Pamela Bandeira Gerber

Verificação do Cadastro de Pessoas Físicas

Itajaí - SC

2020

Pamela Bandeira Gerber

Verificação do Cadastro de Pessoas Físicas

Avaliação 4 da disciplina de ICC

Universidade do Vale do Itajaí - UNIVALI
Bacharelado em Ciências da Computação

Orientador: Cristina Ono Horita

Itajaí - SC
2020

Lista de ilustrações

Figura 1 – "Fluxograma do código-fonte". 6
Figura 2 – "Código-fonte". 9
Figura 3 – "Primeira tabela disponibilizada". 10
Figura 4 – "Segunda tabela disponibilizada". 11
Figura 5 – "Verificação". 12

Lista de tabelas

Tabela 1 – "Teste de Mesa dos algoritmos". 7

Sumário

1 INTRODUÇÃO 5

2 DESENVOLVIMENTO 6

2.0.1 Projeto 6

2.0.2 Implementação 8

2.0.3 Verificação 12

Conclusão 13

REFERÊNCIAS 14

1 Introdução

Este presente documento é um relatório sobre o projeto desenvolvido na disciplina de Algoritmo e Programação, criando um algoritmo que solicita ao usuário, um valor inteiro de 8 a 9 dígitos representando o número de um CPF já existente e que, em seguida, acharemos os dois dígitos que estão faltando, o verificando, exibindo assim, o CPF por completo.

Além disso, o trabalho contará com fluxograma e teste de mesa de como foi pensada e concretizada para a obtenção de seus resultados.

Equipe Univali

Pamela Bandeira Gerber

2 Desenvolvimento

2.0.1 Projeto

Vindo da disciplina de Algoritmo e Programação, o nome da matéria em si é chamado de algoritmo sequencial, que deve-se calcular os dígitos de um CPF obtendo os resultados do penúltimo e último dígito que falta para completá-lo. Abaixo está o fluxograma (Figura 1) e teste de mesa(Tabela 1) declarando as variáveis e cada resultado nela adquirida.

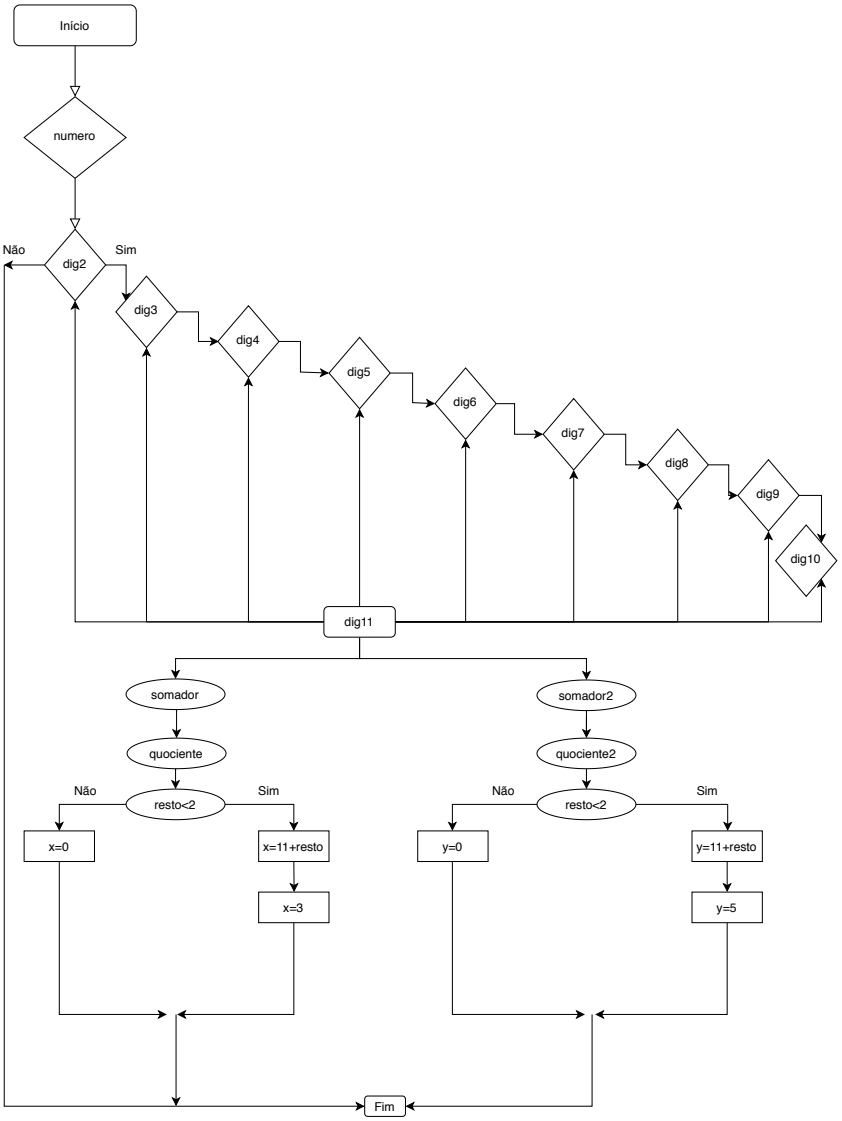

Figura 1 – "Fluxograma do código-fonte".

Tabela 1 – "Teste de Mesa dos algoritmos".

Variáveis	**Resultados**
numero	111444777
dig2	7
dig3	7
dig4	7
dig5	4
dig6	4
dig7	4
dig8	1
dig9	1
dig10	1
dig11	0
somador	162
somador2	204
quociente	14
quociente2	18
resto	8
resto2	6
x	3
y	5

2.0.2 Implementação

Abaixo está o projeto feito em C++:(Figura 2 em PDF)

```
#include <iostream>

using namespace std;

int main()
{
    int numero, dig2, dig3, dig4, dig5, dig6, dig7, dig8, dig9, dig10,
       dig11, somador, somador2, quociente,quociente2, resto, resto2, x,
        y;
    cout << "Digite␣numero␣de␣8␣a␣9␣digitos:␣" << endl;
    cin >> numero;

    dig2 = numero % 10;
    dig11 = numero / 10;

    dig3 = dig11% 10;
    dig11 = dig11/ 10;

    dig4 = dig11% 10;
    dig11 = dig11/10;

    dig5 = dig11%10;
    dig11 = dig11/10;

    dig6 = dig11%10;
    dig11 = dig11/10;

    dig7 = dig11%10;
    dig11 = dig11/10;

    dig8 = dig11%10;
    dig11 = dig11/10;

    dig9 = dig11%10;
    dig11= dig11/10;

    dig10= dig11%10;
    dig11= dig11/10;

    somador = (dig10 * 10)+ (dig9 * 9)+ (dig8 * 8)+ (dig7 * 7)+ (dig6 *
       6)+ (dig5 * 5)+ (dig4 * 4)+ (dig3 * 3)+ (dig2 * 2);
    quociente = somador / 11;
    resto = somador % 11;
    if (resto < 2) {
```

```
        x = 0;
    } else{
        x = 11 - resto;
    }
    somador2= (dig10 * 11)+ (dig9 * 10)+ (dig8 * 9) + (dig7 * 8)+ (dig6
       * 7)+ (dig5 * 6)+ (dig4 * 5)+ (dig3 * 4)+ (dig2 * 3)+ (x * 2);
    quociente2 = somador2 / 11;
    resto2 = somador2 %11;
    if (resto2 < 2) {
        y= 0;
    } else {
        y= 11 - resto2;
    }
    cout << "Digitos Verificadores:" <<"\n"<<x<<"\n"<<y;
    cout << "\nVerifica-se que o CPF completo:" <<dig10<<dig9<<dig8<<
       dig7<<dig6<<dig5<<dig4<<dig3<<dig2<<x<<y;
    return 0;
}
```

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int numero, dig2, dig3, dig4, dig5, dig6, dig7, dig8, dig9, dig10, dig11, somador, somador2, quociente,
quociente2, resto, resto2, x, y;
8      cout << "Digite numero de 8 a 9 digitos: " << endl;
9      cin >> numero;
10
11     dig2 = numero % 10;
12     dig11 = numero / 10;
13
14     dig3 = dig11% 10;
15     dig11 = dig11/ 10;
16
17     dig4 = dig11% 10;
18     dig11 = dig11/10;
19
20     dig5 = dig11%10;
21     dig11 = dig11/10;
22
23     dig6 = dig11%10;
24     dig11 = dig11/10;
25
26     dig7 = dig11%10;
27     dig11 = dig11/10;
28
29     dig8 = dig11%10;
30     dig11 = dig11/10;
31
32     dig9 = dig11%10;
33     dig11= dig11/10;
34
35     dig10= dig11%10;
36     dig11= dig11/10;
37
38     somador = (dig10 * 10)+ (dig9 * 9)+ (dig8 * 8)+ (dig7 * 7)+ (dig6 * 6)+ (dig5 * 5)+ (dig4 * 4)+ (dig3 *
3)+ (dig2 * 2);
39     quociente = somador / 11;
40     resto = somador % 11;
41     if (resto < 2) {
42         x = 0;
43     } else{
44         x = 11 - resto;
45     }
46     somador2= (dig10 * 11)+ (dig9 * 10)+ (dig8 * 9) + (dig7 * 8)+ (dig6 * 7)+ (dig5 * 6)+ (dig4 * 5)+ (dig3
* 4)+ (dig2 * 3)+ (x * 2);
47     quociente2 = somador2 / 11;
48     resto2 = somador2 %11;
49     if (resto2 < 2) {
50         y= 0;
51     } else {
52         y= 11 - resto2;
53     }
54     cout << "Digitos Verificadores:" <<"\n"<<x<<"\n"<<y;
55     cout << "\nVerifica-se que o CPF completo:" <<dig10<<dig9<<dig8<<dig7<<dig6<<dig5<<dig4<<dig3<<dig2<<x<<
y;
56     return 0;
57 }
```

Figura 2 – "Código-fonte".

Primeiramente utilizei a linguagem de programação C++,(1) no CodeBlocks, utilizei o tipo de dado “int” indicando as variáveis inteiras, pelo fato de que CPF não obtém virgula e número nulo, logo depois estão as variáveis que representa respectivamente cada desenvolvimento, tais como: CPF inteiro representado pela variável numero, dig 2 até dig10 são as variáveis que representam os 9 dígitos de qualquer CPF que tenhas optado para compilar, dig11 é a variável que se repete pegando a décima parte da unidade e dividindo por 10, somador, quociente, resto, somador2, quociente2, resto2, x e y eles estão como letras porque é o resultado do penúltimo e último digito que estamos procurando.

Embaixo teremos a parte da impressão do dado quando console “cout”, aonde estará escrito, quando processar, a escrita do que se colocou("Digite numero de 8 a 9 digitos: "), depois tem o “cin” que realiza a leitura dos dados, que no acaso a variavél utilizada é o “numero” de até 9 dígitos, exemplo 111444777.

Logo depois se tem os dig2 a dig10 são os números decimais menos significativos de trás para frente, ou seja, o dig2 ele irá representar o último número do CPF que utilizou, como exemplo, mencionei o 111444777, ele representa o ultimo 7, assim sucessivamente, até chegar ao dig10 que é o primeiro número do CPF que foi utilizado, nesse exemplo é o número 1, e abaixo deles tem o dig11 que dá os valores de cada dig.

Depois temos a variável somador, que pega os dig10 ao dig2 multiplicando pelo peso de cada um que a professora disponibilizou no material (Figura 3). Depois de achar o resultado, somamos eles, dará 162. O quociente significa a divisão, pegamos o resultado da soma e dividimos por 11 que é o total de dígitos de um CPF, porém queremos o resto dessa divisão. Se o resto for menor que 2, o penúltimo dígito que estamos à procura, ele será igual a 0, se for maior que 2, terá que ser feito o 11 menos o resultado do resto, o resto deu 8, ele é maior que dois, ai utilizaremmos o recurso de 11 menos o 8, o resultado do x deu 3.

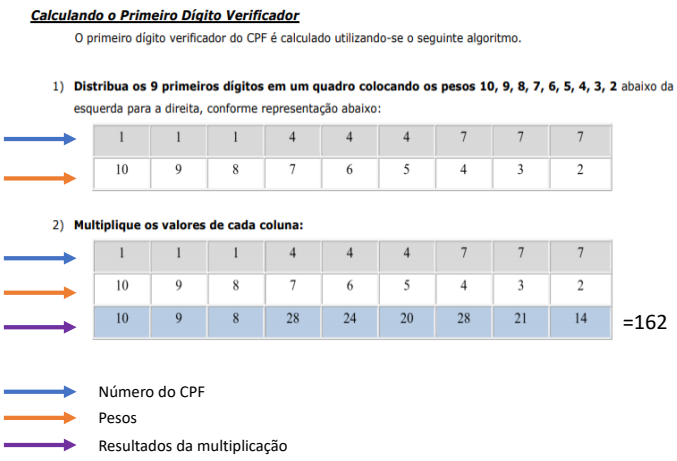

Figura 3 – "Primeira tabela disponibilizada".

Depois temos mais uma variável o somador2, pega novamente o dig10 até o dig2, porém multiplicando com pesos diferentes do anterior, como a professora disponibilizou a tabela para acharmos o segundo dígito que falta (Figura 4) e foi adicionado o x pela questão de termos achado o penúltimo digito que faltava. Depois de achar o resultado, somamos eles, dará o resultado 204. O quociente2 é a divisão, pegamos o resultado da soma e dividimos por 11 que é o total de dígitos de um CPF, porém queremos o resto dessa divisão, o resto2 deu 6, por ele não ter sido dado resultado menor que 2, deve-se fazer 11 menos o resultado do resto2, y deu 5. Então o penúltimo digito que faltava deu 3 e o último deu 5. CPF completo é: 11144477735.

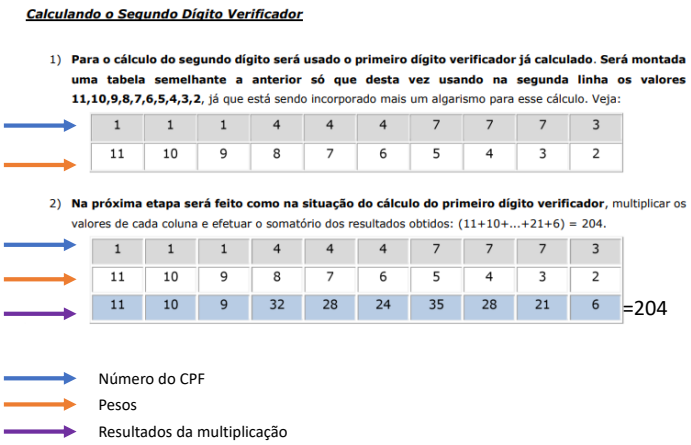

Figura 4 – "Segunda tabela disponibilizada".

2.0.3 Verificação

Na Figura 55, está a verificação concluída com sucesso, destacado na parte superior o botão "build and run"de sua funcionalidade para o processamento.

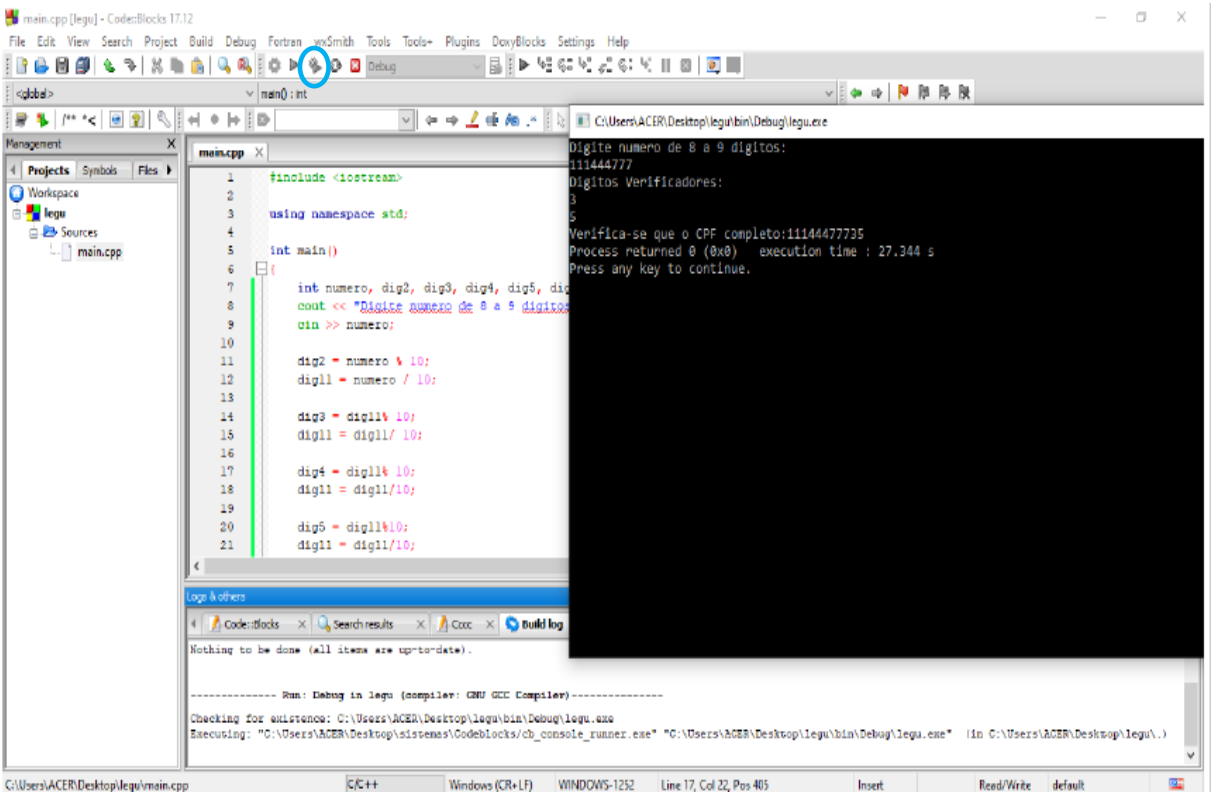

Figura 5 – "Verificação".

Conclusão

Esta atividade foi tratado em criar um algoritmo em linguagem C++ utilizando o CodeBlocks.

Criado de acordo com dicas que a professora disponibilizou, assim, demonstrado os códigos e a explicado um por um da sua utilização, este sistema foi testado pelo ambiente de programa que utilizei, assim verificou seu funcionamento apertando no “build and run” o compilando, obtendo o resultado esperado.

A função dessa verificação, não é apenas atender só o caractere "xxx.xxx.xxx-xx", e sim, consolidar mais conhecimentos lógicos operacionais de que envolve todo um cálculo por trás para ser considerado um CPF válido ou inválido.

Referências

1 AGUILAR, L. J. *Programação em C++-: Algoritmos, estruturas de dados e objetos*. [S.l.]: Bookman Editora, 2008. 10 Nenhuma citação no texto.Citado na página 10.