**Engenharia de Software 3 – Teste de Caixa Preta**

1. Crie um Algoritmo que diga se um CPF é válido ou não

* Retorne true se o CPF for válido e false se não for
* O CPF deve estar no formato String
* O CPF tem que ter caracteres numéricos e pode ter “.” e “-”
* Deve ter somente 11 números
* O CPF nullo ou String vazia deve ser validado como false
* O CPF não pode ser uma sequência, por exemplo 111.111.1111-11
* Você precisa fazer conta que irá gerar um novo cpf o qual deve ser igual ao cpf passado como paramêtro

Vamos começar trabalhando com um CPF, usando de exemplo o número: 145.382.206-20

O cálculo de validação do CPF é bem direto. Ele funciona através de pesos associados a cada número e uma divisão pelo número primo 11 ao final. Vamos vê-lo em etapas.

Começamos utilizando os 9 primeiros dígitos multiplicando-os pela sequência decrescente de 10 à 2 e somamos esse resultado.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **4** | **5** | **3** | **8** | **2** | **2** | **0** | **6** |
| X | X | X | X | X | X | X | X | X |
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 10 | 36 | 40 | 21 | 48 | 10 | 8 | 0 | 12 |

10 + 36 + 40 + 21 + 48 + 10 + 8 + 0 + 12 = 185

Com esse resultado em mãos, vamos dividi-lo por 11, mas o importante para nós não é resultado, mas sim o módulo (resto) da divisão.

185 % 11 = 9

O resto da divisão é 9. Agora para calcular o dígito verificador vamos subtrair este resto do número 11:

11 - 9 = 2

Como o resultado da da subtração foi 2, o primeiro dígito verificador é igual a 2. Caso o resultado dessa divisão for 10 ou maior, o penúltimo dígito verificador será o 0.

A validação do segundo dígito é semelhante a primeira, porém vamos considerar o primeiro dígito verificador calculado anteriormente. Por isso a multiplicação é feita de 11 à 2.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **4** | **5** | **3** | **8** | **2** | **2** | **0** | **6** | **2** |
| X | X | X | X | X | X | X | X | X | X |
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| 11 | 40 | 45 | 24 | 56 | 12 | 10 | 0 | 18 | 4 |

11 + 40 + 45 + 24 + 56 + 12 + 10 + 0 + 18 + 4 = 220

Novamente vamos efetuar a divisão por 11 usando o módulo:

220 % 11 = 0

E vamos fazer a subtração:

11 - 0 = 11

Como o valor é igual ou maior que 10, o último dígito é 0.

Assim, confirmamos os dois dígitos verificadores do nosso CPF 145.382.206-**20** e sabemos que esse CPF é válido. Outra regra muito importante é que CPFs com números iguais como: 111.111.111-11, 222.222.222-22, entre outros, são CPFs válidos pelo algoritmo, mas não existem no registro oficial. Assim esse tipo de CPF não pode ser usado.

|  |  |  |
| --- | --- | --- |
| **Condições de Entrada** | **Classes Válidas** | **Classes Inválidas** |
| CPF é vazio  (booleana) | Não **(1)** | Sim **(2)** |
| CPF é nulo  (booleana) | Não **(3)** | Sim **(4)** |
| CPF caracteres  (Conjunto) | **{numéricos, “.”,”-”} (5)** | **Conjunto != especificado (6)** |

|  |  |  |
| --- | --- | --- |
| CPF quantidade de números  (condição de valor) | **CPF.qtdNum = 11 (7)** | **CPF.qtdNum < 11 (8)**  **CPF.qtdNum > 11 (9)** |

|  |  |  |
| --- | --- | --- |
| CPF é uma sequência  (condição de valor) | **Não(10)** | **SIM(11)** |

**Para as classes válidas:**

CT1: (1,3,5,7,10)  
ler isValid = CPF(“145.382.206-**20**”).validar(); // deve retornar true

CT2: (1,3,5,7,10,12)  
ler isValid = CPF(“145382206**20**”),validar(); // deve retornar true

**Para as classes inválidas:**

Irei desconsiderar cpf nula ou vazia

CT3: (1,3,6,7,10,)  
ler isValid = CPF(“145382206**2A**”),validar(); //deve retornar false

CT4: (1,3,5,8,10) LIE(classe 8)  
ler isValid = CPF(“145382206**2**”).validar(); //deve retornar false

CT5: (1,3,5,9,10) LSE(classe 9)  
ler isValid = CPF(“145382206**209**”).validar();//deve retornar false

CT6: (1,3,5,7,11)  
ler isValid = CPF(“777.7777.777-77”).validar(); // deve retornar false