

**Relatório**

**Algoritmos e Estruturas de Dados 1**

**Aluno/os: Pedro Vieria Simões / Diogo Espírito Santo**

**Numero/os: 21140 / 21146**

**Professor/es: Marta Martinho**

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, janeiro, 2021

Resumo

Relatório sobre o Trabalho Prático de Algoritmos e Estruturas de Dados I, envolvendo 20 exercícios e resolução dos mesmos em fluxograma, pseudocódigo e código em C.

Índice

[1. Introdução 8](#_Toc527567531)

[1.1. Contextualização 8](#_Toc527567532)

[1.2. Motivação e Objetivos 8](#_Toc527567533)

[1.3. Estrutura do Documento 8](#_Toc527567534)

[2. Instruções de Decisão 9](#_Toc527567535)

[2.1. Questão 1 9](#_Toc527567536)

[2.1.1. Descrição e abordagem do problema 9](#_Toc527567537)

[2.1.2. Fluxograma 9](#_Toc527567538)

[2.1.3. Pseudocódigo 9](#_Toc527567539)

[2.1.4. Testes 9](#_Toc527567540)

[2.2. Questão 2 10](#_Toc527567541)

[2.2.1. Descrição e abordagem do problema 10](#_Toc527567542)

[2.2.2. Pseudocódigo 10](#_Toc527567543)

[2.2.3. Testes 10](#_Toc527567544)

[3. Instruções de Repetição 11](#_Toc527567545)

[3.1. Questão 1 11](#_Toc527567546)

[3.1.1. Descrição e abordagem do problema 11](#_Toc527567547)

[3.1.2. Fluxograma 11](#_Toc527567548)

[3.1.3. Pseudocódigo 11](#_Toc527567549)

[3.1.4. Traçagens e testes 11](#_Toc527567550)

[3.2. Questão 2 12](#_Toc527567551)

[3.2.1. Descrição e abordagem do problema 12](#_Toc527567552)

[3.2.2. Pseudocódigo 12](#_Toc527567553)

[3.2.3. Testes 12](#_Toc527567554)

[4. Funções e Procedimentos 13](#_Toc527567555)

[4.1. Questão 1 13](#_Toc527567556)

[4.1.1. Descrição e abordagem do problema 13](#_Toc527567557)

[4.1.2. Fluxograma 13](#_Toc527567558)

[4.1.3. Pseudocódigo 13](#_Toc527567559)

[4.1.4. Testes 13](#_Toc527567560)

[4.2. Questão 2 13](#_Toc527567561)

[4.2.1. Descrição e abordagem do problema 13](#_Toc527567562)

[4.2.2. Pseudocódigo 13](#_Toc527567563)

[4.2.3. Testes 13](#_Toc527567564)

[5. Arrays 14](#_Toc527567565)

[5.1. Questão 1 14](#_Toc527567566)

[5.1.1. Descrição e abordagem do problema 14](#_Toc527567567)

[5.1.2. Pseudocódigo 14](#_Toc527567568)

[5.1.3. Testes 14](#_Toc527567569)

[6. Estruturas 15](#_Toc527567570)

[6.1. Estruturas desenvolvidas 15](#_Toc527567572)

[6.3. Funções e procedimentos desenvolvidos 15](#_Toc527567573)

[6.4. Testes 15](#_Toc527567574)

# Introdução

## Contextualização

Realização de 20 exercícios atribuídos pela professora Marta Martinho, estando este trabalho dividido em 5 entregas distintas: 1ª entrega relacionada a Instruções de Decisão, 2ª entrega relacionada a Instruções de Repetição, 3ª entrega relacionada a Funções e Procedimentos, 4ª entrega relacionada a Arrays, 5ª entrega relacionada a Estruturas de Dados.

Cada entrega tem um número de exercícios diferente, sendo estes todos resolvidos em código na Linguagem C, um fluxograma e um pseudocódigo por exercício nas entregas 1,2,3.

## Motivação e Objetivos

A cada entrega desenvolver e apurar técnicas de Algoritmia(…)

## Estrutura do Documento

# Instruções de Decisão

## Questão **1**

### Descrição e abordagem do problema

Criação de uma aplicação que solicite a quantidade de entregas no final do dia de cada um dos cinco paquetes contratados e que calcule a média das quantidades de refeições entregues por paquete e a soma das refeições entregues pelos paquetes com mais de 4 refeições entregues nesse dia.

### Fluxograma

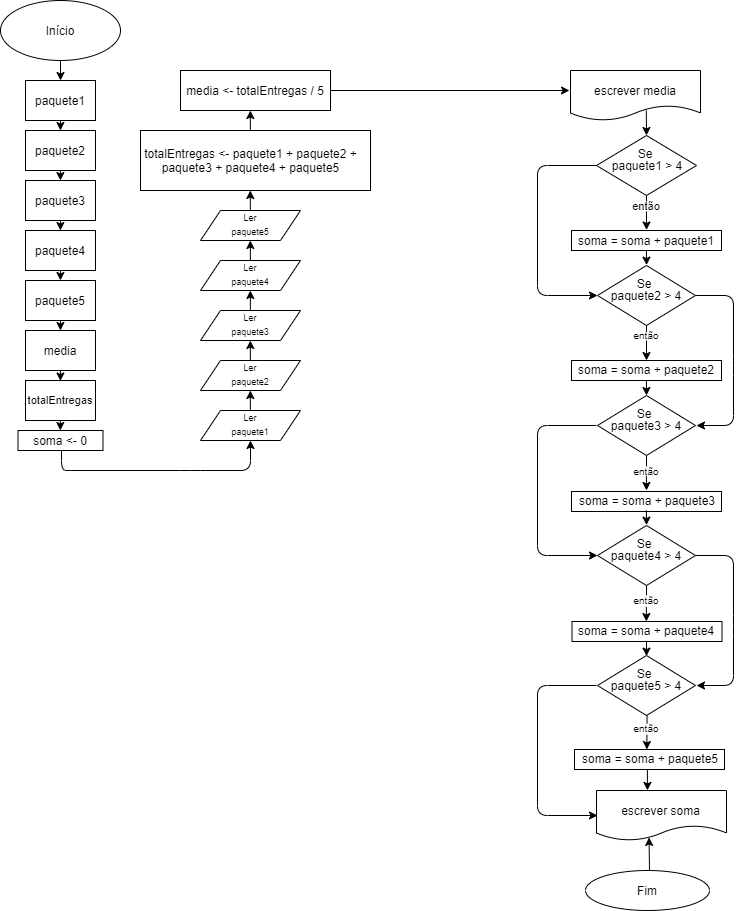
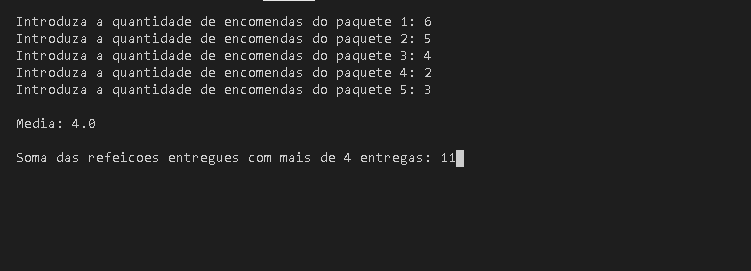
****

Figura 1 - fluxograma 1

### Pseudocódigo

### Pseudocódigo

### Testes



## Questão **2**

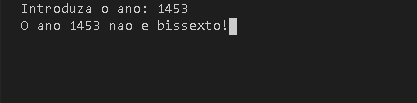
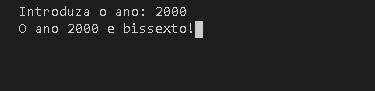
### Descrição e abordagem do problema

Criação de uma aplicação que leia o valor de um ano e escreva se o este é ou não bissexto.

### Fluxograma

### Pseudocódigo

### Testes

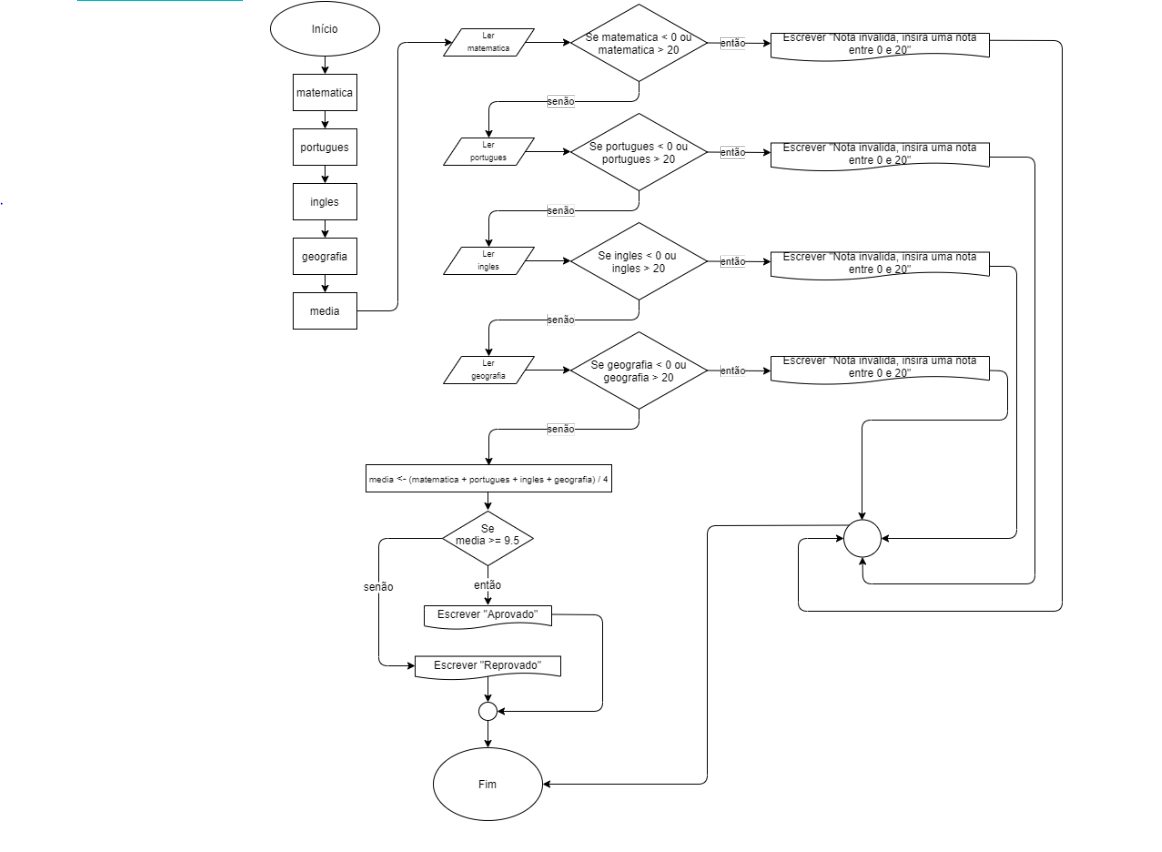


## Questão **3**

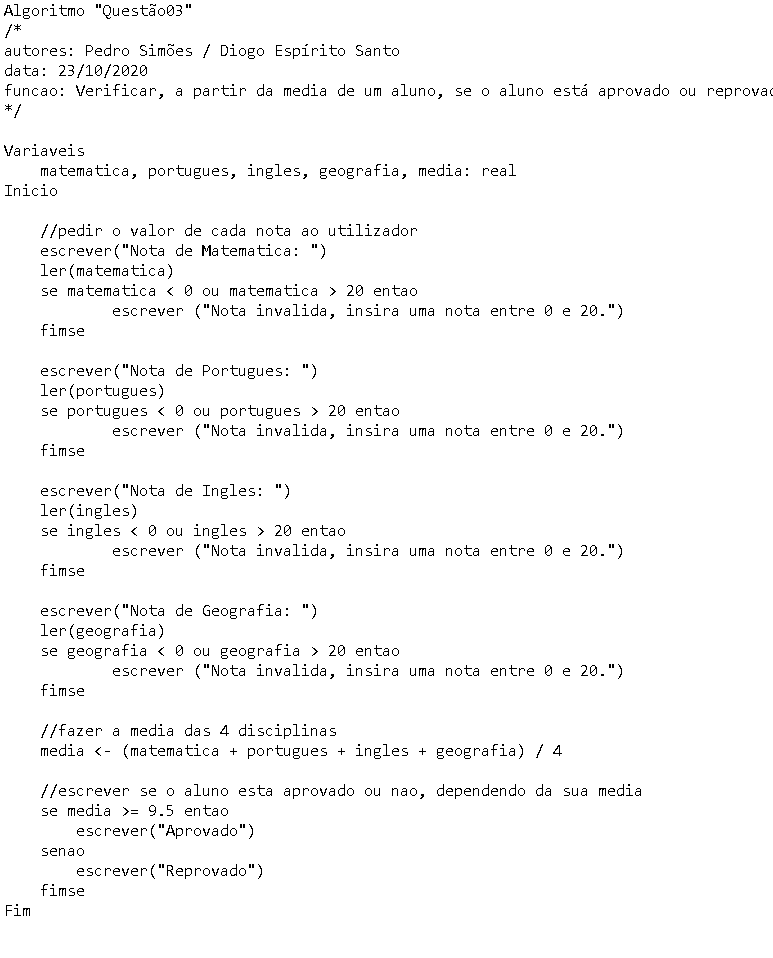
### Descrição e abordagem do problema

Criação de uma aplicação que permita ler as notas de um aluno e calcular a sua média. Em função da média deve ser mostrada uma mensagem com o conteúdo "Aprovado" ou "Reprovado".

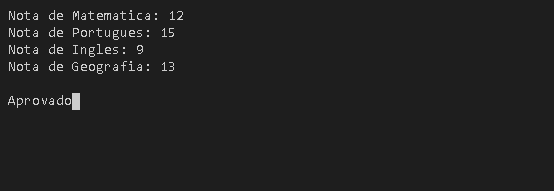
### Fluxograma



### Pseudocódigo



### 2.3.4 Testes

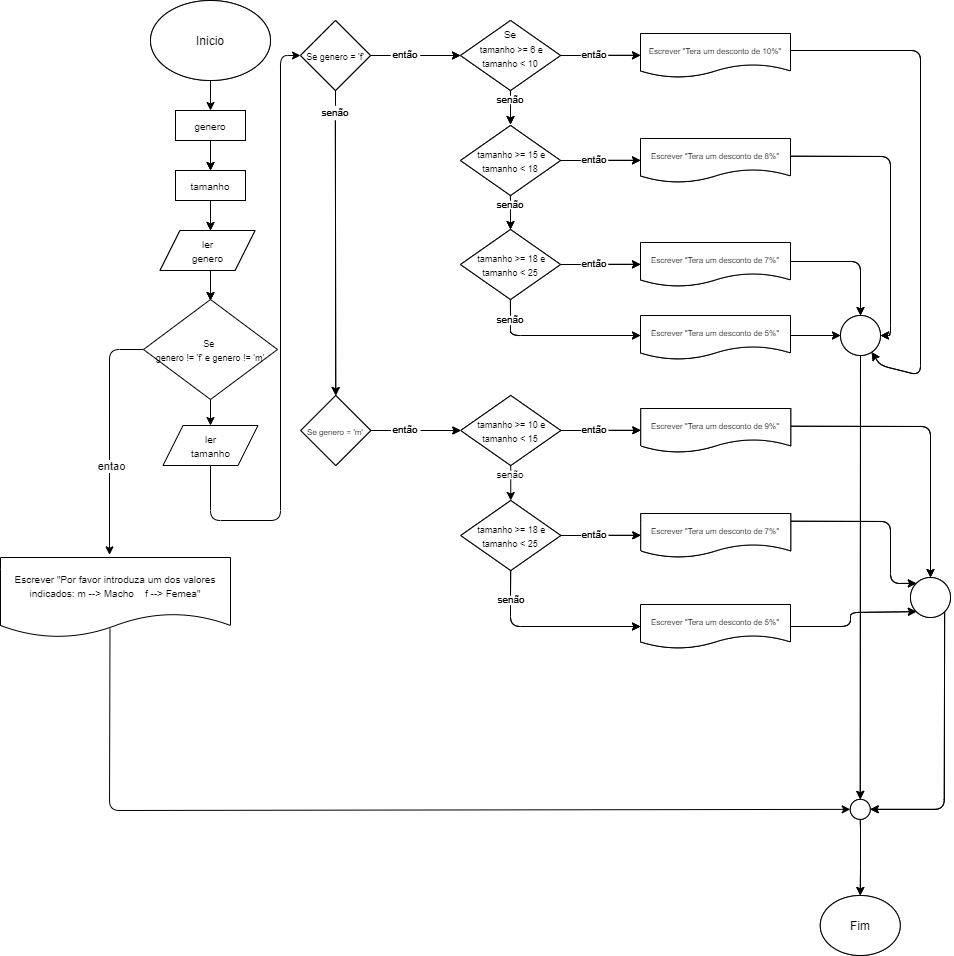


## Questão **4**

### Descrição e abordagem do problema

Criação de uma aplicação que solicite os dados de um gato e que indique ao utilizador qual o desconto a atribuir com base no quadro seguinte.

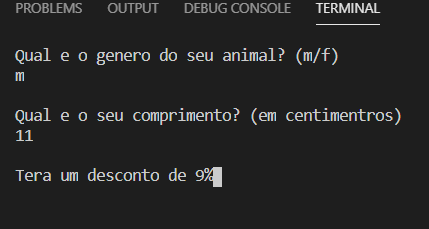
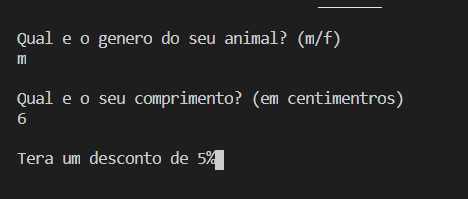
### Fluxograma

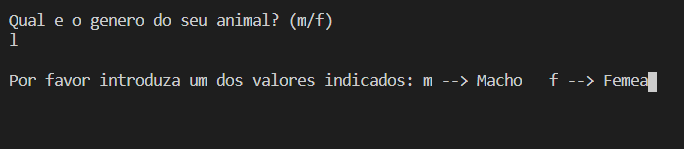


### Pseudocódigo



### Testes





# Instruções de Repetição

## Questão 1

### Descrição e abordagem do problema

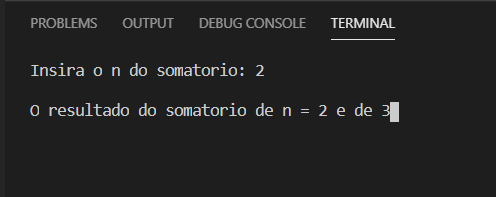
Criação de uma aplicação que solicite ao utilizador um número natural e que mostre ao utilizador o resultado de n ∑ i=1 xi em que xi representa o i^ensino elemento do conjunto de 1 até n(e.g. para n=3 ∑ i=1 x o somatório a realizar pode também ser representado por(1 + 2 + 3 )).

### Fluxograma

### Pseudocódigo

\*

### Testes



## Questão 2

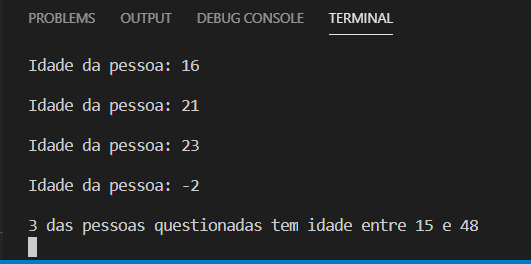
### Descrição e abordagem do problema

Desenvolver um programa que receba um número indefinido de idades e que mostre na consola a quantidade de pessoas que 15 < = idade < 48.

### Fluxograma

### Pseudocódigo

### Testes



## Questão **3**

### **3.3.1** Descrição e abordagem do problema

Criação de uma aplicação capaz de calcular e informar o número de estudantes entrevistados , a percentagem de estudantes por género e quantidade de estudantes do género masculino com menos de 20 anos que não gostam do curso que estão a frequentar.

### Fluxograma

### 3.3.3 Pseudocódigo

### Testes

## Questão **4**

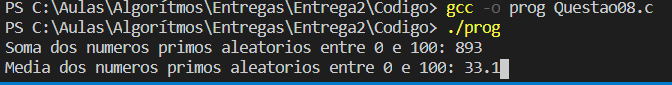
### 3.4.1 Descrição e abordagem do problema

Criação de um programa capaz de gerar de forma automática e aleatória 70 números inteiros positivos entre 0 e 100. Apresentação na consola a soma e média dos números primos existentes no conjunto criado.

### Fluxograma

### 3.4.3 Pseudocódigo

### 3.4.4 Testes

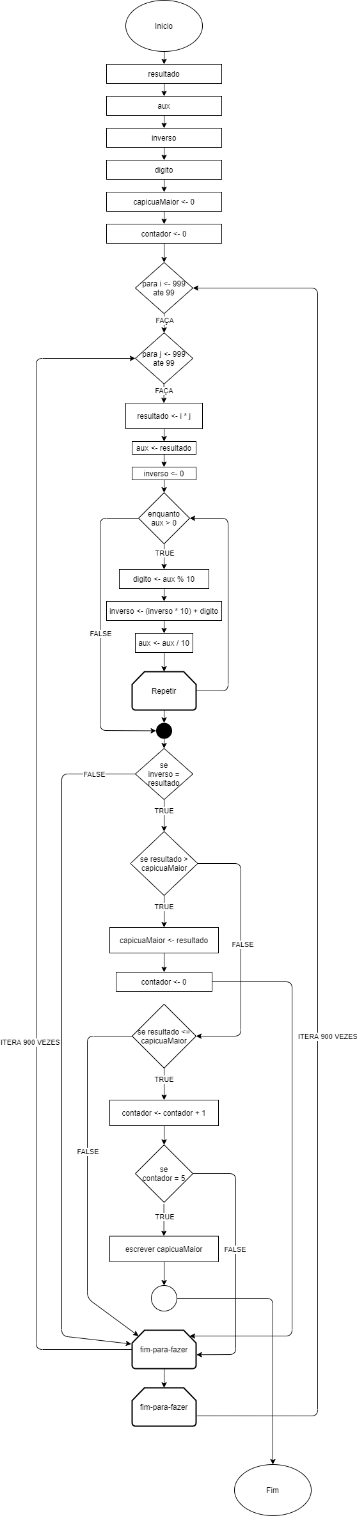


* 1. **Questão** **5**

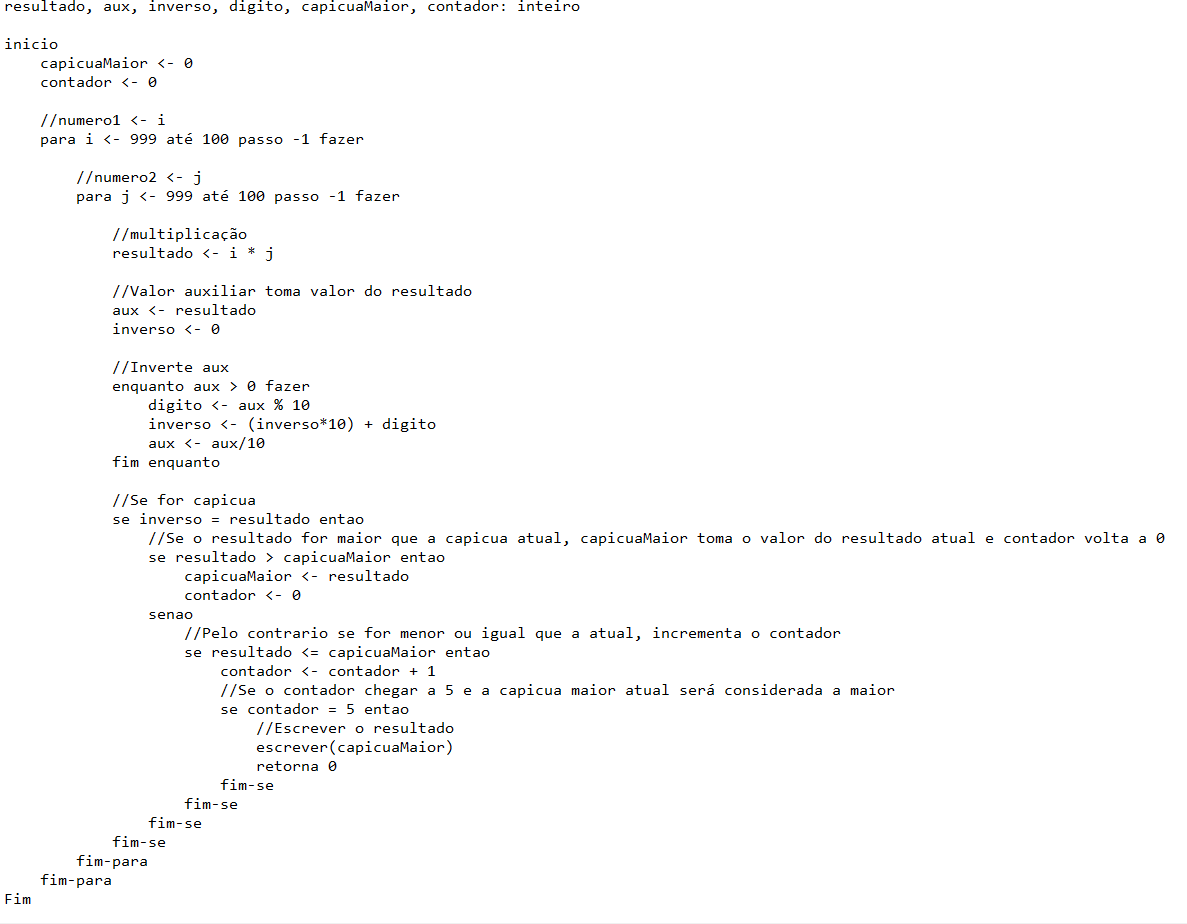
**3.5.1 Descrição e abordagem do problema**

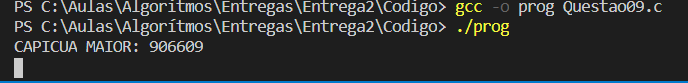
Criação de uma aplicação que encontre o maior número palíndromo resultante do produto entre dois números de três dígitos.

* + 1. **Fluxograma**



**3.5.3 Pseudocódigo**



**3.5.4 Testes**

# Funções e Procedimentos

## Questão 1

### Descrição e abordagem do problema

Reimplementar a questão 5 da entrega 1 para que o somatório seja feito dentro de uma sub-rotina.

## Funções e procedimentos desenvolvidos (assinaturas documentadas e explicadas).

//Assinatura de funcao

int somatorioN(int x);

int somatorioN(int x)

{

    int somatorio = 0;

    for (int i = 1; i <= x; i++)

    {

        somatorio += i;

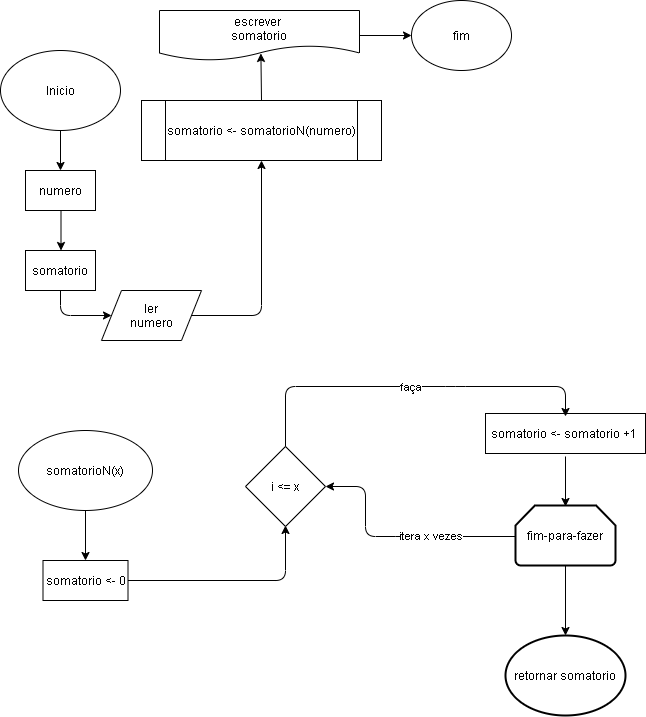
    }

    return (somatorio);

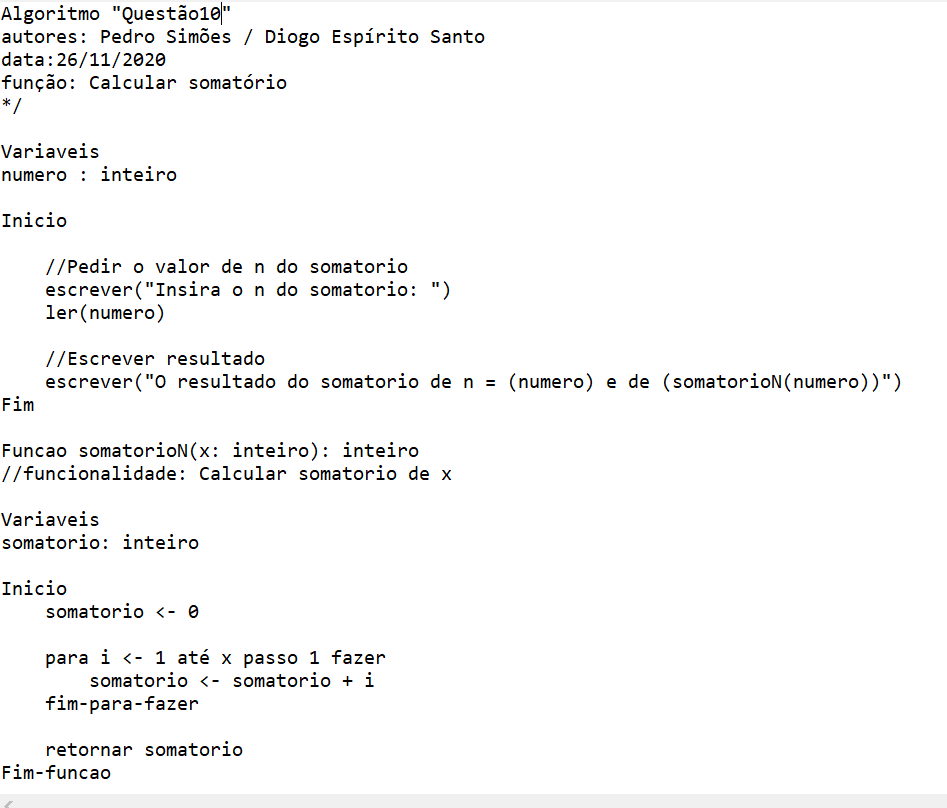
}

Sub-Rotina criada para calcular o Somatório.

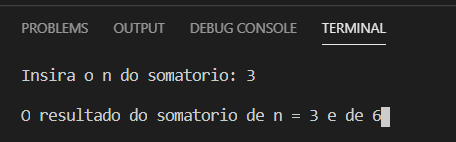
* + 1. **Fluxograma**



* + 1. **Pseudocódigo**



* + 1. **Testes**



## Questão 2

### 4.2.1 Descrição e abordagem do problema

Reimplementar a questão anterior de forma recursiva

## 4.2.2 Funções e procedimentos desenvolvidos (assinaturas documentadas e explicadas).

int somatorioN(int x);

int somatorioN(int x)

{

    int somatorio = 0;

    //Ate o x (numero) ser igual a 0, adicionar sempre o numero anterior (numero - 1)

    if (x == 0)

    {

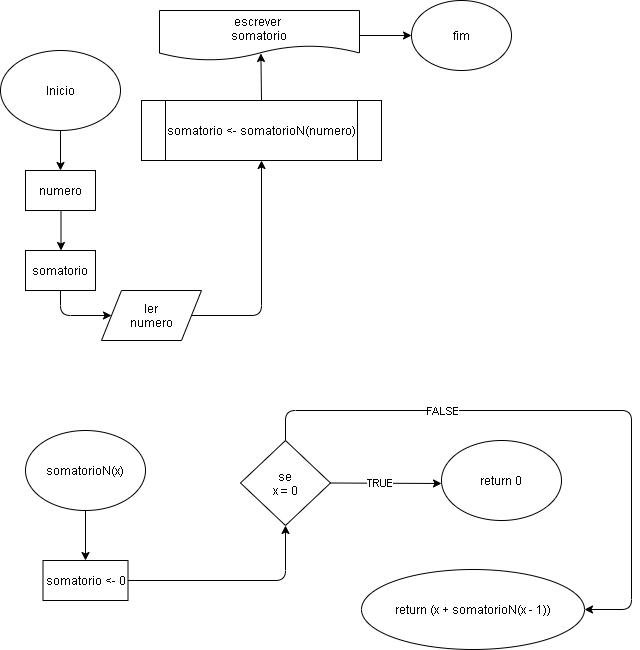
        return 0;

    }

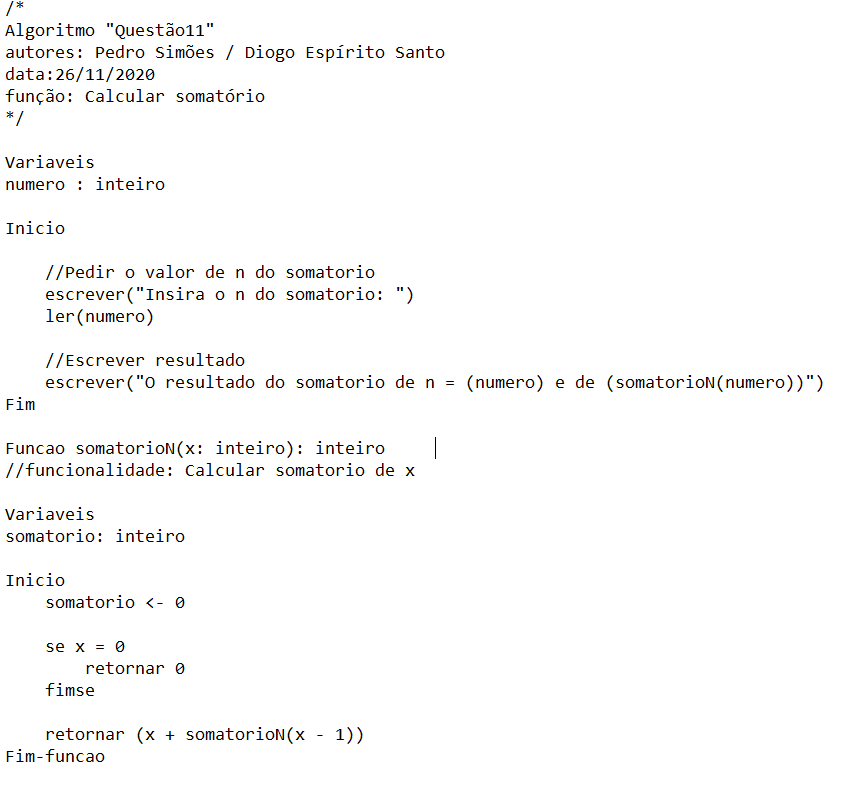
    return x + somatorioN(x - 1);

Sub-Rotina criada para calcular o Somatório.

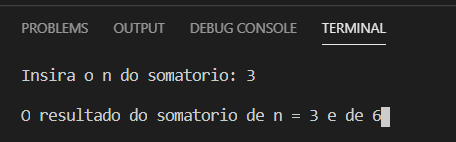
* + 1. **Fluxograma**



* + 1. **Pseudocódigo**



* + 1. **Testes**



## Questão 3

### Descrição e abordagem do problema

Criação de uma função que receba um número e que verifique se o mesmo é palíndromo ou não.

## Funções e procedimentos desenvolvidos (assinaturas documentadas e explicadas).

int palindromo(int x);

int palindromo(int x)

{

    //Declaracao de variaveis

    int aux, inverso = 0, digito;

    aux = x;

    //Inverte aux

    while (aux > 0)

    {

        digito = aux % 10;

        inverso = (inverso \* 10) + digito;

        aux = aux / 10;

    }

    //Se inverso for igual ao numero (neste caso, x) entao o numero e palindromo

    if (inverso == x)

    {

        return 1;

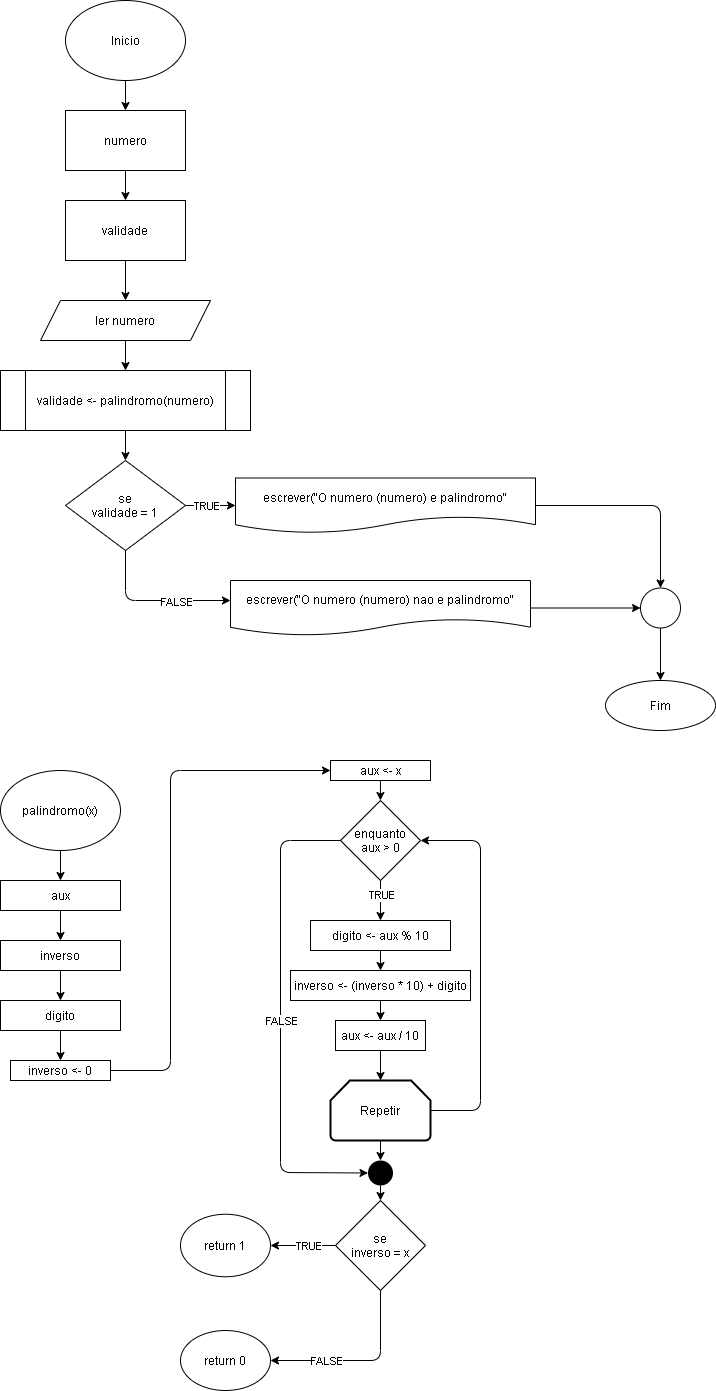
    }

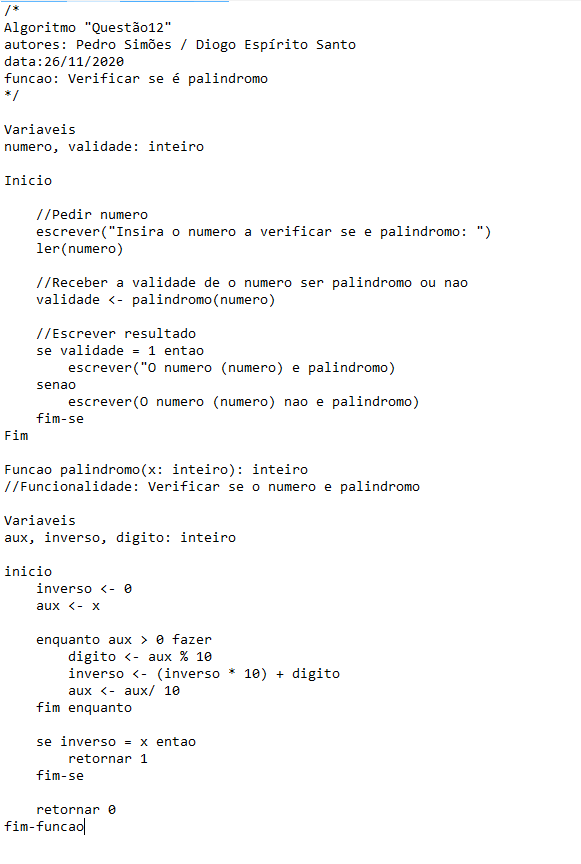
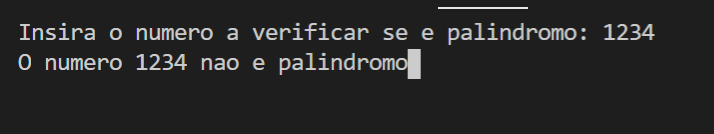
    return 0;

}

Sub-Rotina criada para verificar se o número é palíndromo ou não.

* + 1. **Fluxograma**



* + 1. **Pseudocódigo**
    2. **Testes**

## Questão 4

### Descrição e abordagem do problema

Reimplementar a questão 8 da entrega 2 para que o teste de verificação de primo seja feito dentro de uma função.

## Funções e procedimentos desenvolvidos (assinaturas documentadas e explicadas).

char verificacaoPrimo(int x);

char verificacaoPrimo(int x)

{

    int j;

    //Verificacao se e primo

    for (j = 2; j <= x; j++)

    {

        //Se o numeroRand for divisivel pelo j atual (sem contar quando j == numeroRand), e primo

        if (x % j == 0)

        {

            break;

        }

    }

    //Confirmacao de numero primo e incrementacao na soma dos numeros primos

    if (x != 1 && x == j)

    {

        return 's';

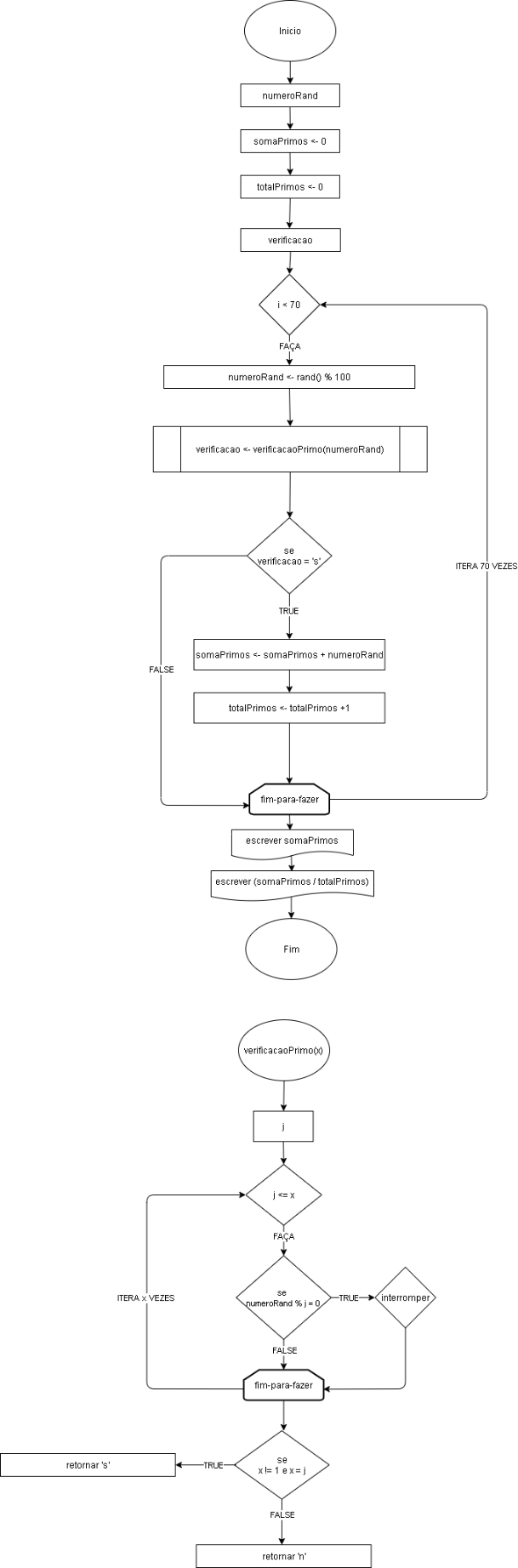
    }

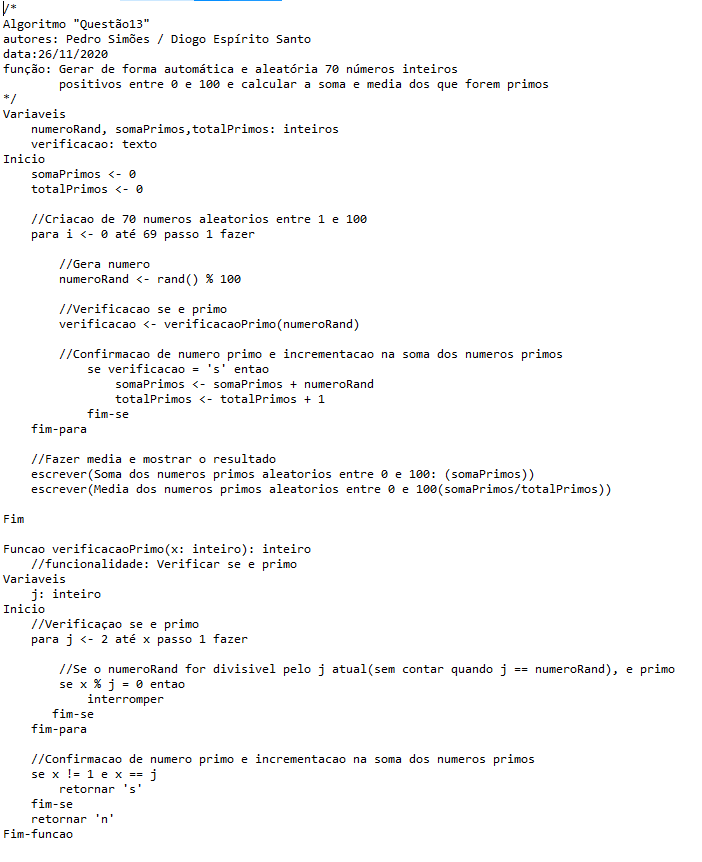
    return 'n';

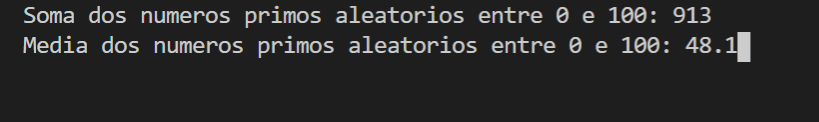
}

Sub-Rotina criada para verificar se o número é primo.

* + 1. **Fluxograma**



* + 1. **Pseudocódigo**
    2. **Testes**



## Questão 5

### Descrição e abordagem do problema

Implementar uma função, que devolve a quantidade de primos existentes entre dois inteiros (i.e. limiteInferior e limiteSuperior).

## 4.5.2 Funções e procedimentos desenvolvidos (assinaturas documentadas e explicadas).

int quantidadePrimos(int x, int y);

int quantidadePrimos(int x, int y)

{

    int somaPrimos = 0, j;

    for (int i = x; i <= y; i++)

    {

        //Verificacao se e primo

        for (j = 2; j <= i; j++)

        {

            //Se o numeroRand for divisivel pelo j atual (sem contar quando j == numeroRand), e primo

            if (i % j == 0)

            {

                break;

            }

        }

        //Confirmacao de numero primo e incrementacao na soma dos numeros primos

        if (i != 1 && i == j)

        {

            somaPrimos++;

        }

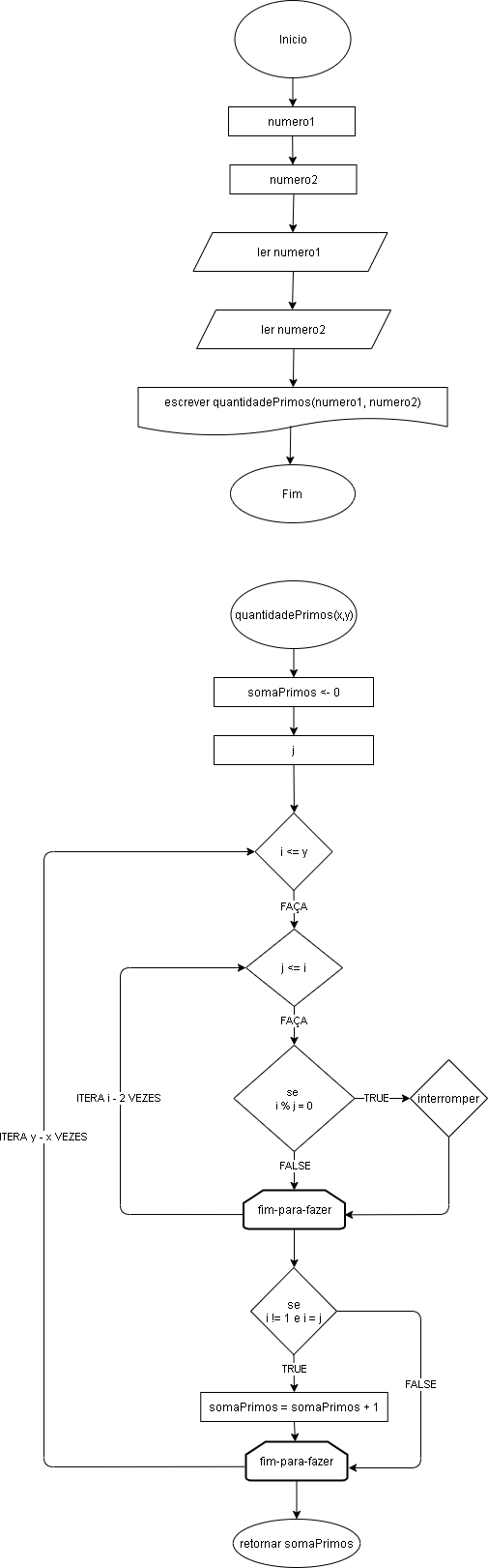
    }

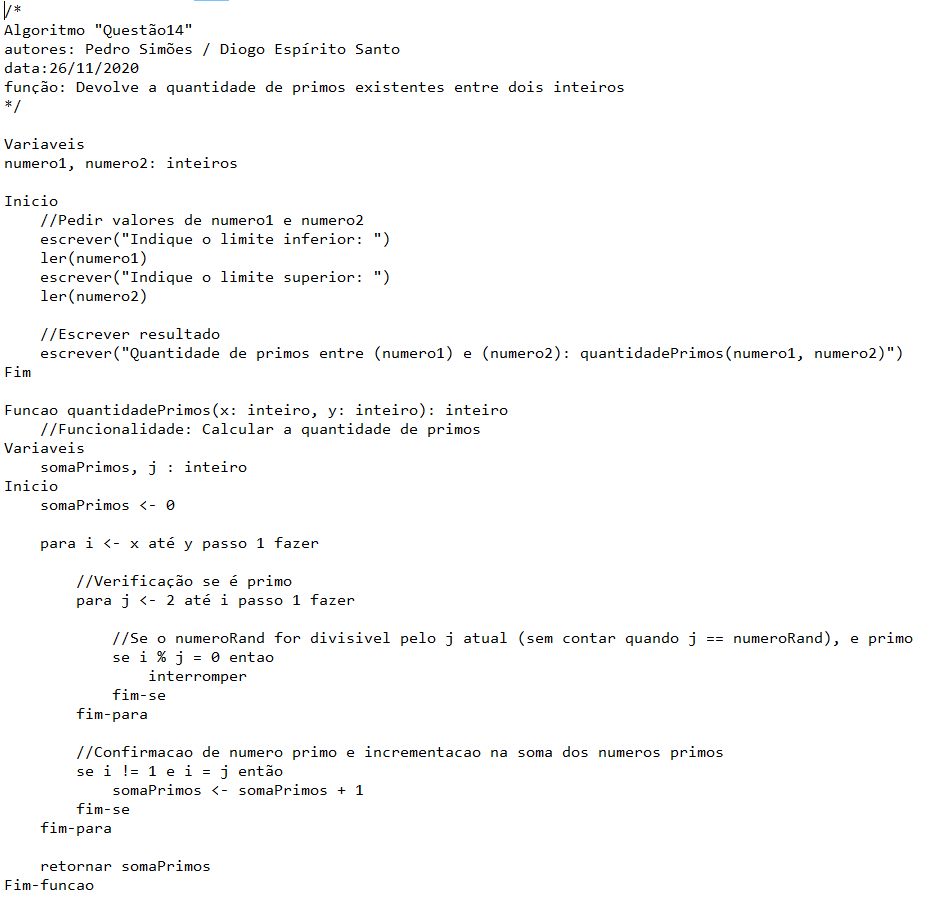
    return (somaPrimos);

}

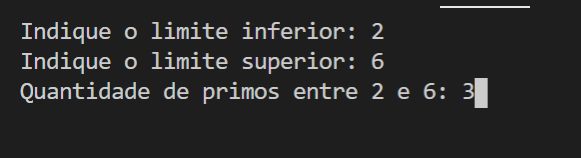
Sub-Rotina criada para calcular a quantidade de primos.

* + 1. **Fluxograma**



**4.5.4 Pseudocódigo**

* + 1. **Testes**



## Questão 6

### Descrição e abordagem do problema

Implementação de uma função que recebe um texto e que devolve o número de ocorrências de um determinado caracter no texto.

## Funções e procedimentos desenvolvidos (assinaturas documentadas e explicadas).

int verificarLetra(char \*x, char y);

int verificarLetra(char \*x, char y)

{

    int contador = 0;

    while (\*x != '\0')

    {

        if (\*x == y)

        {

            contador++;

        }

        //Passar para letra seguinte

        x++;

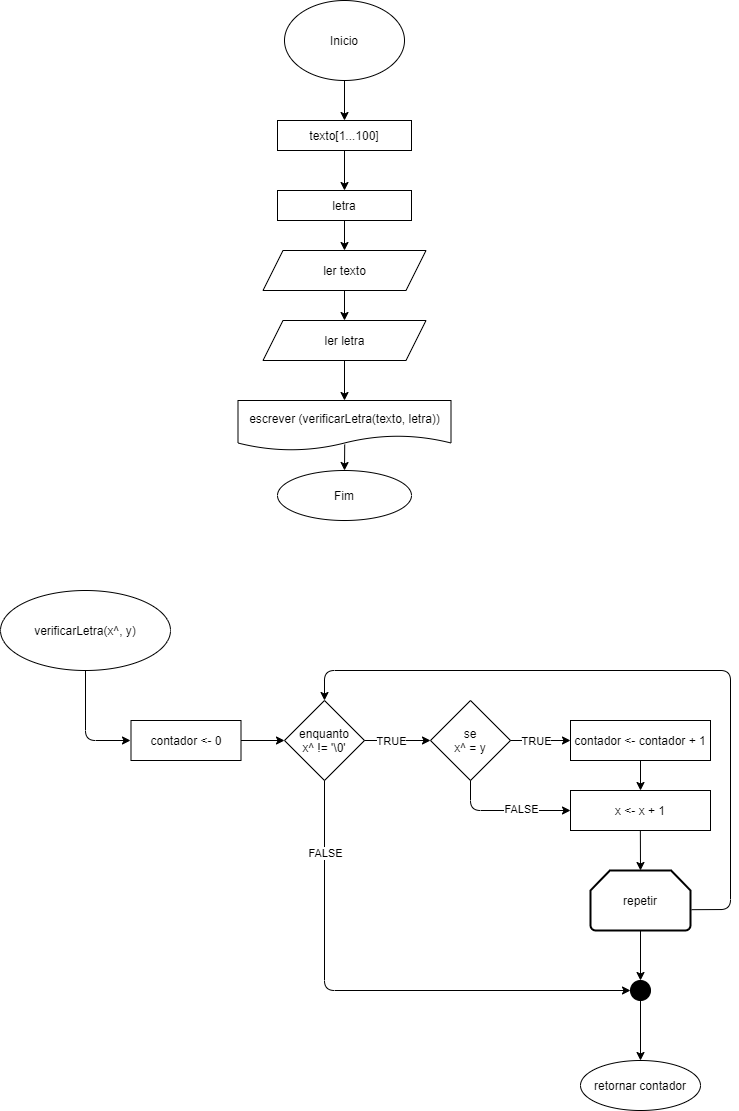
    }

    return contador;

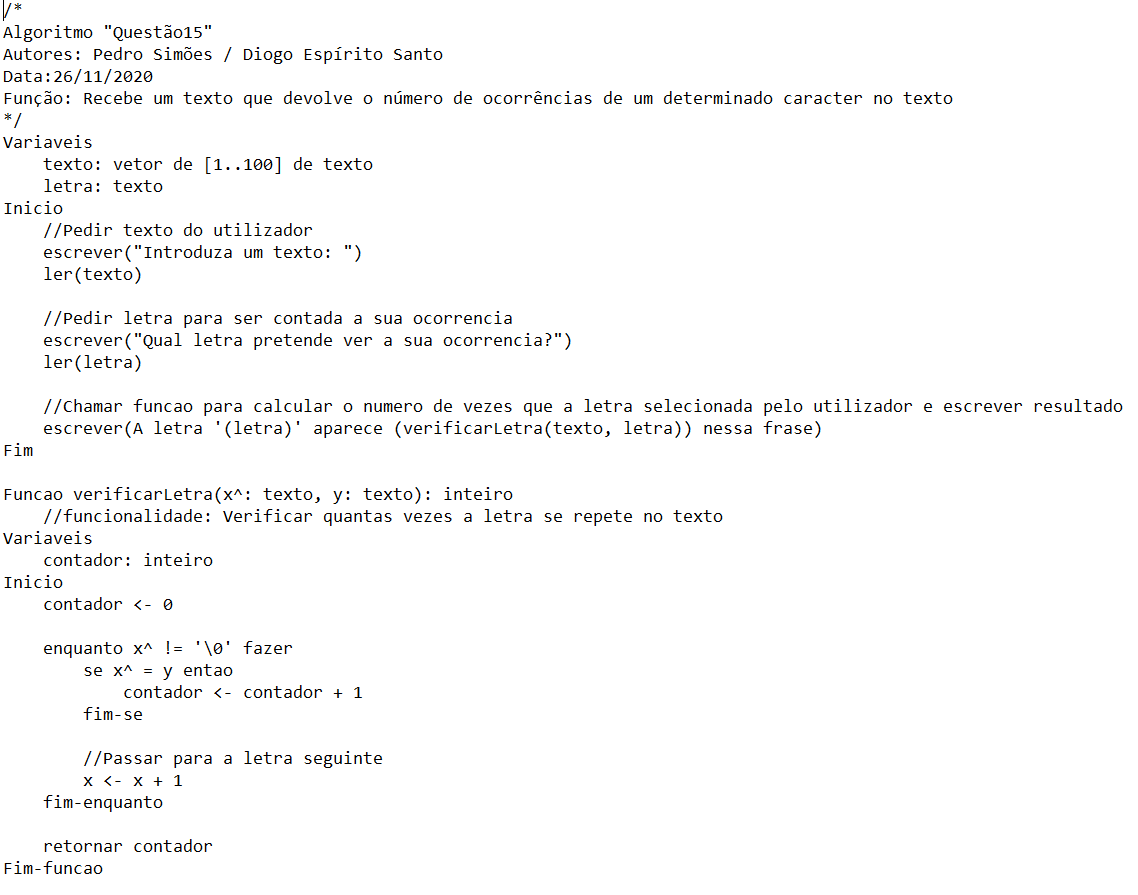
}

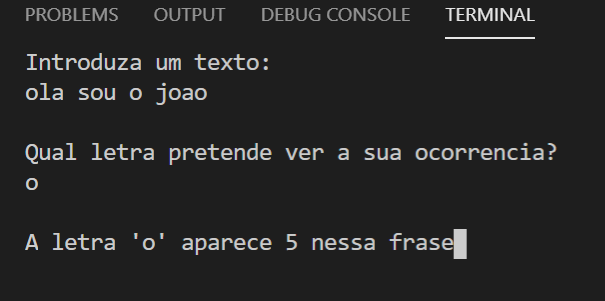
Sub-Rotina criada para contar o número de ocorrências de um determinado caracter no texto.

* + 1. **Fluxograma**



**4.6.4 Pseudocódigo**



**4.6.5 Testes**

## Questão 7

### Descrição e abordagem do problema

Implementação de uma função que receba um texto e que conte o número de palavras existentes no texto.

## 4.7.2 Funções e procedimentos desenvolvidos (assinaturas documentadas e explicadas).

int contarEspacos(char\* x);

int contarEspacos(char\* x)

{

    int contador = 0;

    while (\*x != '\0')

    {

        if (\*x == ' ')

        {

            contador++;

        }

        //Passar para letra seguinte

        x++;

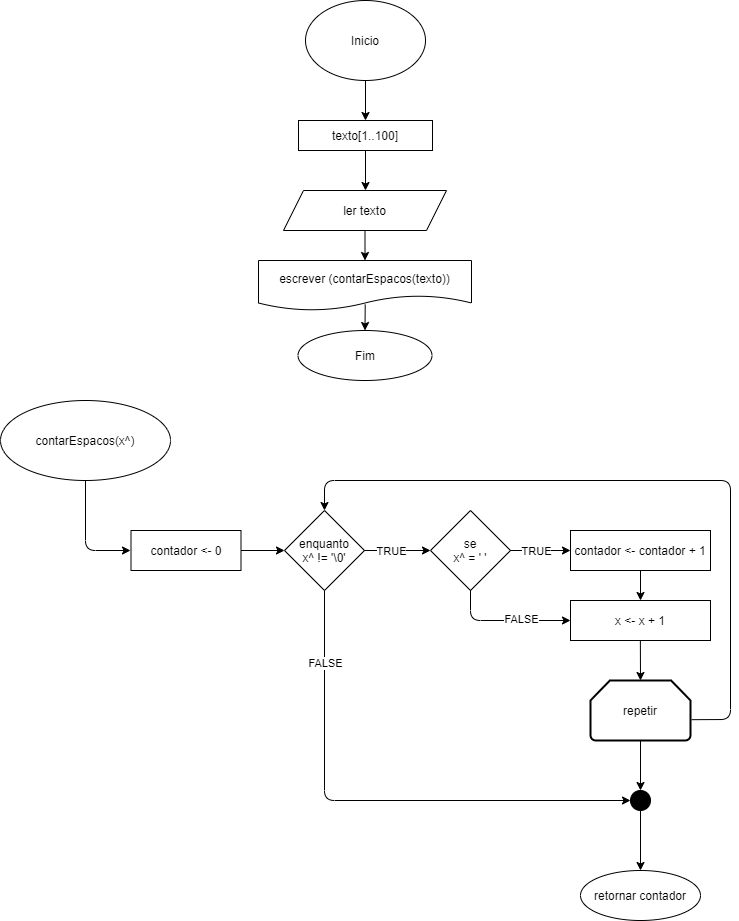
    }

    return contador;

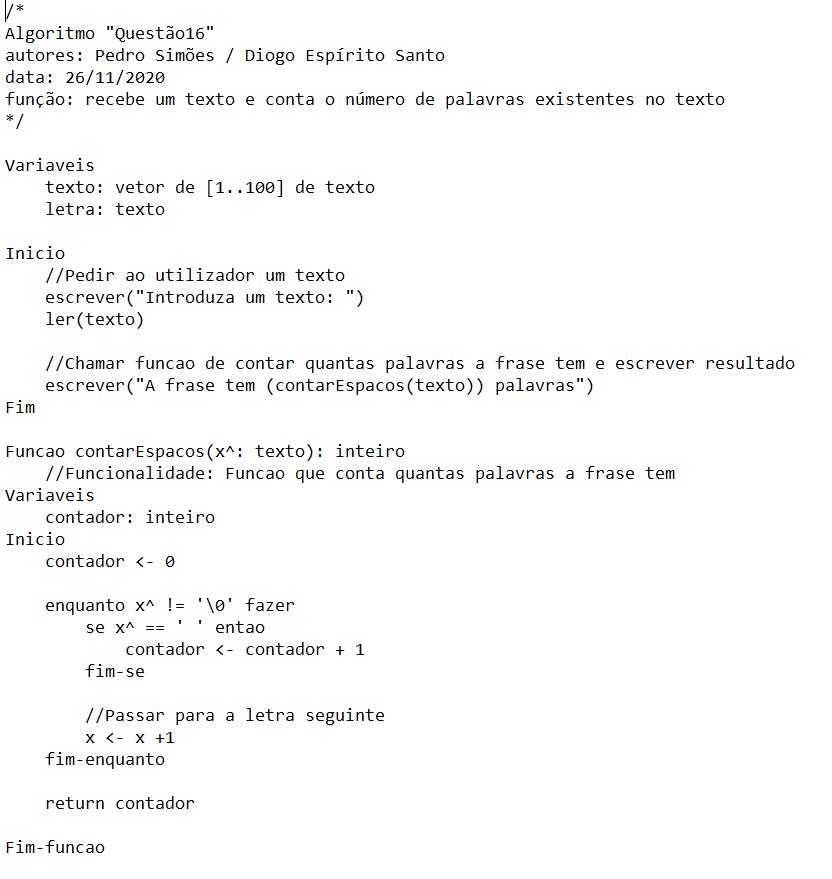
}

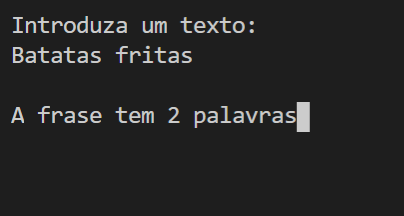
Sub-Rotina criada para contar o numero de palavras existentes no texto.

* + 1. **Fluxograma**



**4.7.4 Pseudocódigo**



**4.7.5 Testes**

## Questão 8

### Descrição e abordagem do problema

Implementação de um procedimento que receba um texto e que lhe retire todos os espaços em branco.

## Funções e procedimentos desenvolvidos (assinaturas documentadas e explicadas).

void apagarEspacos(char \*x, char \*y);

void apagarEspacos(char \*x, char \*y)

{

    while (\*y != '\0')

    {

        if (\*y != ' ')

        {

            \*x = \*y;

            x++;

        }

        y++;

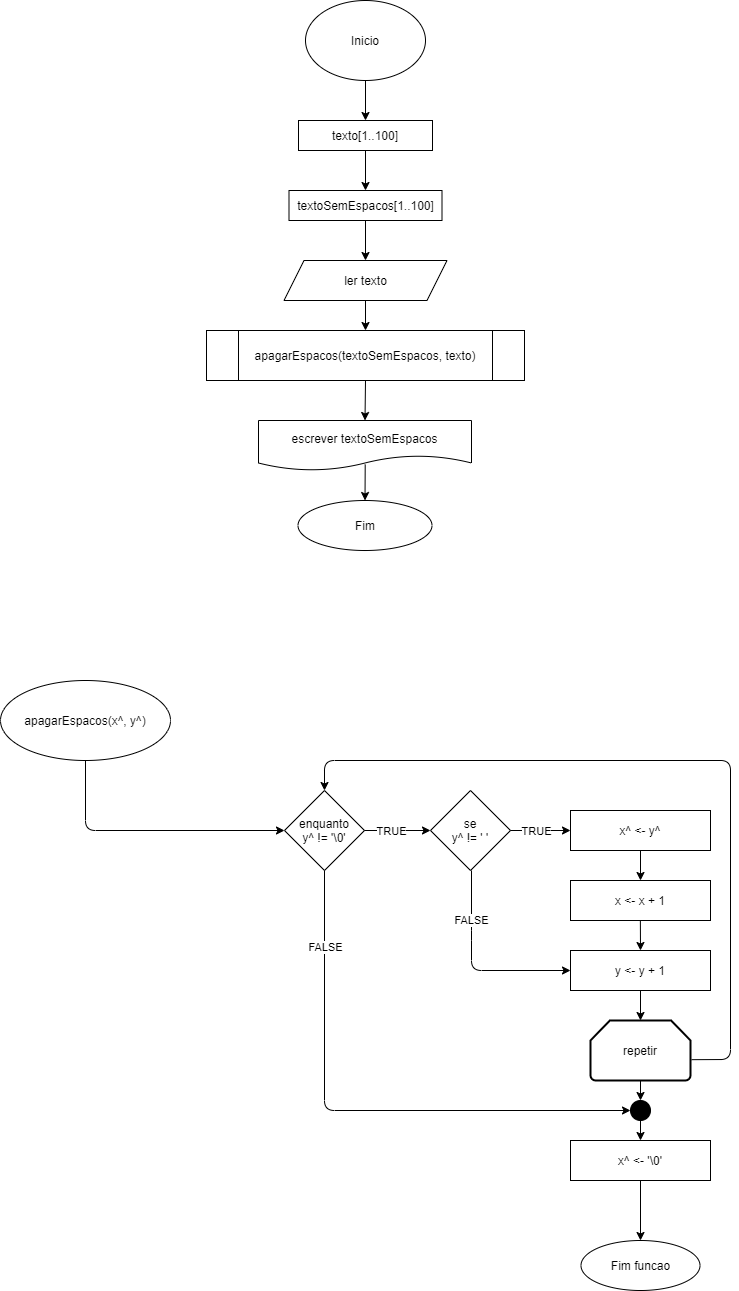
    }

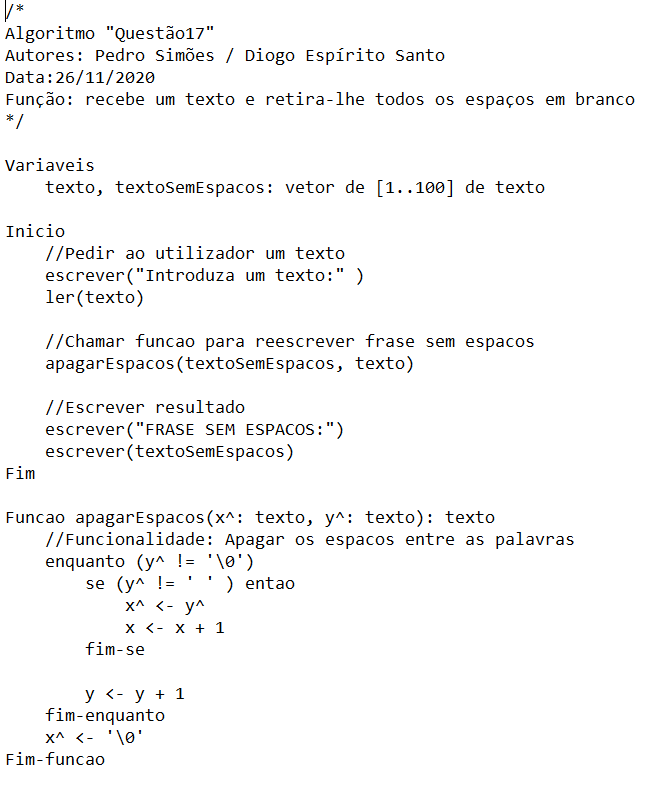
    \*x = '\0';

}

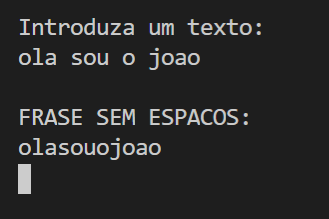
Sub-Rotina criada para retirar os espaços em branco ao texto.

* + 1. **Fluxograma**



**4.8.4 Pseudocódigo**

* + 1. **Testes**



# Arrays

**5.1 Questão 1**

**5.1.1 Descrição e abordagem do problema**

Desenvolver um programa que faça conversões entre as duas escalas de temperaturas, Kelvin e Celsius com base em dois valores de entrada: a escala de conversão pretendida e a lista das temperaturas.

## Funções e procedimentos desenvolvidos (assinaturas documentadas e explicadas).

void mostrarTabela(int temperaturas[], int tamanho, char conversao);

void mostrarTabela(int temperaturas[], int tamanho, char conversao){

    //Caso utilizador nao chegar a atribuir nenhuma temperatura

    if (tamanho == 0)

    {

        printf("\n!!!!NENHUMA TEMPERATURA INTRODUZIDA!!!!\n");

    }

    else

    {

        //parte de cima da tabela

        printf("\n---------------------------------------\n");

        printf("|           Celsius|            Kelvin|\n");

        printf("---------------------------------------\n");

        //Corpo da tabela com as respetivas temperaturas

        if (conversao == 'k' || conversao == 'K'){

            for (int i = 0; i < tamanho; i++){

                printf("|%18d|%18d|\n", temperaturas[i], temperaturas[i] + 273);

            }

        }

        else if (conversao == 'c' || conversao == 'c'){

            for (int i = 0; i < tamanho; i++){

                printf("|%18d|%18d|\n", temperaturas[i] - 273, temperaturas[i]);

            }

        }

        //Parte de baixo da tabela

        printf("---------------------------------------\n");

Procedimento criado para mostrar uma tabela com todas as temperaturas.

int menorTemp(int temperaturas[], int tamanho);

int menorTemp(int temperaturas[], int tamanho){

    int menor = 99999;

    for (int i = 0; i < tamanho; i++)

    {

        if (menor > temperaturas[i]) menor = temperaturas[i];

    }

    return menor;

Função criada para calcular a menor temperatura.

int maiorTemp(int temperaturas[], int tamanho);

int maiorTemp(int temperaturas[], int tamanho){

    int maior = -99999;

    for (int i = 0; i < tamanho; i++)

    {

        if (maior < temperaturas[i]) maior = temperaturas[i];

    }

    return maior;

}

Função criada para calcular a maior temperatura.

float mediaTemp(int temperaturas[], int tamanho);

float mediaTemp(int temperaturas[], int tamanho){

    int somaTotal = 0;

    //Somar todas a temperaturas

    for (int i = 0; i < tamanho; i++)

    {

        somaTotal += temperaturas[i];

    }

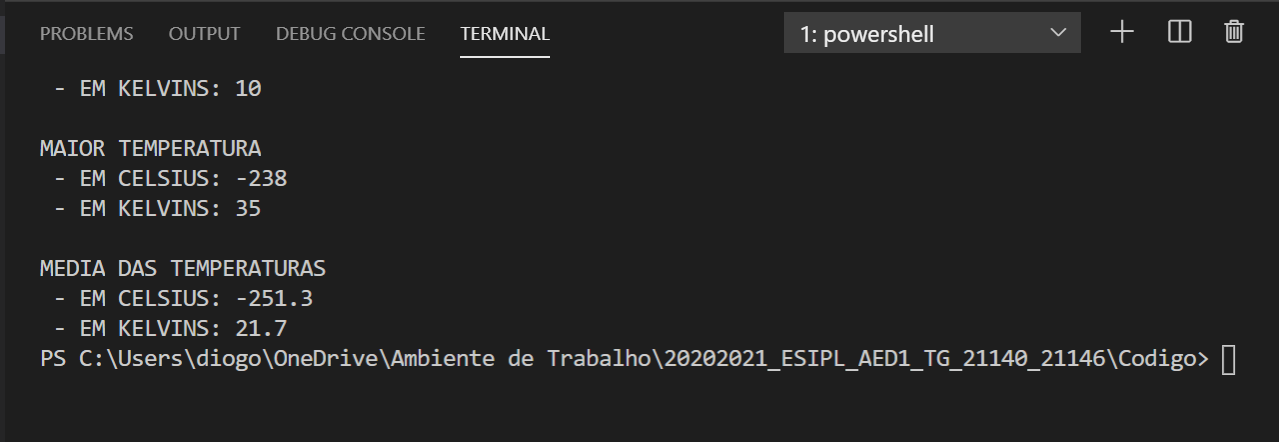
    //Calcular media e devolver esse valor

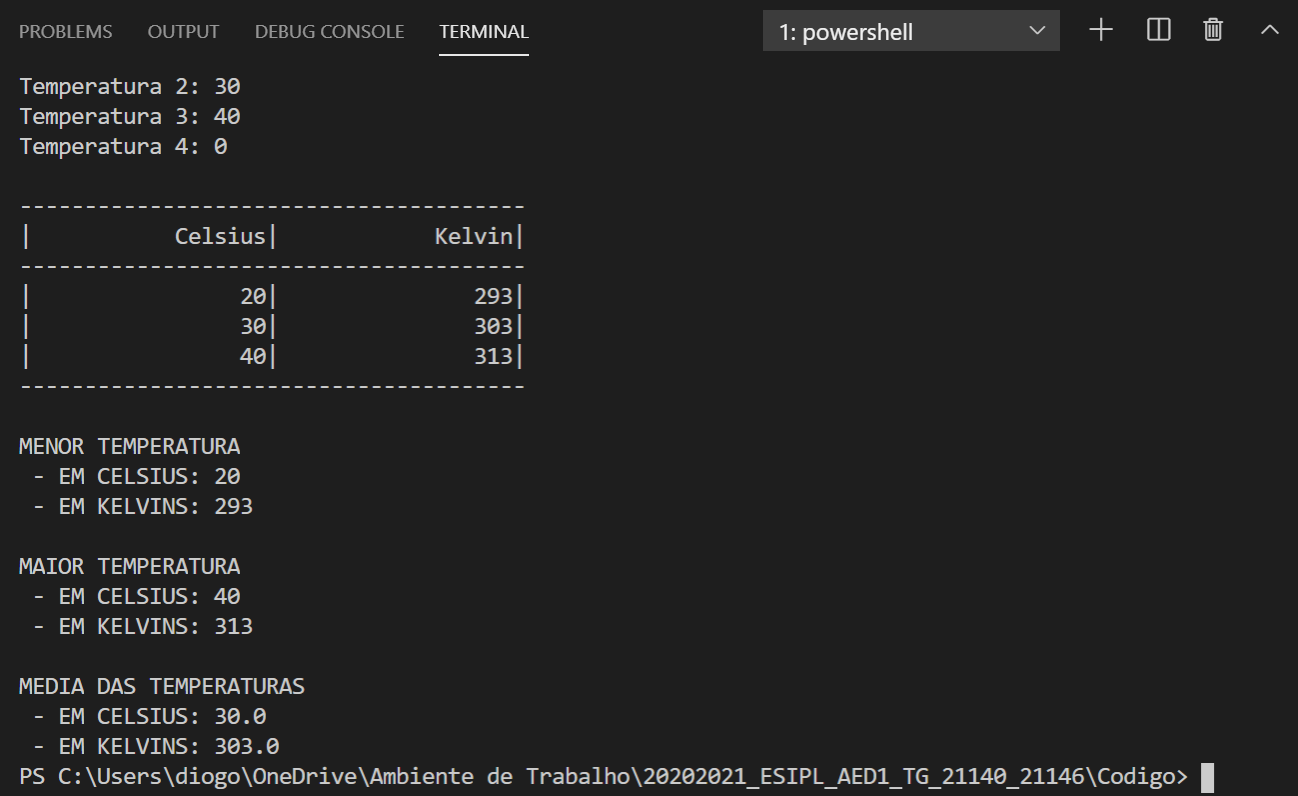
    return ((float)somaTotal / tamanho);

}

Função criada para calcular a média das temperaturas.

## Testes





* 1. **Questão 2**

**5.2.1** **Descrição e abordagem do problema**

Desenvolver um programa capaz de gerar de forma automática e aleatória números inteiros positivos, em que a quantidade de números a gerar seja sorteada entre [90;150].

## Funções e procedimentos desenvolvidos (assinaturas documentadas e explicadas).

void mostrarMenu();

void mostrarMenu(){

    printf("\t\t\t\t\t/----------------------------\\\n");

    printf("\t\t\t\t\t|-----NUMEROS ALEATORIOS-----|\n");

    printf("\t\t\t\t\t\\----------------------------/\n");

    printf("\n\t\t\t     Este programa gerou 30 numeros aleatorios entre 90 e 150!\n");

Procedimento criado para mostrar um menu

void verificarOpcao(char opcao, int numeros[], int tamanho);

void verificarOpcao(char opcao, int numeros[], int tamanho){

    if (opcao == 'm' || opcao == 'M') maiorMenor(numeros, tamanho);

    else if (opcao == 'a' || opcao == 'A') amplitude(numeros, tamanho);

    else if (opcao == 'i' || opcao == 'I') imparesOrdDes(numeros, tamanho);

    else if (opcao == 'v' || opcao == 'V') verConjunto(numeros, tamanho);

}

Procedimento criado para verificar escolho do utilizador

void maiorMenor(int numeros[], int tamanho);

void maiorMenor(int numeros[], int tamanho){

    int maior = -1;

    int menor = 500;

    for (int i = 0; i < tamanho; i++)

    {

        if (maior < numeros[i]) maior = numeros[i];

        if (menor > numeros[i]) menor = numeros[i];

    }

    printf("\nMAIOR NUMERO: %d\nMENOR NUMERO: %d", maior, menor);

}

Procedimento criado para ver o maior e o menor número.

void amplitude(int numeros[], int tamanho);

void amplitude(int numeros[], int tamanho){

    int maior = -1;

    int menor = 160;

    for (int i = 0; i < tamanho; i++)

    {

        if (maior < numeros[i]) maior = numeros[i];

        if (menor > numeros[i]) menor = numeros[i];

    }

Procedimento criado para calcular a amplitude.

void imparesOrdDes(int numeros[], int tamanho);

void imparesOrdDes(int numeros[], int tamanho){

    int impares[TAM], contador = 0, aux;

    for (int i = 0; i < tamanho; i++)

    {

        //se for impar, armazenar valor atual num array impares de indice contador

        if (numeros[i] % 2 == 1)

        {

            impares[contador] = numeros[i];

            contador++;

        }

    }

Procedimento criado para ordenar de forma decrescente os números impares.

void verConjunto(int numeros[], int tamanho);

void verConjunto(int numeros[], int tamanho){

    printf("\nCONJUNTO GERADO:\n");

    for (int i = 0; i < tamanho; i++)

    {

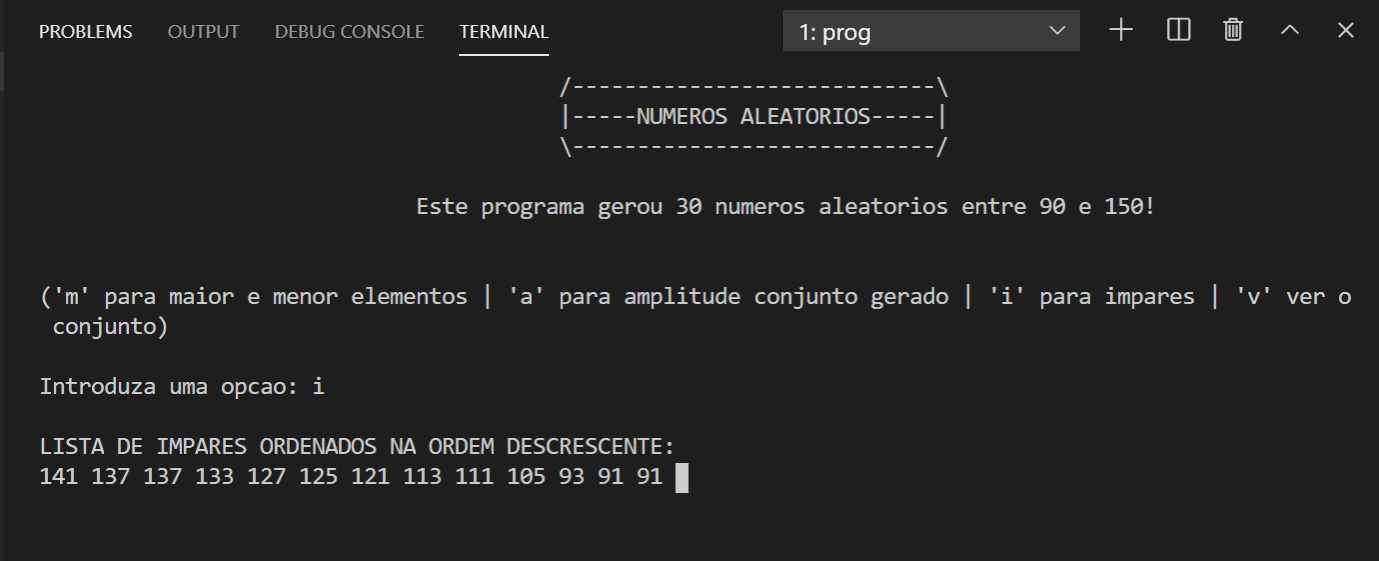
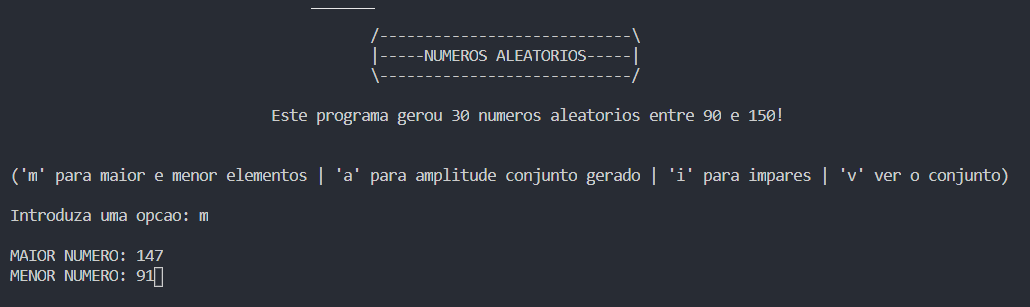
        printf("%d ", numeros[i]);

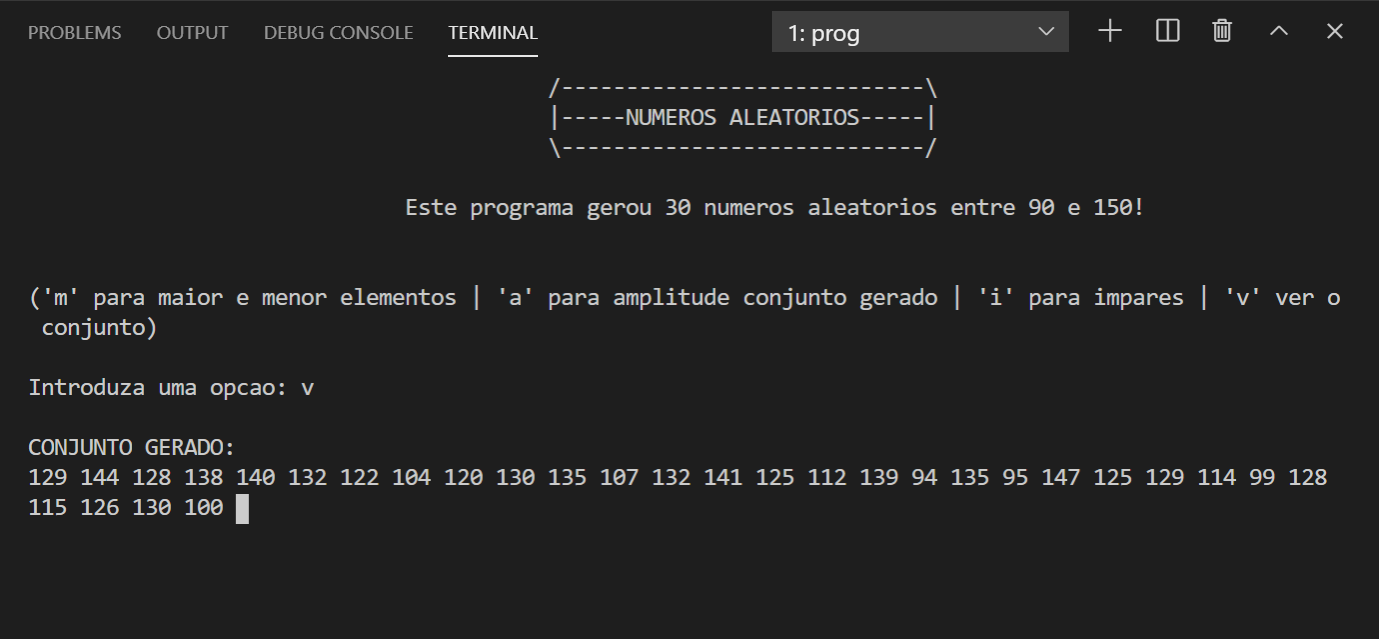
    }

}

Procedimento criado para ver o conjunto gerado.

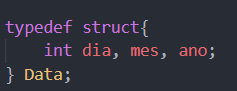
## Testes



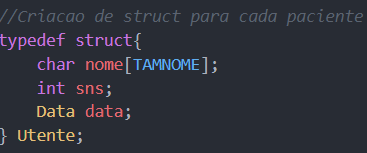


# Questão 2

## Estruturas desenvolvidas (explicadas)

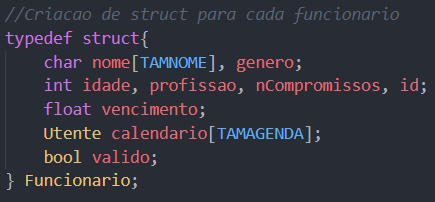


Estrutura de dados que armazena um dia, um mês, um ano



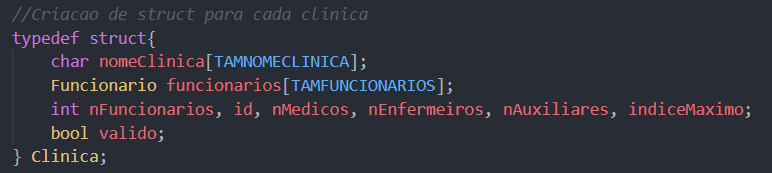
Estrutura que armazena informação de utentes, contém:

* Nome do utente
* número SNS
* data de consulta



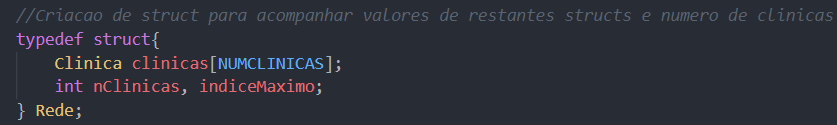
Estrutura que armazena a informação dos funcionários, contém:

* Nome do funcionário
* Género
* Idade
* Profissão (médico, enfermeiro, auxiliar)
* Numero de consultas
* ID
* Vencimento
* Array de utentes com dados para cada utente, chamado calendário
* Booleano que indica se este é válido ou não (se não for é como se tivesse “apagado”)



Estrutura que armazena a informação sobre uma clínica, contém:

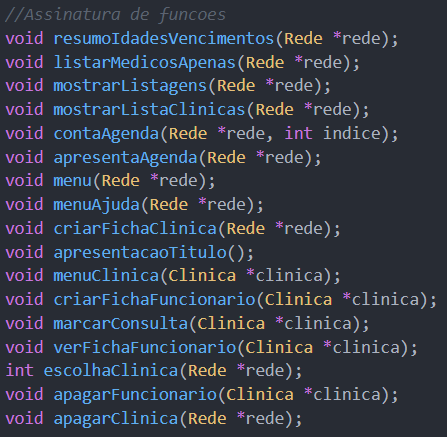
* Nome da clínica
* Array de funcionários com dados para cada funcionário, chamado funcionários
* Numero de funcionários a trabalhar nela
* ID
* Numero de médicos, enfermeiros e auxiliares
* Variável que armazena o indíce mais alto criado
* Booleano que indica se esta é válido ou não (se não for é como se tivesse “apagado”)



Estrutura que armazena dados da rede, contém:

* Array de clinicas com dados para cada clinica
* Numero de clinicas na rede
* Variável que armazena o indíce mais alto criado

## Funções e procedimentos desenvolvidos



## Testes

Estes serão feitos em plena defesa oral.