacional(pa, pb, g1, g2):
    pa = int(pa)
    pb = int(pb)
    g1 = float(g1)
    g2 = float(g2)
    anos = 0
    if (100 <= pa <= 1000000 and pa < pb <= 1000000 and 0.1 <= g1 <= 10.0 and 0.0 <= g2 <= 10.0 and g2 < g1):
        while(pa <= pb):
            pa += int((pa * g1)/100)
            pb += int((pb * g2)/100)
            anos += 1
            if (anos > 100):
                break
        if (anos <= 100):
            #print(anos, "anos")
            return str(anos) + " anos"
        else:
            #print(anos)
            #print("Mais de um século")
            return "Mais de um século"
    else:
        #print("Inválido")
        return "Inválido"
```

O grafo de fluxo de controle abaixo foi feito a partir de “crescimento_populacional()”:



Implementação das suites de teste:

```python
import unittest
import crescimento_populacional

class RoteiroTeste(unittest.TestCase):
    def test_ct01(self):
        res = crescimento_populacional.crescimento_populacional(99, 600, 5.0, 2.5)
        self.assertEqual (res, "Inválido")

    def test_ct02(self):
        res = crescimento_populacional.crescimento_populacional(100, 101, 5.0, 2.5)
        self.assertEqual (res, "1 anos")

    def test_ct03(self):
        res = crescimento_populacional.crescimento_populacional(1000000, 1000001, 5.0, 2.5)
        self.assertEqual (res, "Inválido")

    def test_ct04(self):
        res = crescimento_populacional.crescimento_populacional(1000001, 1000002, 5.0, 2.5)
        self.assertEqual (res, "Inválido")

    def test_ct05(self):
        res = crescimento_populacional.crescimento_populacional(500, 500, 5.0, 2.5)
        self.assertEqual (res, "Inválido")

    def test_ct06(self):
        res = crescimento_populacional.crescimento_populacional(500, 1000000, 5.0, 2.5)
        self.assertEqual (res, "Mais de um século")

    def test_ct07(self):
        res = crescimento_populacional.crescimento_populacional(500, 1000001, 5.0, 2.5)
        self.assertEqual (res, "Inválido")
```

```python
    def test_ct08(self):
        res = crescimento_populacional.crescimento_populacional(500, 600, 0.0, -0.1)
        self.assertEqual (res, "Inválido")

    def test_ct09(self):
        res = crescimento_populacional.crescimento_populacional(500, 600, 0.1, 0.0)
        self.assertEqual (res, "Mais de um século")

    def test_ct10(self):
        res = crescimento_populacional.crescimento_populacional(500, 600, 10.0, 9.9)
        self.assertEqual (res, "Mais de um século")

    def test_ct11(self):
        res = crescimento_populacional.crescimento_populacional(500, 600, 10.1, 10.0)
        self.assertEqual (res, "Inválido")

    def test_ct12(self):
        res = crescimento_populacional.crescimento_populacional(500, 600, 5.0, -0.1)
        self.assertEqual (res, "Inválido")

    def test_ct13(self):
        res = crescimento_populacional.crescimento_populacional(500, 600, 9.0, 10.1)
        self.assertEqual (res, "Inválido")

    def test_ct14(self):
        res = crescimento_populacional.crescimento_populacional(500, 600, 5.0, 5.0)
        self.assertEqual (res, "Inválido")

if __name__ == "__main__":
    unittest.main()
```

A imagem abaixo mostra pytest --cov verificando se os testes têm 100% de cobertura do código:

```
helder@helder-X451CA:~/Área de Trabalho/src$ pytest --cov=crescimento populacional
============================= test session starts ==============================
platform linux -- Python 3.6.5, pytest-7.0.1, pluggy-1.0.0
rootdir: /home/helder/Área de Trabalho/src
plugins: arraydiff-0.2, doctestplus-0.1.3, openfiles-0.3.0, remotedata-0.3.3, cov-3.0.0
collected 14 items

test crescimento populacional.py ..............                          [100%]

---------- coverage: platform linux, python 3.6.5-final-0 ----------
Name                         Stmts   Miss  Cover
------------------------------------------------
crescimento populacional.py      17      0   100%
------------------------------------------------
TOTAL                            17      0   100%


============================== 14 passed in 0.18s ==============================
```