

**Relatório**

**Linguagem de programação 1**

**Alunos:   
Roberto Filipe Manso Barreto  
Vincent Rebena**

**Henrique Cartucho**

**Professor: João Carlos Cardoso da Silva**

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, janeiro, 2020

Resumo

Este documento refere-se então à descrição do trabalho a realizado pelos alunos Roberto Filipe Manso Barreto, Vincent Rebena e Henrique Cartucho.

Este trabalho proposto pelo professor da disciplina visa a utilização dos conhecimentos de programação lecionados em aula para a realização do mesmo, com a finalidade de desenvolver um programa capaz de ler ficheiros e apresentar as suas informações tratadas referentes a uma prova de rally.

**Palavras-Chave:**

Lista de Abreviaturas e Siglas

Índice de Figuras

Figura 1 - fluxograma 1 9

Índice

[1 Estrutura do Documento 1](#_Toc59888746)

[2 Desenvolvimento do trabalho 2](#_Toc59888747)

[2.1 Esquematização 2](#_Toc59888748)

[2.2 GitHub 2](#_Toc59888749)

[2.3 Primeira fase do trabalho (leitura dos ficheiros) 3](#_Toc59888750)

[2.3.1 Leitura do ficheiro contendo os tempos da prova 3](#_Toc59888751)

[2.3.2 Leitura do ficheiro contendo as distâncias da prova 6](#_Toc59888752)

[2.3.3 Leitura do ficheiro contendo as informações dos Pilotos 8](#_Toc59888753)

# Estrutura do Documento

Este documento é composto por uma descrição do trabalho realizado, mencionando problemas encontrados e soluções para os mesmos, descrito também com pequenas porções de código.

# Desenvolvimento do trabalho

## Esquematização

Primeiramente para obter uma boa base de trabalho foi então realizado uma esquematização do trabalho a realizar bem como a distribuição de tarefas, esta fase permitiu perceber como realizar as várias tarefas do trabalho e também sempre ter uma referência para verificar quando alguma dúvida sobre a base do trabalho aparecer.

## GitHub

Para a partilha das várias tarefas do grupo foi criado um repositório no github, neste repositório foram criados vários *branches* para assim cada membro do grupo conseguir trabalhar na sua tarefa sem problemas de colocar em risco o ficheiro principal e por fim simplesmente juntar ao ramo principal quando for verificado que nenhum problema vai resultar de juntar porções de código, ao código principal. Esta ferramenta permitiu também a exploração de várias formas de realizar as diferentes tarefas pois é possível simplesmente criar mais ramos para fazer testes e assim aprender mais formas de realizar e resolver certos problemas q podem vir a aparecer de novo no futuro.

## Primeira fase do trabalho (leitura dos ficheiros)

Nesta fase do trabalho o objetivo era a leitura dos diferentes ficheiros e a colocação dos valores dos ficheiros em variáveis, para isso foram criadas vários *structs* para guardar assim cada *struct* os respetivos valores do ficheiro e assim com mais facilidade manipular e tratar os dados contidos nestes ficheiros.

### Leitura do ficheiro contendo os tempos da prova

Nesta fase do trabalho o objetivo era ler um ficheiro contendo informações de número do piloto, etapa inicial, etapa final e tempo que o piloto demorou entre a etapa inicial e a etapa final, este ficheiro tinha ainda no seu começo anotado o número de etapas e de pilotos.

Exemplo do ficheiro:

3;2

2;E1;E2;21672

1;P;E1;10501

1;E1;E2;37203

2;P;E1;12383

1;E2;C;28465

2;E2;C;23567

Primeiramente foi criado um *struct* para guardar os valores lidos neste ficheiro, para isso foi criado o *struct* *infoTempos* que foi definido como *TEMPOS* que tinha dentro do mesmo dois inteiros, o numero do piloto e o tempo que demorou a ir da etapa inicial até a etapa final, e dois vetores de caracteres para guardar a etapa inicial e a etapa final, e primeiramente foi colocado como tamanho do vetor 1 valor, pois pensado em *“E1”* tem dois caracteres *‘E’* e *‘1’* ou seja 0, 1, mas foi depois entendido que era necessário um vetor de tamanho 3 para guardar esses valores pois senão iriam acontecer erros ao colocar valores dentro destes mesmo vetores.

typedef struct infotempos

{

    int num, tempo;

    char etapaI[3]; *//Etapa inicial*

    char etapaF[3]; *//Etapa Final*

} TEMPOS;

Foi então depois criada uma função que obtivesse apenas os primeiros dois números, para isso bastava apenas ler a primeira linha e guardar os resultados num vetor que guardava na posição 0 a quantidade de etapas e na posição 1 a quantidade de pilotos, para realizar isto foi feito um apontador para o vetor *n[1],* realizado a abertura do ficheiro no modo de leitura e de seguida a leitura da primeira linha e atribuição dos valores lidos ao vetor n através da função *fscanf*.

Estes valores eram também importantes para obter o tamanho do vetor do tipo *TEMPOS* pois este vetor guardaria todas as informações sobre os tempos dos pilotos neste ficheiro logo a quantidade de informações a ler seria a quantidade de etapas a multiplicar pela quantidade de pilotos.

void Etapas(int \**n*)

{

    FILE \*f;

    f = fopen("tempos.txt", "r");

    fscanf(f, "%d;%d\n", &*n*[0], &*n*[1]);

    fclose(f);

}

Obtido então a quantidade de informações que o vetor tempos do tipo *TEMPOS* teria foi este inicializado com o tamanho *nTotal* que era a variável que continha a multiplicação do número total de etapas pelo número de pilotos contidos na primeira linha do ficheiro, números estes que estavam igualados às variáveis *nEtapas* e *nPilotos*.

int n[1], nTotal, nEtapas, nPilotos;

    Etapas(n);

    nEtapas = n[0];

    nPilotos = n[1];

    nTotal = nEtapas \* nPilotos;

    TEMPOS tempos[nTotal];

De seguida foi então criada a função *loadTempos*, esta função tinha um apontador para o vetor acima mostrado, abria o ficheiro *“tempos.txt”* e utilizando a função *fseek* foi avançado a primeira linha deste ficheiro, pois a primeira linha apenas continha o número de etapas e de pilotos. Para conseguir ler o ficheiro até ao seu término foi criado a variável *res* que continha o resultado devolvido pela função *fscanf*, esta função quando encontra o fim do ficheiro devolve *“EOF”,* *end of file*, logo aproveitando isso foi criado um ciclo *while* que tem como condição *res* ser diferente de “EOF”, e dentro deste ciclo foi colocado que a variável *res* que recebe o valor da função *fscanf* ao ficheiro *“tempos.txt”.* Ao igualar esta função à variável, esta é também executada e aproveitando isto foi então já guardado no vetor os valores retirados do ficheiro pela função *fscanf*, para ser possível retirar apenas as strings até *‘;’* foi então utilizado *“%[^;]”* querendo esta expressão dizer para ler todos os caracteres até encontrar o caracter *‘;’*, pois se fosse utilizado *“%s”* iria ser lido o resto da linha toda como uma *string* e seria necessário separar em duas *strings* levando a mais tempo de processamento desnecessário. O vetor recebe então os valores na posição i, este i é uma variável inicializada a 0 que a cada ciclo realizado no *while* é incrementado por 1 valor conseguindo assim percorrer o vetor por completo e guardar os respetivos valores em cada posição.

void loadTempos(TEMPOS \**etapa*)

{

    FILE \*f;

    int res, i = 0;

    f = fopen("tempos.txt", "r");

    fseek(f, sizeof(int), SEEK\_SET);

*//enquanto tiver o que ler, vai ler*

    while (res != EOF)

    {

        res = fscanf(f, "%d;%[^;];%[^;];%d\n", &*etapa*[i].num,

*etapa*[i].etapaI, *etapa*[i].etapaF, &*etapa*[i].tempo);

        i++;

    }

    fclose(f);

}

### Leitura do ficheiro contendo as distâncias da prova

Para esta fase do trabalho o objetivo era necessário a leitura de um ficheiro que continha as distâncias entre as várias etapas, ou seja, este ficheiro teria em cada linha a etapa inicial, a etapa final e a distância entre as etapas. Exemplo:

E1;E2;32.230

P;E1;23.100

E2;C;25.720

Para receber os dados deste ficheiro foi então criado um *struct* que foi definido como *DISTANCIAS* com as variáveis *etapaI* e *etapaF* que são dois vetores que recebem o nome da etapa, estes vetores tem tamanho 3, o *struct* tem também uma variável do tipo *float* para guardar as distâncias entre as etapas.

typedef struct infoDistancias

{

    char etapaI[3]; *//Etapa inicial*

    char etapaF[3]; *//Etapa Final*

    float distancia;

} DISTANCIAS;

Depois para guardar os valores, foi criado um vetor do tipo *DISTANCIAS* para guardar as informações, para iniciar este vetor foi então aproveitado a contagem de etapas recebido anteriormente, pois o máximo de distancias a obter é a quantidade de etapas.

nEtapas = n[0];

DISTANCIAS distancias[nEtapas];

    loadDistancias(distancias);

De seguida para ler o ficheiro e guardar as variáveis foi um processo igual ao ficheiro anterior, mas agora sem a necessidade da utilização da função *fseek*, pois não existia dados “amais” no começo do código, o que altera agora é o nome do ficheiro a abrir que agora será *“distancias.txt”* e também o vetor que está a apontar, que é o vetor distâncias agora e não o vetor tempos.

Para ler as duas etapas foi utilizado *“[^;]”* para ler todos os caracteres até encontrar *‘;’* conseguindo assim obter a etapa inicial e final, já para ler a distância foi utilizado *“%f”* pois estamos a ler uma variável do tipo *float*.

void loadDistancias(DISTANCIAS \**distancias*)

{

    FILE \*f;

    int res, i = 0;

    f = fopen("distancias.txt", "r");

    while (res != EOF)

    {

        res = fscanf(f, "%[^;];%[^;];%f\n", *distancias*[i].etapaI,

*distancias*[i].etapaF, &*distancias*[i].distancia);

        i++;

    }

    fclose(f);

}

### Leitura do ficheiro contendo as informações dos Pilotos

Este ficheiro continha então as várias informações dos pilotos, estando este organizado por número do piloto, nome do piloto e carro do piloto, exemplo:

1;Joao;Subaru

2;Maria;Subaru

3;Joana;BMW

5;Jose;Lancia

6;Carlos;Audi

Para receber as informações dos pilotos foi então criado um *struct* definido como PILOTO que continha uma variável do tipo inteiro chamada num e dois vetores do tipo caracter para guardar o nome do piloto e do carro.

typedef struct infoPiloto

{

    int num;

    char nome[50];

    char carro[50];

} PILOTO;

Depois de criado o *struct* foi então criado um vetor do tipo *PILOTO*, partindo do princípio que o número de pilotos poderia ser diferente do número de pilotos no ficheiro *“tempos.txt”*, foi então criada uma função que serve para realizar a contagem de pilotos.

Esta função vai verificar todos os caracteres existentes num ficheiro utilizando um ciclo for que tem como condição o caracter lido ser diferente de *“EOF”,end of file*, e que a cada ciclo iguala a variável *c* ,que é a variável que guarda cada caracter lido, à função *getc* que vai atribuir o caracter seguinte do ficheiro a *c* e sempre que *c* for igual a *‘\n’* vai então incrementar por 1 o contador do numero de linhas, como cada linha equivale a informações de um piloto, então assim contando o numero de linhas é obtido o numero de pilotos existentes na prova, após tudo esta função retorna a quantidade de linhas.

int nPilotosCount()

{

    FILE \*f;

    int contador = 1;

    char c;

    f = fopen("pilotos.txt", "r");

    for (c = getc(f); c != EOF; c = getc(f))

    {

        if (c == '\n')

            contador++;

    }

    fclose(f);

    return contador;

}

Após isto já seria possível criar um vetor do tipo *PILOTO* para guardar todos os pilotos e assim foi declarado o vetor pilotos que tem como tamanho *nPilotos* que é antes igualada à função atrás mostrada para assim o valor retornado pela função ser atribuído a *nPilotos*.

nPilotos = nPilotosCount();

    PILOTO pilotos[nPilotos];

    loadPilotos(pilotos);

Após isto para colocar valores no vetor pilotos tal como para o ficheiro anterior foi criado uma função que tem um apontador para o vetor pilotos e foi alterado o ficheiro a abrir pois agora seria o ficheiro *“pilotos.txt”*, pois em todos os ficheiros a forma como se realiza a leitura é igual.

Nesta função para ler de forma correta todas as informações foi primeiramente lido o número *“%d”* pois número é inteiro, e depois as duas *strings*, o nome e carro, o nome foi lido com *“[^;]”* para assim se possível ler até chegar ao caracter *‘;’* e o carro foi lido com *“[^\n]”* para assim ler até encontrar o caracter *‘\n’* que significa avanço de linha logo seria aqui o término da nossa linha de informações do vetor. Cada informação é guardada no respetivo campo sendo eles “.num”, ”.nome” e “.carro”.

void loadPilotos(PILOTO \**piloto*)

{

    FILE \*f;

    int res, i = 0;

    f = fopen("pilotos.txt", "r");

    while (res != EOF)

    {

        res = fscanf(f, "%d;%[^;];%[^\n]\n", &*piloto*[i].num,

*piloto*[i].nome, *piloto*[i].carro);

        i++;

    }

    fclose(f);

}