Relatório

AED/ LP

Aluno: André Filipe Pereira Cerqueira

Professores: Marta Susana Lopes Martinho,

Célio Domingos de Faria Carvalho

Curso Técnico Superior Profissional Aplicações Móveis

Braga, Janeiro, 2021

Índice de Figuras

Índice

1. Introdução

1.1. Contextualização

O presente relatório foi elaborado no âmbito das disciplinas de Linguagens de Programação e Algoritmos e Estruturas de Dados, e destina-se a apresentar todos os trabalhos individuais elaborados no contexto das disciplinas.

1.2. Motivação e Objetivos

Procurei atingir alguns objetivos pessoais, tais como: adquirir conhecimentos na área da programação, adquirir capacidades de desenvolvimento ágil de software, melhorar a organização e estrutura de código.

1.3. Estrutura do Documento

O presente documento está dividido em 5 partes, cada parte refere a uma entrega diferente do trabalho individual, cada entrega é composta por questões diferentes.

2. Instruções de Decisão

2.1. Questão 01

6.3.1. Descrição e abordagem do problema

Solicite a idade de 10 pessoas, apresente a média das idades ímpar e a soma das idades par abaixo dos 13 anos.

6.3.2. Fluxograma

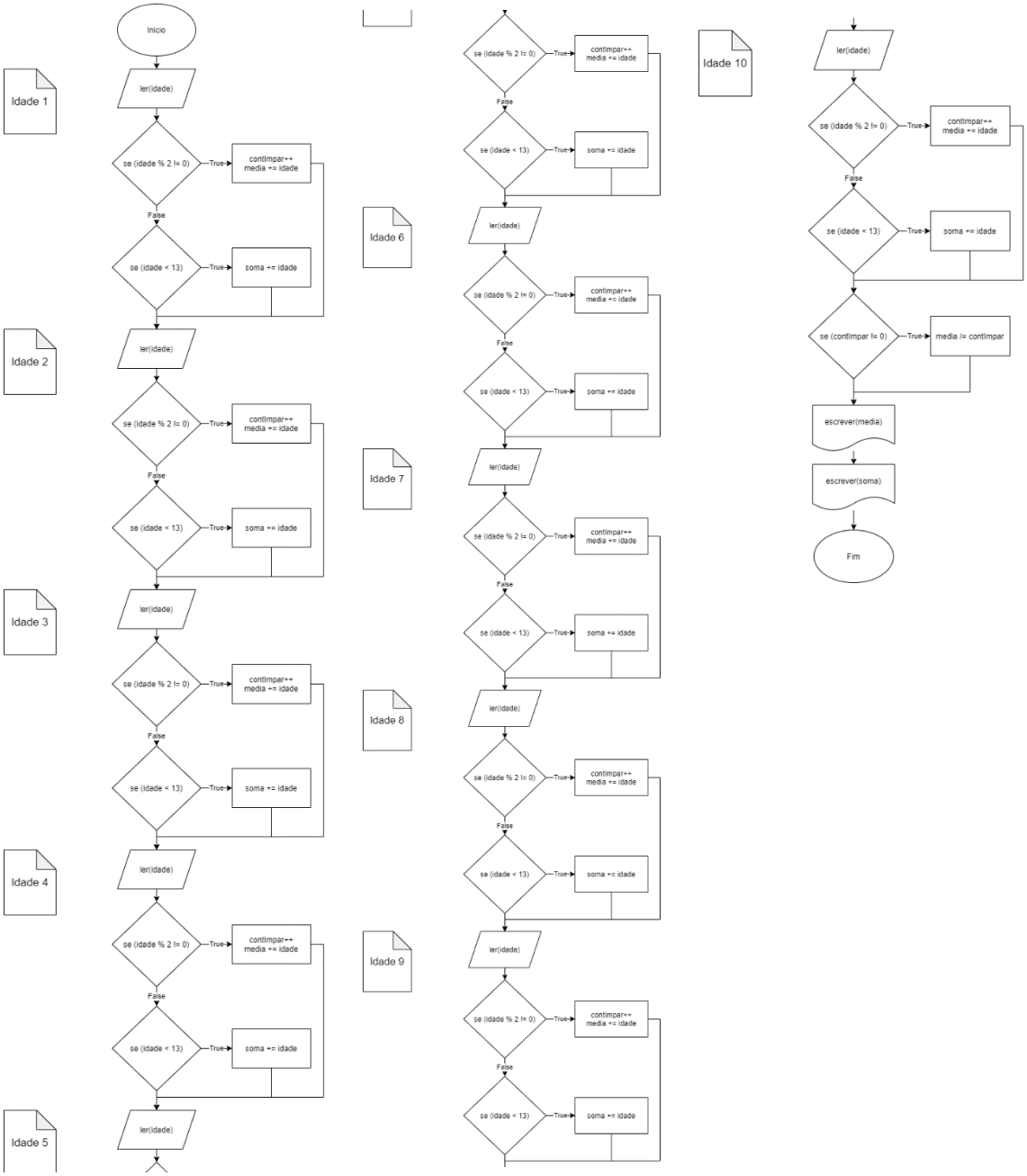


Figura 1, 2 e 3 - Fluxograma Questão01

6.3.3. Pseudocódigo

```
Algoritmo "Questao01"

    // Função: Calcular a média de ídades ímpares e a soma de ídades ínferiores a 13 e pares
    // Autor: André Cerqueira
    // Data: 05/11/2020
    // Versão: 1.0

Variaveis

    idade, contImpar, soma: inteiro
    media: real

Inicio

    // definir valores iniciais
    contImpar = 0
    soma = 0
    media = 0

    // ---------------- Pessoa 1 ---------------- \\

    // Inserir Idade
    ler(idade)

    // Verificar se é impar
    se ( idade % 2 != 0 ) entao
        contImpar++
        media += idade

    // Verificar se é par e menor que 13
    senao se (idade < 13) entao
        soma += idade
    fim se
```

```
170
171     // Verificar se foi inserido pelo menos uma idade impar
172     se (contImpar != 0) entao
173         media /= contImpar
174     fim se
175
176     // Resultados
177     escrever(media)
178     escrever(soma)
179
180 Fim
```

Figura 4 e 5 - Pseudocodigo Questão01

6.3.4. Testes

```
Insira a idade da pessoa número [1]: 4
Insira a idade da pessoa número [2]: 8
Insira a idade da pessoa número [3]: 34
Insira a idade da pessoa número [4]: 15
Insira a idade da pessoa número [5]: 18
Insira a idade da pessoa número [6]: 21
Insira a idade da pessoa número [7]: 67
Insira a idade da pessoa número [8]: 4
Insira a idade da pessoa número [9]: 34
Insira a idade da pessoa número [10]: 20

A média das idades ímpares é: 34,33
A soma das idades pares e inferiores a 13 é: 16
--------------------------------
```

Figura 6 - Teste Questão01

2.2. Questão 02

6.3.5. Descrição e abordagem do problema

Solicite os dados de um gato e indicar ao utilizador qual o desconto a atribuir com base no quadro apresentado.

6.3.6. Fluxograma

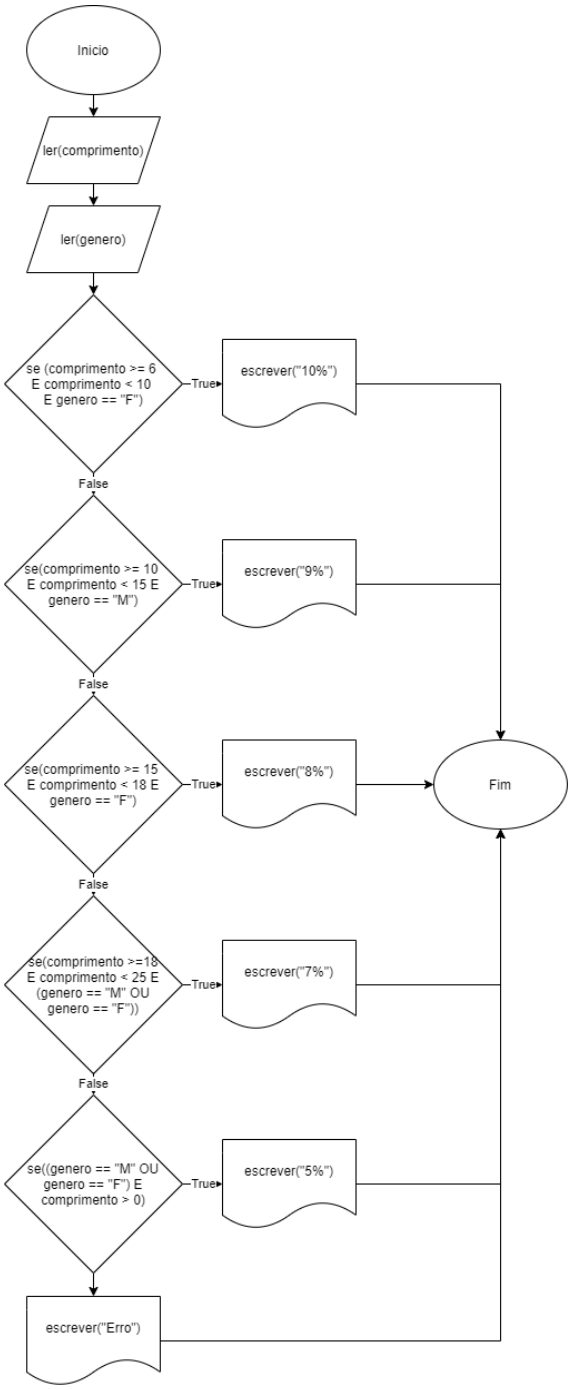


Figura 7 - Fluxograma Questão02

6.3.7. Pseudocódigo

```
Algoritmo "Questao01"

    // Função: Calcular a média de ídades ímpares e a soma de ídades ínferiores a 13 e pares
    // Autor: André Cerqueira
    // Data: 05/11/2020
    // Versão: 1.0

Variaveis

    comp: inteiro
    genero: caracter

Inicio

    // Inserir Dados
    ler(comprimento)
    ler(genero)

    // Condições
    se (comprimento >= 6 E comprimento < 10 E genero == "F") entao
        escrever("10%")
    senao se(comprimento >= 10 E comprimento < 15 E genero == "M") entao
        escrever("9%")
    senao se(comprimento >= 15 E comprimento < 18 E genero == "F") entao
        escrever("8%")
    senao se(comprimento >= 18 E comprimento < 25 E (genero == "M" OU genero == "F")) entao
        escrever("7%")
    senao se((genero == "M" OU genero == "F") E comprimento > 0) entao
        escrever("5%")
    senao
        escrever("Erro")
    fim se
Fim
```

Figura 8 - Pseudocodigo Questão02

6.3.8. Testes

```
Insira o comprimento (cm) do seu gato: 10
Insira o genero (F/M) do seu gato: f

O desconto é de: 5%
--------------------------------
```

```
Insira o comprimento (cm) do seu gato: 9
Insira o genero (F/M) do seu gato: f

O desconto é de: 10%
--------------------------------
```

Figura 9 e 10 - Testes Questão02

3. Instruções de Repetição

3.1. Questão 03

6.3.9. Descrição e abordagem do problema

Solicitar ao utilizador um número natural e apresentar o resultado da soma dos números compreendidos entre 1 e o numero inserido e representar na forma de (1 + 2 + 3 + ...).

6.3.10. Fluxograma

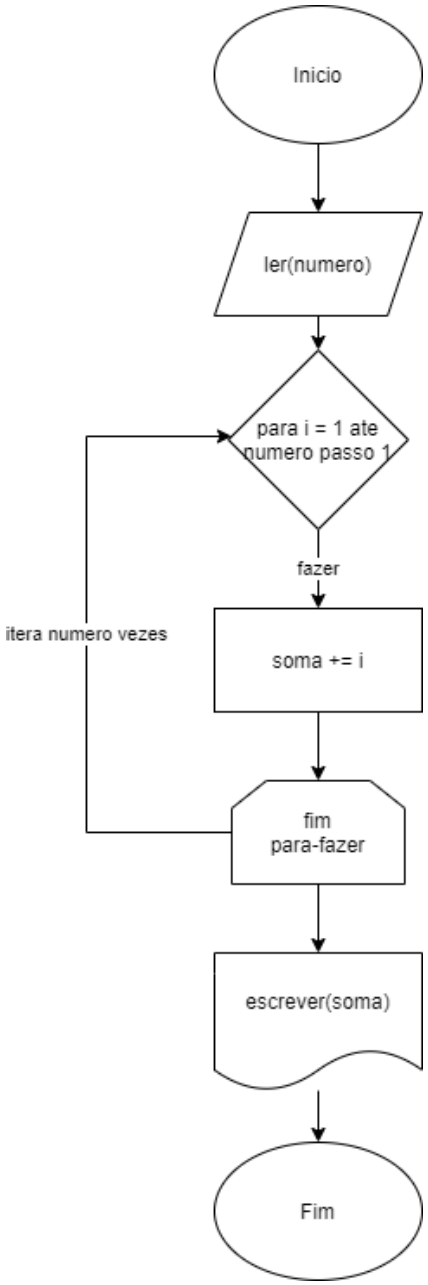


Figura 11 - Fluxograma Questão03

6.3.11. Pseudocódigo

```
Algoritmo "Questao03"

    // Função: Mostrar ao utilizador o resultado da soma dos numeros entre 1 e o numero inserido
    // Autor: André Cerqueira
    // Data: 16/11/2020
    // Versão: 1.0

Variaveis

    numero, i, soma : inteiro

Inicio

    // definir valores iniciais
    soma = 0

    // Inserir dados
    ler(numero)

    // Loop para adicionar os numeros á equação
    para i = 1 ate numero passo 1 fazer
        soma += i
    fimpara

    // Apresentar resultado da soma no final da equação
    escrever(soma)

Fim
```

Figura 12 - Pseudocódigo Questão03

6.3.12. Traçagens e testes

Numero	Resultado		Numero	Resultado
0	0		6	21
1	1		7	28
2	3		8	36
3	6		9	45
4	10		10	55
5	15		11	56

Figura 13 - Traçagem Questão03

```
Insira um número natural: 11
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 = 66
--------------------------------
```

Figura 14 - Teste Questão03

3.2. Questão 04

6.3.13. Descrição e abordagem do problema

Solicitar ao utilizador um número indefinido de idades e apresentar a quantidade de pessoas cuja idade seja igual ou superior a 15 e menor que 48.

6.3.14. Fluxograma

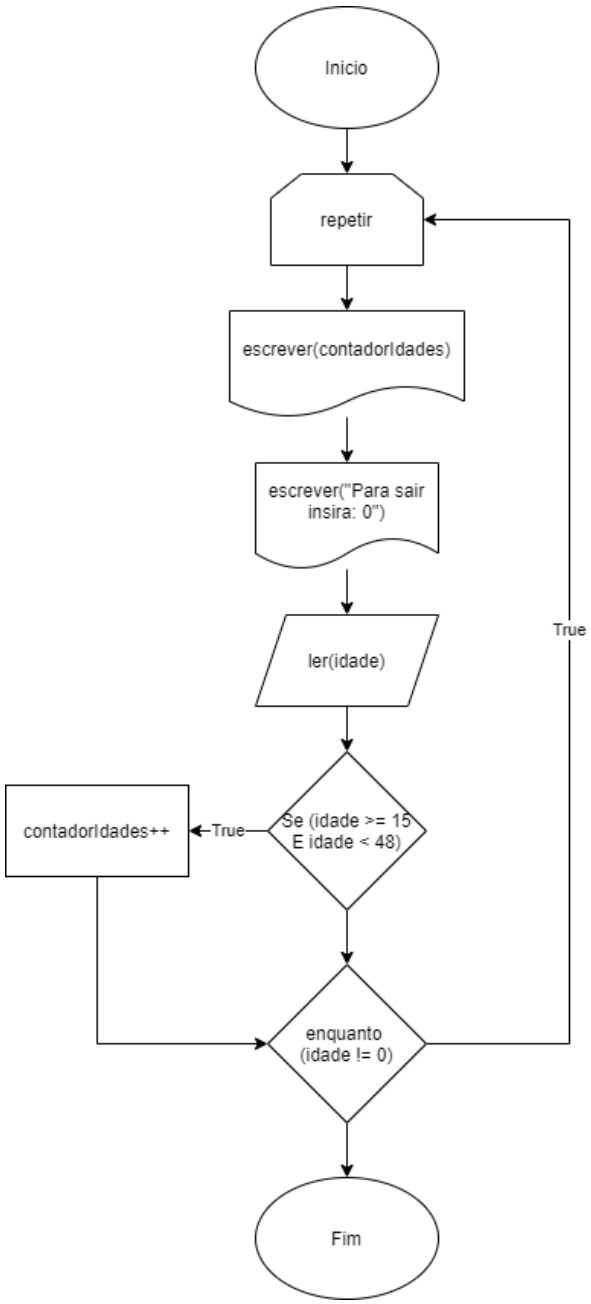


Figura 15 - Fluxograma Questão04

6.3.15. Pseudocódigo

```
Algoritmo "Questao04"

    // Função: Receber um numero indefinido de idades e mostrar na
    consola a quantidade de pessoas com idade entre 15 e 48
    // Autor: André Cerqueira
    // Data: 16/11/2020
    // Versão: 1.0

Variaveis
    contadorIdades, idade : inteiro

Inicio
    // Repetir até o utilizador inserir um 0
    repetir

        limparEcra()

        // Mostrar quantidade de idades inseridas entre 15 e 47
        escrever(contadorIdades)

        // Mostrar como sair do programa
        escrever("Para sair insira: 0")

        // Inserir idade
        ler(idade)

        // Incrementar o contador de idades
        Se (idade >= 15 E idade < 48) então
            contadorIdades++;
        fimse

    enquanto (idade != 0)
Fim
```

Figura 16 - Pseudocódigo Questão04

6.3.16. Testes

```
Total de idades inseridas entre 15 e 47 anos: [6]                [Para sair insira: 0]

Insira uma idade: 0

--------------------------------
```

Figura 17 - Testes Questão04

3.3. Questão 05

6.3.17. Descrição e abordagem do problema

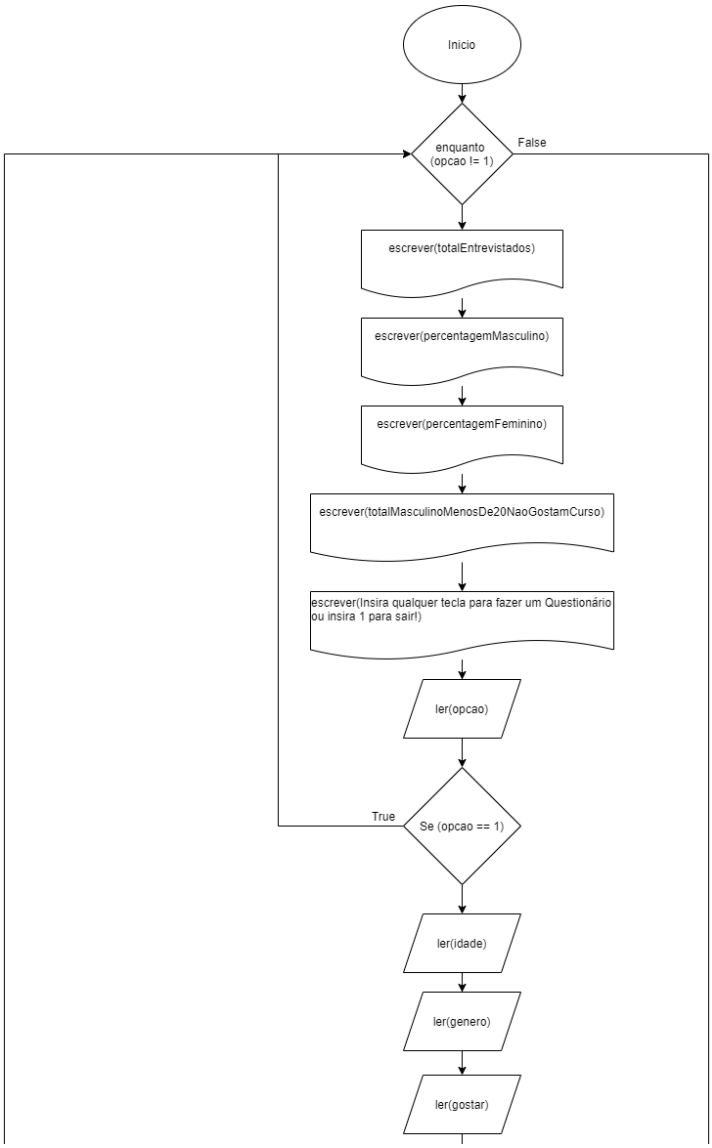
Efetuar questionários a um número indeterminado de estudantes numa universidade. A todos os estudantes é solicitado o género, a idade e se está a gostar ou não do curso que está a frequentar. A aplicação deve ser capaz de calcular e informar:

a) O número de estudantes entrevistados;

b) Percentagem de estudantes por género;

c) Quantidade de estudantes de género masculino com menos de 20 anos que não gostam do curso que estão a frequentar.

6.3.18. Fluxograma



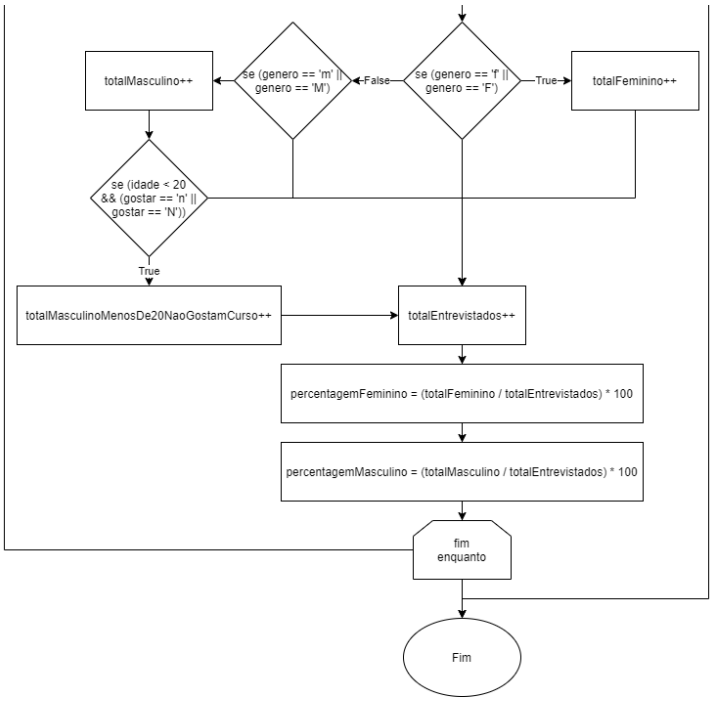


Figura 18 - Fluxograma Questão05

6.3.19. Pseudocódigo

```
Algoritmo "Questao05"

    // Função: Apresentar:
    //  O número de estudantes entrevistados;
    //  Percentagem de estudantes por género;
    //  Quantidade de estudantes de género masculino com menos de 20 anos que não gostam do curso que estão a frequentar.

    // Autor: André Cerqueira
    // Data: 16/11/2020
    // Versão: 1.0

Variaveis

    opcao, idade, totalEntrevistados, totalMasculino, totalFeminino, totalMasculinoMenosDe20NaoGostamCurso : inteiro
    percentagemFeminino, percentagemMasculino : real
    genero, gostar : caracter

Inicio

    // definir valores iniciais
    opcao = 0
    totalEntrevistados = 0
    totalMasculino = 0
    totalFeminino = 0
    totalMasculinoMenosDe20NaoGostamCurso = 0
    percentagemFeminino = 0
    percentagemMasculino = 0

    // Repetir enquanto o utilizador nao quiser sair
    enquanto (opcao != 1)

        limparEcra()

        // Apresentar resultados no ecra
        escrever(totalEntrevistados)
        escrever(percentagemMasculino)
        escrever(percentagemFeminino)
        escrever(totalMasculinoMenosDe20NaoGostamCurso)


        // Verificar se é para fazer um questionario ou para sair
        escrever("escrever(Insira qualquer tecla para fazer um Questionário ou insira 1 para sair!)")
```

```
        ler(opcao)

        // Se o 1 foi enserido entao o ciclo acaba
        se (opcao == 1) entao
            continuar
        fimse

        limparEcra()

        // Inserir Dados

        // Inserir idade
        ler(idade)
        // Inserir genero
        ler(genero)
        // Inserir se está gostar do curso
        ler(gostar)


        // Incrementar no contador do respetivo genero
        se (genero == 'f' || genero == 'F') entao
            totalFeminino++
        senao se (genero == 'm' || genero == 'M')

            totalMasculino++

            // Incrementar quem é do género masculino com menos de 20 anos e nao gosta do curso
            se (idade < 20 && (gostar == 'n' || gostar == 'N')) entao
                totalMasculinoMenosDe20NaoGostamCurso++
            fimse

        fimse

        // Incrementar o total de entrevistados
        totalEntrevistados++

        // calcular a percentagem de femininos e masculinos
        percentagemFeminino = (totalFeminino / totalEntrevistados) * 100
        percentagemMasculino = (totalMasculino / totalEntrevistados) * 100

    fimenquanto
Fim
```

Figura 19 - Pseudocodigo Questão05

6.3.20. Testes

```
Insira a sua idade: 18
Insira o seu genero [F/M] : m

Está a gostar do curso que está a frequentar? [S -> Sim] [N -> Não] s
```

```
Insira a sua idade: 16
Insira o seu genero [F/M] : m

Está a gostar do curso que está a frequentar? [S -> Sim] [N -> Não] n
```

```
Insira a sua idade: 15
Insira o seu genero [F/M] : f

Está a gostar do curso que está a frequentar? [S -> Sim] [N -> Não] n
```

```
Número de estudantes entrevistados: [3]

Género Masculino: 66,67%   |   Género Feminino: 33,33%

Quantidade de estudantes de género masculino com menos de 20 anos
que não gostam do curso que estão a frequentar: [1]


Insira qualquer tecla para fazer um Questionário! Insira [1] para sair!
-> _
```

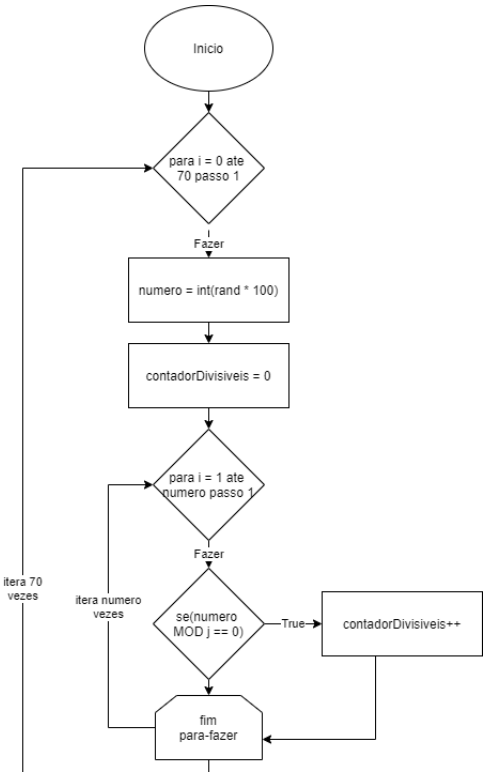
Figura 20, 21, 22 e 23 - Testes Questão05

3.1. Questão 06

6.3.21. Descrição e abordagem do problema

Gerar de forma automática e aleatória 70 números inteiros positivos entre 0 e 100. Apresentar a soma e média dos números primos existentes no conjunto criado.

6.3.22. Fluxograma



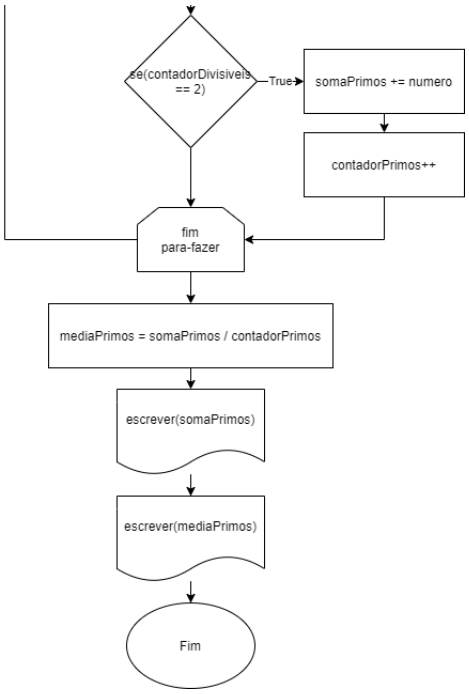


Figura 24 - Fluxograma Questão06

6.3.23. Pseudocódigo

```
Algoritmo "Questao06"
    // Função: Receber 70 numeros aleatorios entre 0 e 100 e fazer a soma e a media dos primos
    // Autor: André Cerqueira
    // Data: 15/11/2020
    // Versão: 1.0

Variaveis
    numero, i, j, contadorDivisiveis, contadorPrimos, somaPrimos : inteiro
    mediaPrimos : real

Inicio

    // definir valores iniciais
    contadorDivisiveis = 0
    contadorPrimos = 0
    somaPrimos = 0

    printf("Números Primos: ");
    para i = 0 ate 70 passo 1 fazer

        // Inicializar o numero randomicamente, e resetar o contador de Divisiveis
        numero = int(rand * 100)
        contadorDivisiveis = 0

        // Contar o total de divisiveis do numero
        para j = 1 ate numero passo 1 fazer

            se (numero MOD j == 0) entao
                contadorDivisiveis++
            fimse

        fimpara
```

```
        // Verificar se é primo
        se (contadorDivisiveis == 2) entao
            somaPrimos += numero
            contadorPrimos++
        fimse

    fimpara

    // Calcular a média dos numeros primos
    mediaPrimos = somaPrimos / contadorPrimos

    // Mostrar Resultados
    escrever(somaPrimos)
    escrever(mediaPrimos)

Fim
```

Figura 25 - Pseudocódigo Questão06

3.1.1. Traçagens e testes

Tentativa	Soma de Primos	Média de Primos
1	750	53,57
2	809	42,58
3	1180	59
4	1436	49,52
5	713	47,53
6	806	53,73
7	674	44,93
8	592	31,16
9	718	47,87
10	394	28,14

Figura 26 - Traçagem Questão06

```
Números Primos: 5, 59, 3, 61, 29, 7, 37, 3, 97, 13, 19, 47, 2, 43, 5, 79, 11, 53, 19

A soma dos números Primos é: 592
A média dos números Primos é: 31,16

--------------------------------
```

Figura 27 - Teste Questão06 - Tentativa 8

4. Funções e Procedimentos

4.1. Questão 07

6.3.24. Descrição e abordagem do problema

Reimplementar a questão 3 para que o somatório seja efetuado dentro de uma sub-rotina.

6.3.25. Pseudocódigo

```
Algoritmo "Questao07"

    // Função: Mostrar ao utilizador o resultado da soma dos numeros entre 1 e o numero inserido
    // Autor: André Cerqueira
    // Data: 28/11/2020
    // Versão: 1.0

Variaveis

    numero, i: inteiro

Inicio

    // Inserir dados
    ler(numero)

    // Apresentar resultado da soma
    escrever(Soma(numero))

Fim


// Função responsável por fazer o somatório
funcao Soma(numero: inteiro): inteiro

Variaveis

    i, soma = 0: inteiro

Inicio

    // Ciclo para incrementar no somatório
    para i = 1 ate numero passo 1 fazer
        soma += i

    retornar soma

Fim-funcao
```

Figura 28 - Pseudocódigo Questão07

4.1.1. Testes

```
Insira um número natural: 7
1 + 2 + 3 + 4 + 5 + 6 + 7 = 28
--------------------------------
```

Figura 29 - Teste Questão07

4.2. Questão 08

6.3.26. Descrição e abordagem do problema

Reimplemente a sub-rotina anterior de forma recursiva.

6.3.27. Pseudocódigo

```
// Função responsável por fazer o somatório
funcao Soma(numero: inteiro): inteiro

Inicio

    // Condição de paragem
    se (numero == 1) entao
        retornar numero
    fim-se

    // Realizar o somatório enquanto a condição de paragem não for ativada
    retornar numero + Soma(numero - 1)

Fim-funcao
```

Figura 29 - Pseudocódigo Questão08

4.2.1. Testes

```
Insira um número natural: 8
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 = 36
--------------------------------
```

Figura 30 - Teste Questão08

4.3. Questão 09

6.3.28. Descrição e abordagem do problema

Reimplementar a questão 6 para que o teste de verificação de primo seja feito dentro de uma função $Primo(x)$.

6.3.29. Pseudocódigo

```
Algoritmo "Questao09"

    // Função: Receber 70 numeros aleatorios entre 0 e 100 e fazer a soma e a media dos primos
    // Autor: André Cerqueira
    // Data: 28/11/2020
    // Versão: 1.0

Variaveis

    numero, i, contadorPrimos = 0, somaPrimos = 0: inteiro
    mediaPrimos: real

Inicio

    para i = 0 ate 70 passo 1 fazer

        // Inicializar o numero randomicamente
        numero = random(0..100)


        // Verificar se é primo
        se (Primo(numero)) entao

            somaPrimos += numero;
            contadorPrimos++;

        fim-se

    fim-para

    // Calcular a média dos numeros primos
    mediaPrimos = somaPrimos / contadorPrimos

    // Mostrar Resultados
    escrever(somaPrimos)
    escrever(mediaPrimos)

Fim

// Função que verifica se o numero é primo
funcao Primo(numero: inteiro): inteiro

Variaveis

    i: inteiro

Inicio
```

```
    // Contar o total de divisiveis do numero
    para i = 2 ate numero passo 1 fazer

        // Verificação de divisiveis
        se (numero % i == 0) entao
            retornar 0
        fim-se

    fim-para

    // Retornar o valor lógico correspondente a ser primo (1 - verdade) (0 - falso)
    se (numero < 2) entao
        retornar 0
    senao
        retornar 1
    fim-se

Fim-funcao
```

Figura 31 - Pseudocódigo Questão09

4.3.1. Testes

```
Números Primos: 7, 2, 41, 5, 19, 31, 3, 7, 43, 13, 7, 7, 97, 13, 61, 71, 73, 23, 53

A soma dos números Primos é: 576
A média dos números Primos é: 30,32

--------------------------------
```

Figura 32 - Teste Questão09

4.4. Questão 10

6.3.30. Descrição e abordagem do problema

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

6.3.31. Pseudocódigo

```
Algoritmo "Questao10"

    // Função: Ver a quantidade de numeros primos comprendidos entre 2 numeros recebidos pelo utilizador
    // Autor: André Cerqueira
    // Data: 28/11/2020
    // Versão: 1.0

Variaveis

    limiteInferior, limiteSuperior: inteiro

Inicio
```

```
    // inserir dados
    ler(limiteInferior)
    ler(limiteSuperior)

    // Apresentar Resultados
    escrever(Primos(limiteInferior, limiteSuperior)

Fim

// Função que verifica se o numero é primo
funcao Primos(limiteInferior: inteiro, limiteSuperior: inteiro): inteiro

Variaveis

    i, j, contadorPrimos, contadorDivisiveis: inteiro

Inicio

    // Ciclo onde verifica os numeros primos comprendidos entre os dois valores
    para i = limiteInferior ate limiteSuperior passo 1 fazer

        // Resetar contador de divisiveis
        contadorDivisiveis = 0


        // Contar o total de divisiveis do numero
        para j = 1 ate i passo 1 fazer

            // Verificação de divisiveis
            se (i % j == 0) entao
                contadorDivisiveis++
            fim-se

        fim-para

        // Contar os primos
        se (contadorDivisiveis == 2) entao
            contadorPrimos++
        fim-se

    fim-para

    retornar contadorPrimos

Fim-funcao
```

Figura 33 - Testes Questão10

4.4.1. Testes

```
Insira o Limite Inferior: 10
Insira o Limite Superior: 20
O total de Números Primos entre [10] e [20] é: 4
--------------------------------
```

```
Insira o Limite Inferior: 4
Insira o Limite Superior: 100
O total de Números Primos entre [4] e [100] é: 23
--------------------------------
```

Figura 34 - Testes Questão10

5. Arrays

5.1. Questão 11

6.3.32. 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: i) a escala de conversão pretendida; e ii) a lista das temperaturas. As entradas 38, 'K', significam que o utilizador pretende converter a temperatura 38 Celsius para Kelvin.

a) O utilizador deverá poder introduzir um numero indeterminado de temperaturas;
b) Deverá ser apresentada um quadro com todas as temperaturas inseridas e a respetiva conversão, indicando qual a escala de cada coluna apresentada;
c) Deverá ainda informar o utilizador acerca da média das temperaturas, assim como a maior e a menor delas, na escala original e de conversão;

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

```
/*
    Este Procedimento é usado para receber dados e retorna-los por apontador
    @escala = Escala de Temperatura pretendida
    @kelvin = Array com todas as temperaturas em kelvin
    @celcius = Array com todas as temperaturas em celcius
    @i = indice da temperatura nos arrays
*/
void ReceberDados(char *escala, float *kelvin, float *celcius, int i)
```

Figura 35 - Questão11 – Procedimento ReceberDados

```c
/*
    Este Procedimento é responsavel por apresentar a tabela de temperaturas
    @kelvin = Array com todas as temperaturas em kelvin
    @celcius = Array com todas as temperaturas em celcius
    @n = quantidade de temperaturas
*/
void MostrarTabela(float *kelvin, float *celcius, int n)
```

Figura 36 - Questão11 – Procedimento MostrarTabela

```c
/*
    Esta função é responsavel por calcular a media de um conjunto de numeros reais
    @numeros = Array de numeros reais
    @n = quantidade de numeros
*/
float Media(float *numeros, int n)
```

Figura 37 - Questão11 – Função Media

```c
/*
    Esta função é responsavel por retornar o maior numero de um conjunto de numeros reais
    @numeros = Array de numeros reais
    @n = quantidade de numeros
*/
float Maior(float *numeros, int n)
```

Figura 38 - Questão11 – Função Maior

```c
/*
    Esta função é responsavel por retornar o menor numero de um conjunto de numeros reais
    @numeros = Array de numeros reais
    @n = quantidade de numeros
*/
float Menor(float *numeros, int n)
```

Figura 39 - Questão11 – Função Menor

6.3.33. Testes

```
Celcius         Kelvin
2,00            275,00
-173,00         100,00
-73,00          200,00
150,00          423,00

A média das temperaturas é: [Celcius -> -23,50] | [Kelvin -> 249,50]
A maior temperatura é: [Celcius -> 150,00] | [Kelvin -> 423,00]
A menor temperatura é: [Celcius -> -173,00] | [Kelvin -> 100,00]

Insira [X] para sair.
Insira a Escala de Temperatura Pretendida [K - Kelvin / C - Celcius] : x

--------------------------------
```

Figura 40 – Testes Questão11

6. Estruturas

6.1. Descrição e abordagem do problema

Desenvolver um programa para gerir o pessoal de unidades clinicas.

Cada clínica tem a sua equipa médica, enfermagem e pessoal auxiliar. Cada funcionário deve possuir nome, género, idade e vencimento.

Os médicos e enfermeiros trabalham com agenda. Deve ser registado na agenda do profissional em questão as consultas. Indicando o respetivo Utente (nome e número do Serviço Nacional de Saúde (SNS)).

O programa alem de operações típicas relacionadas com edição de dados, também deve possuir:

a) Apresentar um resumo que mostre a quantidade e média de idades de médicos, enfermeiros e auxiliares, por clínica, assim como a soma de todos os vencimentos, por género, de cada um dos grupos de funcionários.
b) Listar os médicos e respetivos vencimentos agrupando-os por clínica. Esta listagem deve apresentar no final o somatório de vencimentos dos médicos listados;
c) Apresentar um resumo que apresente o número de compromissos de agenda de cada enfermeiro de uma clínica escolhida pelo utilizador;
d) Apresentar os compromissos de agenda do médico ou enfermeiro selecionado pelo utilizador, indicando qual o Utente associado a cada um dos compromissos.

6.2. Diagrama de Pacotes

De modo a resolver o problema proposto, facilitar o desenvolvimento e entendimento do mesmo. Foi elaborado um Diagrama de Pacotes onde o programa é divido em três pacotes.

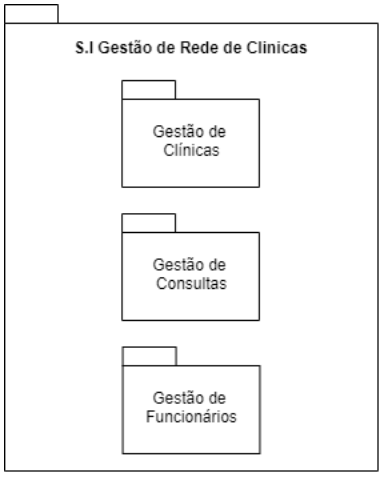


Figura 41 – Diagrama de Pacotes Questão12

6.1. Diagrama de Classes

De modo a entender quais estruturas o sistema deve possuir e como as mesmas interagem entre si. Foi elaborado um Diagrama de Classes.

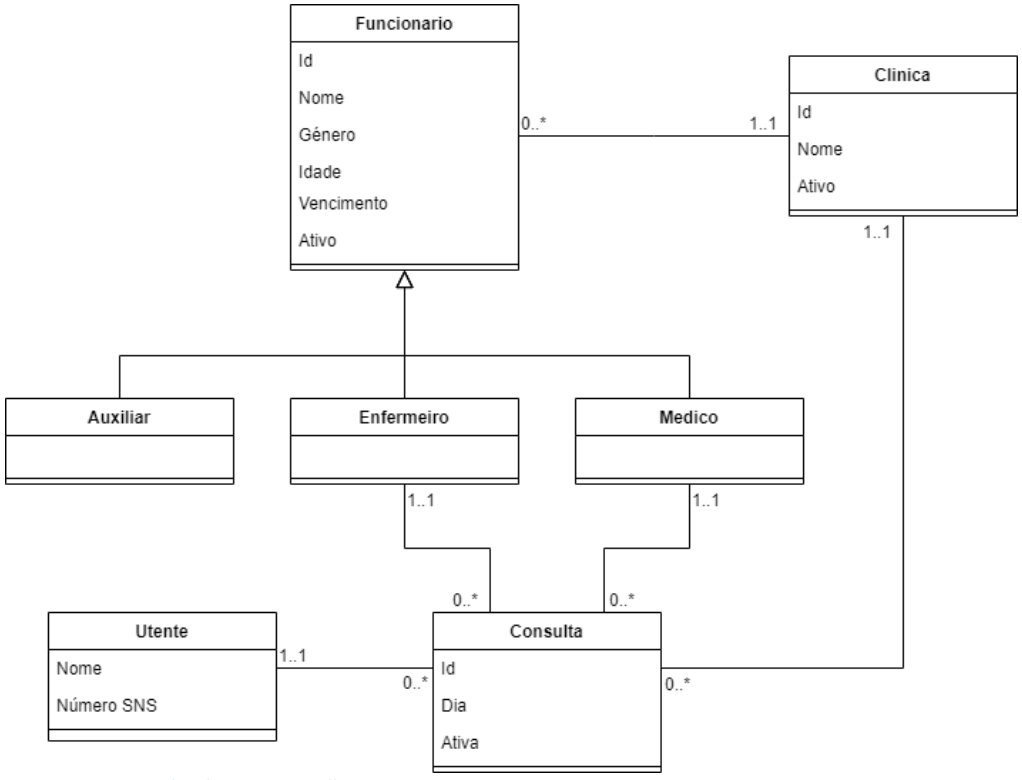


Figura 42 – Diagrama de Classes Questão12

6.2. Estruturas desenvolvidas (explicadas)

6.3.34. Clínica

Esta estrutura representa uma Clinica. Possui um ID pelo qual é identificada. Um campo ativo para ser possível saber se a clinica foi apagada do sistema. A Clinica também possui um campo com a lista de funcionários. E um campo com a lista de Consultas, bem como o número total das mesmas.

```
typedef struct clinica
{
    int id;
    char nome[50];
    Funcionario funcionarios[20];
    int totalFuncionarios;
    Consulta consultas[20];
    int totalConsultas;
    char ativo;
} Clinica;
```

Figura 43 – Estrutura Clinica Questão12

6.3.35. Funcionário

Esta estrutura representa um Funcionário. Possui um ID pelo qual é identificada. Um campo ativo para ser possível saber se o funcionário está ativo no sistema. Além dos outros campos do funcionário, salienta-se o facto do campo emprego apenas poder ser Médico, Enfermeiro ou Auxiliar.

```
typedef struct funcionario
{
    int id;
    char nome[50];
    char genero[10];
    unsigned int idade;
    float vencimento;
    char emprego[15];
    char ativo;
} Funcionario;
```

Figura 44 – Estrutura Funcionário Questão12

6.3.36. Consulta

Esta estrutura representa uma Consulta. Possui um ID pelo qual é identificada. Um campo ativo para ser possível saber se a consulta foi apagada do sistema. A Consulta também possui um campo com a data, o funcionário e o utente da mesma.

```
typedef struct consulta
{
    int id;
    char data[15];
    Funcionario funcionario;
    Utente utente;
    char ativo;
} Consulta;
```

Figura 45 – Estrutura Consulta Questão12

6.3.37. Utente

Esta estrutura representa um Utente. Apenas possui o nome e o número SNS do mesmo.

```
typedef struct utente
{
    char nome[50];
    char numeroSns[10];
} Utente;
```

Figura 46 – Estrutura Utente Questão12

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

6.3.1. Main

Este é o script principal do programa, onde todos os módulos se juntam.

```
/*
    Este procedimento apresenta o Menu Inicial e retorna através de um ponteiro uma opção selecionada no menu
    @op = opção selecionada no menu
*/
void MenuInicial(int *op)
```

Figura 47 – Procedimento Menu Inicial Questão12

```
/*
    Procedimento extra para carregar dados por Hardcode para simular dados e testar o programa
    @clinicas = lista de clinicas
    @n = quantidade total de clinicas
*/
void CarregarDadosHardCode(Clinica *clinicas, int *n)
```

Figura 48 – Procedimento Carregar Dados HardCode Questão12

6.3.2. Módulo Gestão de Clinicas

Este módulo é composto por todas as funções e procedimentos, relacionados com as clinicas e a sua gestão.

```
/*
    Este procedimento apresenta o Menu da Clinica e retorna através de um ponteiro uma opção selecionada no menu.
    @op = opção selecionada no menu
    @nome = nome da clinica selecionada
*/
void MenuClinica(int *op, char *nome)
```

Figura 49 – Procedimento Menu Clinica Questão12

```
/*
    Este procedimento é responsavel por listar todas as clinicas. E possui 2 parâmetros de entrada.
    @clinicas = lista de clinicas
    @n = quantidade total de clinicas
*/
void ListarClinicas(Clinica *clinicas, int n)
```

Figura 50 – Procedimento Listar Clinica Questão12

```
/*
    Este procedimento é responsavel pela remoção de uma clinica. E possui 2 parâmetros de entrada.
    @clinicas = lista de clinicas
    @n = quantidade total de clinicas
*/
void RemoverClinica(Clinica *clinicas, int n)
```

Figura 51 – Procedimento Remover Clinica Questão12

```
/*
    Este procedimento é responsavel pela criação de uma clinica. E possui 2 parâmetros de entrada.
    @clinicas = lista de clinicas
    @n = quantidade total de clinicas
*/
void AddClinica(Clinica *clinicas, int *n)
```

Figura 52 – Procedimento Adicionar Clinica Questão12

```
/*
    Procedimento para mostrar as informações da clinica selecionada. E possui 1 parâmetros de entrada.
    @clinica = clinica selecionada pelo utilizador
*/
void ClinicaInfo(Clinica clinica)
```

Figura 53 – Procedimento Clinica Info Questão12

```
/*
    Este procedimento é responsavel por listar todas os médicos de todas as clinicas. E possui 2 parâmetros de entrada.
    @clinicas = lista de clinicas
    @n = quantidade total de clinicas
*/
void ListarTodosMedicos(Clinica *clinicas, int n)
```

Figura 54 – Procedimento Listar todos os Médicos Questão12

6.3.3. Módulo Gestão de Funcionários

Este módulo é composto por todas as funções e procedimentos, relacionados com os funcionários e a sua gestão.

```
/*
    Este procedimento é responsavel pela criação de um Funcionario. E possui 2 parâmetros de entrada.
    @funcionarios = lista de funcionarios
    @n = quantidade total de funcionarios
*/
void AddFuncionario(Funcionario *funcionarios, int *n)
```

Figura 55 – Procedimento Adicionar Funcionário Questão12

```
/*
    Este procedimento é responsavel por listar todos os Funcionarios ativos. E possui 2 parâmetros de entrada.
    @funcionarios = lista de funcionarios
    @n = quantidade total de funcionarios
*/
void ListarFuncionarios(Funcionario *funcionarios, int n)
```

Figura 56 – Procedimento Listar Funcionários Questão12

```
/*
    Este procedimento é responsavel pela edição de um Funcionario. Só é possivel alterar a idade e o vencimento.
    E possui 2 parâmetros de entrada.
    @funcionarios = lista de funcionarios
    @n = quantidade total de funcionarios
*/
void EditarFuncionario(Funcionario *funcionarios, int n)
```

Figura 64 – Procedimento Editar Funcionário Questão12

```c
/*
    Este procedimento é responsavel por remover / inativar um Funcionario. E possui 2 parâmetros de entrada.
    @funcionarios = lista de funcionarios
    @n = quantidade total de funcionarios
*/
int RemoverFuncionario(Funcionario *funcionarios, int n)
```

Figura 57 – Função Remover Funcionário Questão12

```c
/*
    Função para retornar a quantidade de funcionarios com um determinado emprego dentro de uma clinica. E possui 3 parâmetros de entrada.
    @funcionarios = lista de funcionarios
    @n = quantidade total de funcionarios
    @emprego = emprego o qual prentende a retornar a quantidade de funcionarios
*/
int QuantidadeFuncionariosPorEmprego(Funcionario *funcionarios, int n, char *emprego)
```

Figura 58 – Função Quantidade Funcionários por Emprego Questão12

```c
/*
    Função para retornar a média de idades dos funcionarios de um determinado emprego dentro de uma clinica. E possui 3 parâmetros de entrada.
    @funcionarios = lista de funcionarios
    @n = quantidade total de funcionarios
    @emprego = emprego o qual prentende a retornar a media de idades dos funcionarios
*/
float MediaIdadesFuncionariosPorEmprego(Funcionario *funcionarios, int n, char *emprego)
```

Figura 59 – Função Media Idades Funcionários por Emprego Questão12

```c
/*
    Função para retornar a soma de vencimentos dos funcionarios de um determinado emprego e genero dentro de uma clinica. E possui 4 parâmetros de entrada.
    @funcionarios = lista de funcionarios
    @n = quantidade total de funcionarios
    @emprego = emprego pretendido
    @genero = genero pretendido
*/
float SomaVencimentosPorGeneroEmprego(Funcionario *funcionarios, int n, char *emprego, char *genero)
```

Figura 60 – Função Soma Vencimentos por Género e Emprego Questão12

6.3.4. Módulo Gestão de Consultas

Este módulo é composto por todas as funções e procedimentos, relacionados com as consultas e a sua gestão.

```c
/*
    Este procedimento é responsavel por listar todas as consultas. E possui 2 parâmetros de entrada.
    @consultas = lista de consultas
    @n = quantidade total de consultas
*/
void ListarConsultas(Consulta *consultas, int n)
```

Figura 61 – Procedimento Listar Consultas Questão12

```c
/*
    Este procedimento é responsavel por listar todas as consultas de um determinado funcionario. E possui 4 parâmetros de entrada.
    @consultas = lista de consultas
    @nC = quantidade total de consultas
    @funcionarios = lista de funcionarios
    @nF = quantidade total de funcionarios
*/
void Agenda(Consulta *consultas, int nC, Funcionario *funcionarios, int nF)
```

Figura 62 – Procedimento Agenda Questão12

```c
/*
    Este procedimento é responsavel pela marcação de uma consulta. E possui 4 parâmetros de entrada.
    @consultas = lista de consultas
    @nC = quantidade total de consultas
    @funcionarios = lista de funcionarios
    @nF = quantidade total de funcionarios
*/
void AddConsulta(Consulta *consultas, int *nC, Funcionario *funcionarios, int nF)
```

Figura 63 – Procedimento Adicionar Consulta Questão12

```c
/*
    Este procedimento é responsavel por remover / inativar uma consulta. E possui 2 parâmetros de entrada.
    @consultas = lista de consultas
    @n = quantidade total de consultas
*/
void RemoverConsulta(Consulta *consultas, int n)
```

Figura 65 – Procedimento Remover Consulta Questão12

```c
/*
    Este procedimento é responsavel pela remarcação de uma consulta. E possui 2 parâmetros de entrada.
    @consultas = lista de consultas
    @n = quantidade total de consultas
*/
void RemarcarConsulta(Consulta *consultas, int n)
```

Figura 66 – Procedimento Remarcar Consulta Questão12

```c
/*
    Este procedimento é responsavel por inativar todas as consultas de um Funcionario. E possui 3 parâmetros de entrada.
    @consultas = lista de consultas
    @n = quantidade total de consultas
    @id = id do funcionario selecionado
*/
void InativarTodasConsultasPorFuncionario(Consulta *consultas, int n, int id)
```

Figura 67 – Procedimento Inativar Todas as Consultas Por Funcionário Questão12

```c
/*
    Este procedimento é responsavel por verificar se o funcionario selecionado pode gerir consultas. E possui 2 parâmetros de entrada.
    @funcionarios = lista de funcionarios
    @id = id selecionado
*/
int VerificarFuncionario(Funcionario funcionario)
```

Figura 68 – Função Verificar Funcionário Questão12

```c
/*
    Esta função é responsavel por retornar o total de compromissos/consultas de um funcionário. E possui 3 parâmetros de entrada.
    @funcionario = funcionario pretendido
    @consultas = lista de consultas
    @n = quantidade total de consultas
*/
int TotalConsultasPorFuncionario(Funcionario funcionario, Consulta *consultas, int n)
```

Figura 69 – Função Total Consultas Por Funcionário Questão12

6.3.5. Utilis

Este módulo é composto por todas as funções e procedimentos extras usados várias vezes por todos os módulos.

```c
/*
    Esta funçao é responsavel por retornar um valor (1 ou 2). Representando Confirmar ou Cancelar uma operação.
*/
int Confirmar()
```

Figura 70 – Função Confirmar Questão12

```c
/*
    Esta função é responsavel por retornar um indice de uma determinada lista, ou um número negativo se pertender Cancelar.
    @n = quantidade total da lista
*/
int SelecionarPorID(int n)
```

Figura 71 – Função Selecionar por ID Questão12

```c
/*
    Este procedimento é responsavel por retornar uma data por apontador. Tendo que inserir o Dia, Mês e Ano.
*/
void GetDate(char *data)
```

Figura 72 – Procedimento GetDate Questão12

6.4. Testes

Lista de Consultas				
ID	Data	Funcionario	Nome Utente	SNS Utente
0	24/02/2021	Antonio	Jorge	165093288
1	12/03/2021	Anibal	Tobias	125343223
2	01/03/2021	Antonio	Antonio	132420045
3	12/03/2021	Ana	Miguel	165093284
4	24/02/2021	Ana	Pedro	125343223
5	11/03/2021	Antonio	Marco	132420045

Figura 73 – Teste Consultas Questão12

Agenda de Antonio			
ID	Data	Nome Utente	SNS Utente
0	24/02/2021	Jorge	165093288
2	01/03/2021	Antonio	132420045
5	11/03/2021	Marco	132420045

Figura 74 – Teste Agenda Questão12

7. Conclusão

No final destes trabalhos individuais é possível concluir que os conhecimentos adquiridos ao longo do semestre proporcionaram uma clara melhoria em programação e acima de tudo em desenvolvimento de software. Foi uma boa experiência e espero continuar a melhorar.