FÍSICA APLICADA A INFORMÁTICA (IFA)

*Prof. **Waldemir** de Paula Silveira - waldemir.silveira@ifsp.edu.br*
*Prof. José Augusto Navarro Garcia **Manzano** - augusto.garcia@ifsp.edu.br*

MODULAÇÃO DE DADOS (DÍGITOS VERIFICADORES)

INTRODUÇÃO

O dígito verificador (*check digit*) é um mecanismo de validação de dados usado para garantir a integridade, autenticidade e precisão de um código, normalmente numérico.

São usados na validação de números de cartões de crédito, contas bancárias, identificadores fiscais, matrículas, números de documentos, etc.

O dígito verificador funciona adicionando-se este dígito ao final de uma sequência de números usado para a identificação de certo elemento.

O dígito verificador é calculado a partir dos outros dígitos usando uma fórmula matemática específica.

O dígito verificador tem como vantagem:

- Prevenção de erros humanos: o dígito verificador ajuda a prevenir erros ao escrever números;
- Poupar tempo e evitar a necessidade de verificações manuais, o que é útil em grandes conjuntos de dados;
- Evita erros de transposição de números.

Para a geração de dígitos verificadores existem diversos algoritmos, sendo populares os algoritmos de módulo 10 e módulo 11.

O termo “módulo” significa que em algum ponto do algoritmo do dígito verificador far-se-á o cálculo da divisão por “10” ou por “11” a fim de obter o valor do resto da divisão.

A seguir são descritos em linhas gerais o mecanismo de cada um dos métodos de geração de dígito verificador.

MÓDULO 10

O Algoritmo *Modulo 10* é usado a partir do seguinte algoritmo:

- As posições do número básico da última posição de duas em duas, da direita para esquerda são multiplicados por "2";
- Os dígitos multiplicados por "2" são adicionados aos dígitos das demais posições que não foram multiplicados;
- O resultado da multiplicação por "2" quando maior que "9" devem ter seus dígitos somados ou subtraídos de "9";
- O somatório dos dígitos deve ser subtraído do maior valor terminado em zero a partir do resultado do somatório, obtendo-se o dígito verificador: **PM = INT((SOMA/10)+1)*10**.

Para demonstrar o módulo 10 considere obter o dígito verificador da matrícula "987.654":

Algoritmo: Módulo 10						
Código numérico	9	8	7	6	5	4
Peso multiplicador	-	x 2	-	x 2	-	x 2
Resultado parcial	-	16	-	12	-	8
Ajuste quando >= 10	9	16 - 9	7	12 - 9	5	8
Resultado final	9	7	7	3	5	8
Σ do resultado final	39					
Maior número seguinte terminando em 0 (*)	40					
DV = 40 – 39	1					

MÓDULO 11

O Algoritmo *Modulo 11* é usado a partir do seguinte algoritmo:

- Para cada posição que formam as unidades que compõe o número básico é atribuído um fator de ponderação. Os fatores de ponderação são valores de 7 até 2 da direita para a esquerda, repetindo-se esta sequência sempre que necessário;

- Cada dígito do número básico é multiplicado pelo fator de ponderação e os produtos obtidos são somados e divididos por 11. O resultado obtido é o dígito verificador;

- Se resto “10”, isto é um erro, pode-se substituir por “X”;

- Se resto “0”, dígito verificador será “0”.

Para demonstrar o módulo 11 considere obter o dígito verificador da matrícula “987.654”:

Algoritmo: Módulo 11						
Código numérico	**9**	**8**	**7**	**6**	**5**	**4**
Peso multiplicador	**x 7**	**x 6**	**x 5**	**x 4**	**x 3**	**x 2**
Resultado parcial	**63**	**48**	**35**	**24**	**15**	**8**
Σ do resultado final	**193**					
Divisão do Σ por 11	**quociente = 17**			**resto = 6**		
DV = 11 – resto	**5**					

VALIDAÇÃO DE CPF

O Algoritmo *Modulo 11* é usado a partir do seguinte algoritmo:

Todo brasileiro ao se inscrever na Receita Federal do Brasil recebe um número de 11 dígitos decimais com a configuração: “**ABC.DEF.GHI-JK**”:

- Os dígitos, **ABCDEFGH**, formam o número-base;

- O dígito **I**, é a Região Fiscal no momento da inscrição;

- O dígito **J**, é o primeiro verificador de **ABCDEFGHI**;

- O dígito **K**, é o segundo de **ABCDEFGHI** com **J**.

A região fiscal de emissão do CPF caracteriza-se:

1 – DF, GO, MS, MT e TO	6 – MG
2 – AC, AM, AP, PA, RO e RR	7 – ES e RJ
3 – CE, MA e PI	8 – SP
4 – AL, PB, PE, RN	9 – PR e SC
5 – BA e S	0 – RS

Para demonstrar o DV do CPF considere o número “551.878.649”:

5	5	1	8	7	8	6	4	9		
1	2	3	4	5	6	7	8	9		
5	10	3	32	35	48	42	32	81		

5 + 10 + 3 + 32 + 35 + 48 + 42 + 32 + 81 = 288

288 : 11 = 26 com resto 2

Se resto for igual a 10 o resto é zero.

5	5	1	8	7	8	6	4	9	2	
0	1	2	3	4	5	6	7	8	9	
0	5	2	24	28	40	36	28	72	18	

0 + 5 + 2 + 24 + 28 + 40 + 36 + 28 + 72 + 18 = 253

253 : 11 = 23 com resto 0

Se resto for igual a 10 o resto é zero.

5	5	1	8	7	8	6	4	9	2	0
0	1	2	3	4	5	6	7	8	9	
0	5	2	24	28	40	36	28	72	18	

Todos os números de um CPF não podem ser iguais, como: **000.000.000-00** a **999.999.999-99** e tão pouco poderá ser **123.456.789-09**.

CPFs para teste: 350.647.831-16
282.134.001-00
271.008.801-06

CODIFICAÇÃO EM LINGUAGEM LUA

Programa para validação de CPF

```
-- cpf.lua

function validaCPF(cpf)

  local digitos = {}
  local soma1, soma2 = 0, 0
  local dv1, dv2

  if cpf:len() ~= 11 or
    cpf == "00000000000" or
    cpf == "11111111111" or
    cpf == "22222222222" or
    cpf == "33333333333" or
    cpf == "44444444444" or
    cpf == "55555555555" or
    cpf == "66666666666" or
    cpf == "77777777777" or
    cpf == "88888888888" or
    cpf == "99999999999" or
    cpf == "12345678909"
  then
    return false
  end

  for i = 1, 11 do
    digitos[i] = tonumber(string.sub(cpf, i, i))
  end

  for i = 1, 9 do
    soma1 = soma1 + digitos[i] * i
  end

  dv1 = soma1 % 11
  if dv1 == 10 then
    dv1 = 0
  end

  for i = 1, 10 do
    soma2 = soma2 + digitos[i] * (i - 1)
  end

  dv2 = soma2 % 11
  if dv2 == 10 then
    dv2 = 0
  end

  if dv1 == digitos[10] and dv2 == digitos[11] then
    return true
  else
    return false
  end

end

io.write("Informe CPF no formato 999.999.999-99: ")
cpf_cfmt = io.read()
cpf_sfmt = cpf_cfmt:gsub("%.", ""):gsub("-", "")
if validaCPF(cpf_sfmt) then
```

```
  print("CPF valido")
else
  print("CPF invalido")
end

-- string:len() = retorna o tamanho da string
-- string.sub(cadeia, índice_ini, índice_fim) = retorna uma sub-cadeia
-- string.gsub(busca, troca) = substitui certa ocorrência
```