

**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.

Índice de Figuras

Figura 1 - Menu principal do programa desenvolvido 7

Figura 2 - Amostra do número de pilotos na prova 7

Figura 3 - Amostra do número de concorrentes com prova válida 8

Figura 4 - Amostra dos concorrentes com prova válida de forma decrescentes 9

Figura 5 - Apresentação das médias ao utilizador 10

Figura 6 - Amostra do piloto mais rápido e mais lento 11

Figura 7 - Apresentação do menor tempo possível de realizar a prova 12

Figura 8 - Amostra das velocidades medias 13

Figura 9 - Apresentação da tabela classificativa 14

Índice

[1 Estrutura do Documento 1](#_Toc60253225)

[2 Desenvolvimento do trabalho 2](#_Toc60253226)

[2.1 Esquematização 2](#_Toc60253227)

[2.2 GitHub 2](#_Toc60253228)

[2.3 Primeira fase do trabalho (leitura dos ficheiros) 2](#_Toc60253229)

[2.3.1 Leitura do ficheiro contendo os tempos da prova 3](#_Toc60253230)

[2.3.2 Leitura do ficheiro contendo as distâncias da prova 5](#_Toc60253231)

[2.3.3 Leitura do ficheiro contendo as informações dos Pilotos 6](#_Toc60253232)

[2.4 Realização dos exercícios 7](#_Toc60253233)

[2.4.1 Apresentação do número de pilotos na prova 7](#_Toc60253234)

[2.4.2 Apresentação do número de concorrentes com prova valida 8](#_Toc60253235)

[2.4.3 Listagem ordenada todos os pilotos que efetuaram uma prova válida 8](#_Toc60253236)

[2.4.4 Cálculo de médias dos tempos por etapa 10](#_Toc60253237)

[2.4.5 Apresentação do concorrente mais rápido e mais lento 11](#_Toc60253238)

[2.4.6 Cálculo do menor tempo possível de efetuar a prova 12](#_Toc60253239)

[2.4.7 Listagem das velocidades médias de toda a prova 13](#_Toc60253240)

[2.4.8 Apresentação da tabela classificativa 14](#_Toc60253241)

# 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.

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.

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*.

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.

### 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.

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.

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*.

### 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.

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.

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*.

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”.

## Realização dos exercícios

Para a realização dos exercícios foi criado um menu para assim aceder às diversas opções do programa a ser criado.

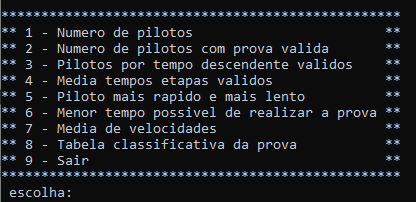
Para desenvolver este menu foram apresentados ao utilizador as várias opções que tem à sua disposição, para obter a escolha do utilizador foi utilizado a função *scanf* que guarda a opção escolhida numa variável. Após obter a escolha do utilizador é realizado um *switch*, dependendo a escolha do utilizador são realizadas apenas as funções necessárias para a amostra das informações que o utilizador pretende. Para o utilizador não ficar confuso foi também realizado uma “espera” no programa antes de o utilizador voltar ao menu após a visualização das informações

Figura 1 - Menu principal do programa desenvolvido

### Apresentação do número de pilotos na prova

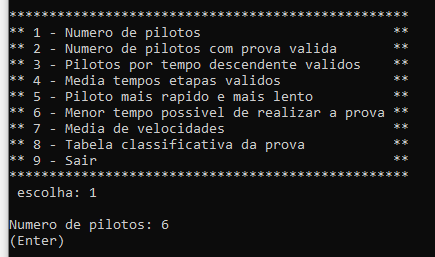
Para a apresentação do número pilotos em prova foi utilizada a função *nPilotosCount*, pois esta função retorna a contagem total de pilotos inscritos na prova, sendo assim só necessário apresentar ao utilizador o valor retornado pela função utilizando um *printf* com a função como variável a mostrar.

Figura 2 - Amostra do número de pilotos na prova

### Apresentação do número de concorrentes com prova valida

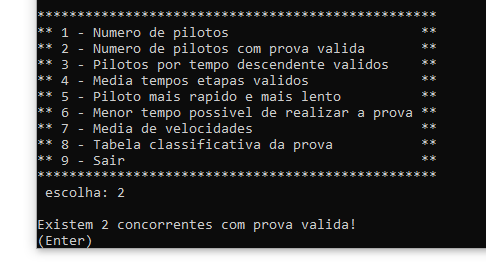
Para a apresentação do número de pilotos aprovados foi criada uma função que possibilitava a verificação das etapas chamada *verificaProva*. Esta função era composta por um ciclo for que percorria o vetor tempos por completo e a cada ciclo, era realizado mais um ciclo for que percorria o restante do vetor e vai comparando os valores lidos ao valor lido no primeiro ciclo, caso sejam iguais o contador de vezes que é registado este número é incrementado por 1, caso o valor do contador seja 3, significa que este piloto concluiu as 3 etapas e é guardado num apontador para um vetor o número deste piloto, logo este é um piloto aprovado. Após isto a função retorna o número de pilotos aprovados e este valor é colocado numa função *printf* e apresentado ao utilizador.

Figura 3 - Amostra do número de concorrentes com prova válida

### Listagem ordenada todos os pilotos que efetuaram uma prova válida

Para a apresentação ordenada de todos os pilotos que efetuaram uma prova válida foi criado um *struct* que tem como fim guardar os dados da prova por completo para assim facilitar o restante do trabalho a realizar. Este *struct* tem 3 variáveis, um do tipo PILOTO para guardar as informações do piloto e duas variáveis do tipo inteiro para guardar o tempo total de prova e se o piloto está ou não aprovado e foi então dado o nome de PROVA a este *struct*, e o mesmo foi inicializado na função main como um vetor chamado prova.

Após isto foi criada uma função chamada loadProva que guarda no vetor prova as informações dos pilotos, o tempo total de prova e se o mesmo está aprovado ou não. Para realizar isto foi então reutilizado o código da função verificaProva , o número do piloto era obtido no primeiro for, após isso sempre que se encontrava o mesmo número de piloto era incrementado ao tempo total o tempo daquele piloto, naquela posição, por fim era então copiado o carro e o nome do piloto e caso se contasse 3 vezes o piloto neste vetor o mesmo contava como aprovado e este valor era então colocado no vetor prova.

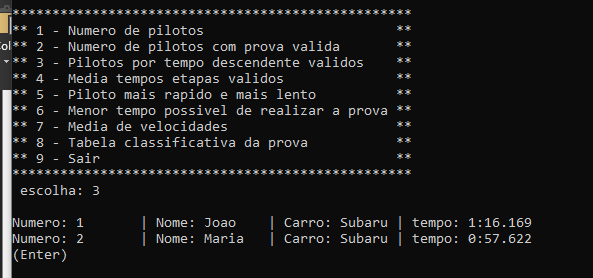
Ao fim de realizar esta função foi criada a função ordenaTemposDesc que no começo copia os pilotos aprovados do vetor prova para outro vetor do tipo PROVA para assim não alterar o vetor original. Para então ordenar este vetor foram utilizados dois ciclos for, o primeiro ciclo percorre o vetor numero e numero e o segundo for verifica os restantes números de cada ciclo do primeiro numero, utilizando esta base cada vez que o segundo ciclo encontrava um numero maior que o numero do primeiro for, os mesmos trocavam de posições no vetor fazendo uso de variáveis auxiliares para guardar as informações de um vetor visto que o primeiro a ser alterado iria perder as suas antigas informações não sendo assim possível igualar os dois de formas diferentes. Após ordenar o vetor é utilizado um ciclo for para apresentar as informações dos pilotos todos ao utilizador, para isso é realizado a conversão de milissegundos para minutos, segundos e milésimos de segundo.

Figura 4 - Amostra dos concorrentes com prova válida de forma decrescentes

### Cálculo de médias dos tempos por etapa

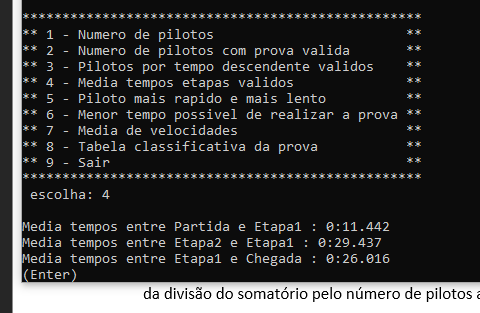
Para a apresentação das médias de tempos por etapa, primeiramente é executada a função *verificaProva* para preencher o vetor que contém o número dos pilotos válidos. De seguida é realizado um apontador para o vetor que contém os números dos pilotos aprovados, foi depois então criado uma função chamada *medTemposEtapa* que através de um ciclo for percorre o vetor tempos e outro ciclo for que percorre todo o vetor aprovados e verifica o numero de piloto que está contido vetor tempos no ciclo for anterior é igual ao numero de piloto contido no vetor aprovados no segundo ciclo for, caso seja vai verificar todas as etapas e vai então acrescentar o tempo deste pilo ao somatório das etapas. Depois disto é calculada a média que é obtida através da divisão do somatório pelo número de pilotos aprovados e por fim estas médias são guardadas num apontador para um vetor para assim se possível “retornar” as 3 medias obtidas. Por fim é então apresentado ao utilizador as medias obtidas, realizando a conversão de milissegundos para minutos, segundos e milésimos de segundo.

Figura 5 - Apresentação das médias ao utilizador

### Apresentação do concorrente mais rápido e mais lento

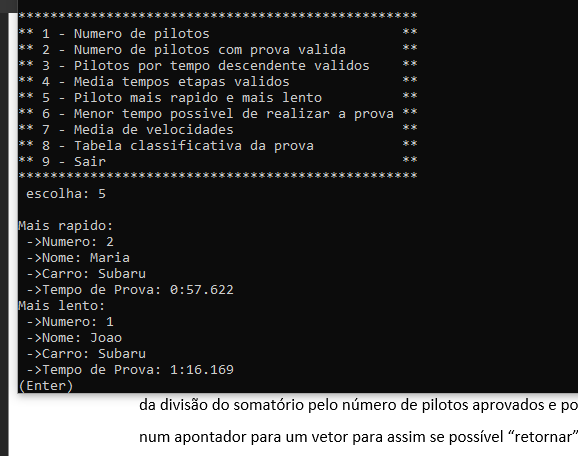
Para a verificação do concorrente mais rápido e mais lento, foi criada a função extremos, esta função contém duas variáveis essenciais que são a menor maior que contêm inicialmente o primeiro valor do vetor prova pois o primeiro valor vai sempre ser o menor valor lido até ao momento que contém um ciclo for para percorrer todo o vetor prova, dentro deste ciclo é verificado se o concorrente está aprovado e caso este esteja aprovado é verificado se o tempo na posição atual é menor ou maior do que o menor e maior tempo lido até ao momento, caso seja o numero de piloto é também guardado numa variável. Após a finalização do ciclo é apresentado ao utilizador as informações do piloto mais lento e do mais rápido e é realizado a conversão de milissegundos para minutos, segundos e milésimos de segundo.

Figura 6 - Amostra do piloto mais rápido e mais lento

### Cálculo do menor tempo possível de efetuar a prova

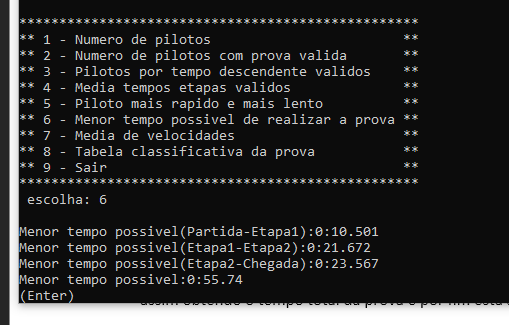
Para obter o menor tempo possível de efetuar a prova, seria o tempo do piloto mais rápido a terminar cada etapa, para então colocar este pensamento em prático foi criado um ciclo for que percorre o vetor tempos todo e verifica qual é o menor tempo de cada etapa, para verificar qual é o menor tempo de cada etapa, foram criadas três variáveis que guardam os menores tempos registados e são iniciadas com o primeiro valor do vetor tempos, após isso dentro do ciclo é verificada a etapa a que se refere o tempo da posição em que se encontra, após isso é comparado com o menor tempo antes registado e caso seja menor, o menor tempo recebe o valor o tempo da prova nesta posição. Para se obter o tempo da prova por completo são somadas as variáveis assim obtendo o tempo total da prova e por fim esta soma é apresentada ao utilizador convertida de milissegundos para minutos, segundos e milésimos de segundo.

Figura 7 - Apresentação do menor tempo possível de realizar a prova

### Listagem das velocidades médias de toda a prova

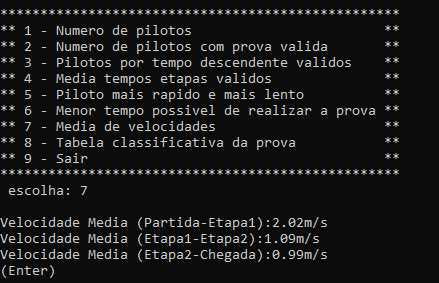
Para verificar as velocidades medias é primeiramente realizado um ciclo for que vai percorrer o vetor distâncias e verificar quais são as etapas e assim guardando estas distâncias numa variável relacionada à distância da etapa, seria então um exemplo *distanciaP\_E1*, após isso é reutilizado o vetor *medTempos* obtido na função *medTemposEtapa* e transformados esses tempos em segundos e assim guardados em variáveis referentes à sua etapa. A fórmula de cálculo da velocidade média é a distância percorrida dividindo pelo tempo que demorou a percorrer essa distância, ou seja, utilizando as variáveis atrás obtidas é aplicada a fórmula e guardado as velocidades médias num vetor e por fim apresentadas ao utilizador.

Figura 8 - Amostra das velocidades medias

### Apresentação da tabela classificativa

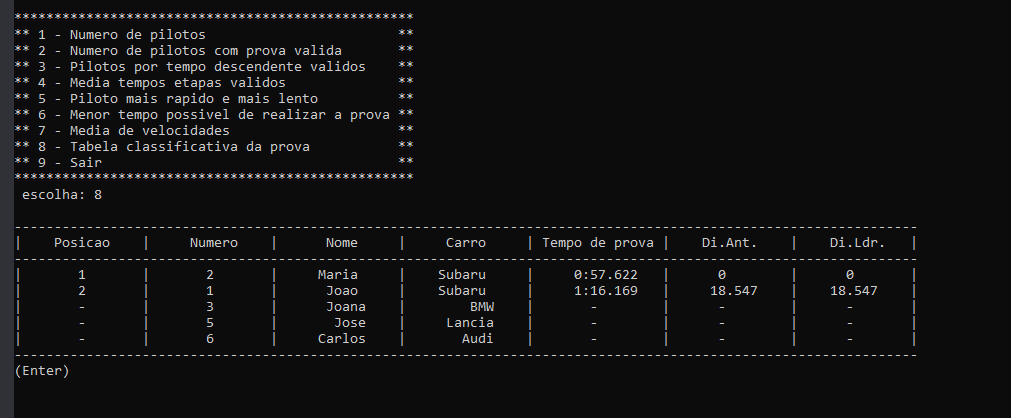
Para a realização desta tarefa é primeiramente ordenado o vetor prova para assim os valores ficarem ordenados dos pilotos mais rápidos para os pilotos mais lentos, reutilizando o método utilizado atrás para ordenar o vetor, mas agora será ordenado de forma crescente de tempo demorado. Após isso, é realizado um ciclo for para percorrer o vetor prova por completo. Dentro deste ciclo é verificado se o piloto é aprovado, caso seja, é obtido o seu tempo de prova e transformado em minutos e segundos, após isso caso seja o primeiro piloto, ou seja, o piloto mais rápido a diferença do líder e do anterior é 0 pois este é o líder e não tem piloto antes do mesmo e é então guardado o tempo do líder numa variável. Caso não seja o primeiro piloto aprovado, então a diferença do líder é calculada subtraindo o tempo do líder ao tempo do piloto e transformando em segundos, de seguida a diferença do anterior é obtida verificando o tempo do piloto do ciclo anterior e retirando o tempo do piloto anterior ao tempo do piloto do ciclo atual e transformando em segundos. Após os cálculos são então apresentadas todas a informações do piloto na prova. De seguida ao ciclo for para os pilotos aprovados, é realizado um ciclo for para os pilotos reprovados, dentro deste ciclo é verificado se o piloto não foi aprovado e caso se verifique é apenas apresentado na tabela o número de piloto, nome e carro do mesmo, estando as outras informações substituídas por ‘-‘.

Figura 9 - Apresentação da tabela classificativa

## Melhoria do projeto para aceitar mais informações variadas