Algoritmo do CPF

Gerador de Dígitos do CPF em Java

Decifrando o Algoritmo

- O algoritmo é dividido em duas etapas;
- A primeira calcula do primeiro dígito;
- A segunda calcula o segundo dígito;
- A explicação está disponível no site <https://www.geradorcpf.com/algoritmo_do_cpf.htm>

Estudo de Caso

- Considere que foi digitado o seguinte CPF: 87329400173;
- O que queremos confirmar é se os dígitos finais estão ou não corretos;
- Para isso executaremos dois procedimentos como descritos na planilha Excel a seguir.

Estudo de Caso

Index	0	1	2	3	4	5	6	7	8	9	10
CPF	8	7	3	2	9	4	0	0	1		
X	10	9	8	7	6	5	4	3	2	X	Y
Mult	80	63	24	14	54	20	0	0	2	7	
Soma		143	167	181	235	255	255	255	**257**		
									4	**Resto**	
Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
CPF	8	7	3	2	9	4	0	0	1	7	
Y	11	10	9	8	7	6	5	4	3	2	Y
Mult	88	70	27	16	63	24	0	0	3	14	3
Soma		158	185	201	264	288	288	288	291	**305**	
									8	**Resto**	

Implementar em Java

```java
package aulas;

import java.util.Scanner;

public class CPF {
    public static void main(String[] args) {
        Scanner ler = new Scanner(System.in);
        int cpf[] = new int[11];
        String cpfValor;
        StringBuilder cpfCompara = new StringBuilder();

        System.out.println("Informe o cpf: ");
        cpfValor = ler.nextLine();
```

Implementar em Java

- Linha 1: é a pasta onde está o código fonte;
- Linha 3: importa a classe Scanner para criar o objeto que fará a leitura a partir da entrada padrão (o teclado);
- Linha 5: a definição da Classe (vai da linha 5 até a linha 70);
- Linha 6: a declaração do método principal da classe (Main);

Implementar em Java

- Linha 7: cria um objeto ler a partir da classe Scanner;
- Linha 8: cria um objeto cpf que é um vetor de inteiros com 11 posições (de 0 a 10);
- Linha 9: cria uma string cpfValor que receberá o valor digitado pelo usuário;
- Linha 10: cria um objeto cpfCompara do tipo StringBuilder; esse objeto receberá o cpf validado.

Implementar em Java

- Linha 12: exibe uma mensagem solicitando que o usuário informe o CPF;
- Linha 13: o valor digitado até o usuário pressionar ENTER será atribuído para cpfValor;

Implementar em Java

```
15            for (int i = 0; i < 9; i++){
16                cpf[i] = Integer.parseInt(String.valueOf(cpfValor.charAt(i)));
17            }
```

- Linha 15: estrutura do laço de repetição for; define uma variável de controle i, que começa em 0 e vai sendo incrementada de 1 em 1 (i++) até chegar em 8 (i < 9), pegando os 9 primeiros dígitos do CPF (de 0 a 8).

Implementar em Java

- Linha 16: converte cada dígito da string em um inteiro.
  - cpfValor.charAt(i): retorna o caractere na posição i da string cpfValor; Mais informações:
    <https://www.w3schools.com/java/ref_string_charat.asp>
  - String.valueOf: converte diferentes tipos de valores em string; Mais informações: <https://www.javatpoint.com/java-string-valueof>
  - Integer.parseInt: obtém o tipo de dado primitivo (inteiro) de uma string; Mais informações:
    <https://www.tutorialspoint.com/java/number_parseint.htm>

Implementar em Java

```
19          System.out.printf("1º Dígito verificador: %d\n", primeiroDigito(cpf));
20          cpf[9] = primeiroDigito(cpf);
21          System.out.printf("2º Dígito verificador: %d\n", segundoDigito(cpf));
22          cpf[10] = segundoDigito(cpf);
```

- As linhas 19 a 20 chamam as funções primeiroDigito e segundoDigito, que recebem como parâmetros de entrada o cpf (que é um vetor de inteiros);
- A função primeiroDigito devolve a 10ª posição do vetor;
- A função segundoDigito devolve a 11ª posição do vetor.

Implementar em Java

```
24              for (int i = 0; i < 11; i++){
25                  cpfCompara.append(Integer.toString(cpf[i]));
26              }
```

- Linha 24: cria uma estrutura de repetição (*loop*) onde:
  - i vai de 0 a 10 variando de 1 em 1;
- Linha 25: converte cada inteiro do vetor cpf em uma String (Integer.toString(cpf[i]));
- Cada valor convertido é acrescentado na variável cpfCompara, cpfCompara.append.

Implementar em Java

```
28            comparaCPF(cpfValor, cpfCompara);
29        }
```

- Linha 28: após converter o vetor de inteiros em uma String, essa linha chama a função comparaCPF, que compara as duas Strings cpfValor e cpfCompara.
- Linha 29: fecha a função principal (main).

Funções

- A aplicação utiliza-se de 3 funções além da função principal (main);
- A função primeiroDigito que retorna o 10º dígito do CPF;
- A função segundoDigito que retorna o 11º dígito do CPF;
- A função comparaCPF para comparar o CPF digitado pelo usuário com o CPF validado pelo algoritmo.

Funções

```
31      public static int primeiroDigito(int[] cpf){
32          int x, soma = 0, prod = 10;
33          for (int i = 0; i < 9; i++){
34              soma += cpf[i]*prod--;
35          }
36
37          x = soma % 11;
38
39          if (x < 2)
40              x = 0;
41          else
42              x = 11 - x;
43
44          return x;
45      }
```

Funções

- Linha 31: cabeçalho da função; define a função primeiroDigito, que retorna um inteiro e recebe um vetor de inteiros como parâmetro de entrada;
- Linha 32: declara 3 variáveis inteiras: x que representa o primeiro dígito verificado; soma para acumular a soma dos 10 primeiros dígitos do CPF; prod para multiplicar cada dígito pelo seu respectivo peso.

Funções

```
33          for (int i = 0; i < 9; i++){
34              soma += cpf[i]*prod--;
35          }
```

- Linha 33: define uma estrutura de repetição em que i vai de 0 a 8 variando de 1 em 1;
- Para cada variação de i pega-se um dígito do cpf multiplica por prod e incrementa em soma; depois decrementa prod;
- Ao percorrer todo o vetor do cpf terei a soma do produto dos 9 primeiros dígitos (de 0 a 8).

Funções

- Linha 37: x recebe o resto da divisão de soma por 11;
- Linha 39: verifica se o valor de x é menor que 2:
  - Linha 40: se for verdade a verificação da linha 39, x recebe 0;
  - Linhas 41 e 42: senão, x recebe a subtração de 11 pelo valor de x;
- Linha 43: retorna o valor de x (primeiro dígito verificador do CPF) para quem chamou a função.

Funções

```
47      public static int segundoDigito(int[] cpf){
48          int y, soma = 0, prod = 11;
49          for (int i = 0; i < 10; i++){
50              soma += cpf[i]*prod--;
51          }
52
53          y = soma % 11;
54
55          if (y < 2)
56              y = 0;
57          else
58              y = 11 - y;
59
60          return y;
61      }
```

Funções

- A função segundoDigito é muito similar a primeiroDigito;
- A diferença é que para calcular o segundo dígito também se considera o primeiro dígito verificador;
- A função percorre o vetor cpf da 1ª a 10ª posição, multiplicando cada dígito pelo seu peso correspondente, definido por prod;
- Depois soma esses valores e verifica o resto da divisão;
- Retorna para quem chamou o segundo dígito identificado por y.

Funções

```
63      public static void comparaCPF(String a, StringBuilder b){
64          if (a.equals(b.toString()))
65              System.out.println("CPF válido!\n");
66          else
67              System.out.println("CPF inválido!\n");
68
69      }
70  }
```

Funções

- Linha 63: declara o cabeçalho da função comparaCPF; essa função não tem retorno, portanto void; ela recebe dois valores distintos, a (String) e b (StringBuilder); para maiores informações sobre os tipos String e StringBuilder acesse o artigo: <encurtador.com.br/nBR12>;
- O parâmetro a representa o CPF digitado pelo usuário do programa (cpfValor);
- O parâmetro b representa o CPF validado pelo programa (cpfCompara).

Funções

- Linha 64: verifica se a é igual a b;
  - como a e b são de tipos diferentes, o método b.toString() converte a variável b em uma String;
  - agora a e b são Strings;
  - o método equals compara duas Strings e retorna True (verdadeiro) se são iguais ou False (falso) se forem distintas.
  - se retornar True exibe “CPF válido!”;
  - senão exibe “CPF inválido!”.

Funções

A chave } na Linha 70 encerra a classe CPF.java.

Desafio

- O que é IRRF?
- Qual é a faixa de isenção para o IRRF no ano corrente?
- Quantas faixas de alíquotas existem?
- Quais são essas faixas?
- Como se calcula a dedução do IRRF?