# Trabalho Prático / Programação Imperativa

Carlos Brandão e Pedro Simões

2 de janeiro de 2023

**Resumo**

Este trabalho resume-se num programa que está relacionado com o bem-estar das pessoas, no contexto atual promove a necessidade de realizar atividades fisícas com regularidade. Para isso criamos um programa que controla e monitoriza atletas e as suas atividades.

Neste relatório, será possível encontrar passo a passo o código que criámos para o contexto explicado a cima, tendo também presente uma explicação mais aprofundada das funções presentes no programa para uma melhor compreensão do código exibido.

# Conteúdo

# Capítulo 1

# Introdução

Este relatório surge no âmbito do trabalho prático de Programação Imperativa do 1º Ano da Licenciatura de Engenharia de Sistemas Informáticos(Pós-Laboral).

O trabalho prático consiste em aplicar a matéria lecionada para a criação de um programa que visa apoiar, gerir e monitorar a realização de atividades físicas, pretendendo também promover as mesmas.

## 1.1 Objetivos

### 1.1.1 Objetivo A

O objetivo A é permitir ao utilizador que seja possível guardar e aceder aos dados sobre atletas, atividades e aos planos das atividades, dados quais estão armazenados em ficheiros.

### 1.1.2 Objetivo B

O objetivo B é aumentar o conhecimento sobre programação imperativa na linguagem C.

## 1.2 Estrutura do documento

No Capítulo 1 é introduzido o trabalho juntamente com a apresentação dos objetivos principais.

No Capítulo 2 é onde está presente o desenvolvimento do trabalho prático, neste capítulo vai ter uma explicação aprofundada nos passos que fizemos juntamente com imagens do código para acompanhar a explicação.

No Capítulo 3 é possível encontrar as dificuldades que encontramos durante o nosso trabalho.

No Capítulo 4 é possível encontrar as conclusões que tirámos ao realizar este trabalho.

# Capítulo 2

# Desenvolvimento

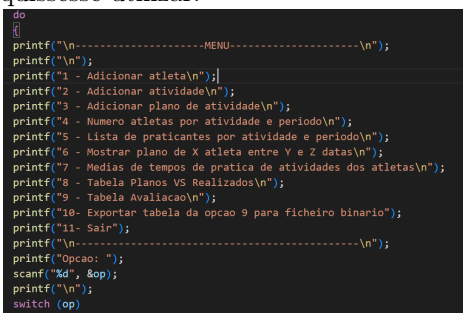
Nós começamos por examinar o enunciado e decidimos começar pela criação das estruturas que iriamos precisar ao longo deste trabalho.

Nós optamos por criar 3 estruturas, uma por cada ficheiro para uma melhor organização de código e dados.

```
typedef struct praticante
{
    char num[5];
    char nome[20];
    int tlm;
    int age;
} Praticante;

typedef struct atividade
{
    char num[5];
    char ativ[15];
    char data[20];
    int duracao, dist;
} Atividade;

typedef struct plano
{
    char num[5];
    char ativ[15];
    int dist;
    char data[20];
    char datain[20];
    char datafim[20];
    char hora[20];
    char horain[20];
    char horafim[20];
} Plano;
```

Figura 1 - Estruturas

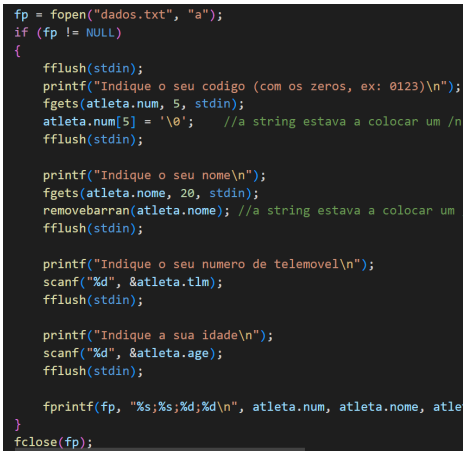
De seguida nós optamos por criar um menu onde fosse possível escolher a funcionalidade que quissesse utilizar.



```
do
{
printf("\n---------------------MENU---------------------\n");
printf("\n");
printf("1 - Adicionar atleta\n");
printf("2 - Adicionar atividade\n");
printf("3 - Adicionar plano de atividade\n");
printf("4 - Numero atletas por atividade e periodo\n");
printf("5 - Lista de praticantes por atividade e periodo\n");
printf("6 - Mostrar plano de X atleta entre Y e Z datas\n");
printf("7 - Medias de tempos de pratica de atividades dos atletas\n");
printf("8 - Tabela Planos VS Realizados\n");
printf("9 - Tabela Avaliacao\n");
printf("10- Exportar tabela da opcao 9 para ficheiro binario");
printf("11- Sair");
printf("\n------------------------------------------------\n");
printf("Opcao: ");
scanf("%d", &op);
printf("\n");
switch (op)
```

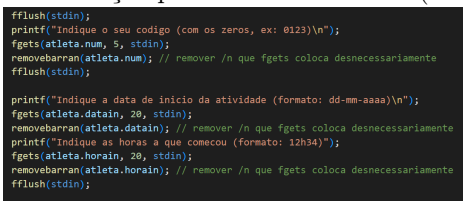
Figura 2 - Menu

Assim que concluimos estes passos, começamos a trabalhar nas funcionalidades do nosso programa, e foi assim que criamos a funções para adicionar dados aos três ficheiros.



```
fp = fopen("dados.txt", "a");
if (fp != NULL)
{
    fflush(stdin);
    printf("Indique o seu codigo (com os zeros, ex: 0123)\n");
    fgets(atleta.num, 5, stdin);
    atleta.num[5] = '\0';    //a string estava a colocar um /n
    fflush(stdin);

    printf("Indique o seu nome\n");
    fgets(atleta.nome, 20, stdin);
    removebarran(atleta.nome); //a string estava a colocar um
    fflush(stdin);

    printf("Indique o seu numero de telemovel\n");
    scanf("%d", &atleta.tlm);
    fflush(stdin);

    printf("Indique a sua idade\n");
    scanf("%d", &atleta.age);
    fflush(stdin);

    fprintf(fp, "%s;%s;%d;%d\n", atleta.num, atleta.nome, atle
}
fclose(fp);
```

Figura 3 - Função para adicionar atletas (Parte 1)



```
fflush(stdin);
printf("Indique o seu codigo (com os zeros, ex: 0123)\n");
fgets(atleta.num, 5, stdin);
removebarran(atleta.num); // remover /n que fgets coloca desnecessariamente
fflush(stdin);

printf("Indique a data de inicio da atividade (formato: dd-mm-aaaa)\n");
fgets(atleta.datain, 20, stdin);
removebarran(atleta.datain); // remover /n que fgets coloca desnecessariamente
printf("Indique as horas a que comecou (formato: 12h34)");
fgets(atleta.horain, 20, stdin);
removebarran(atleta.horain); // remover /n que fgets coloca desnecessariamente
fflush(stdin);
```

Figura 4 - Função para adicionar planos (Parte 1)

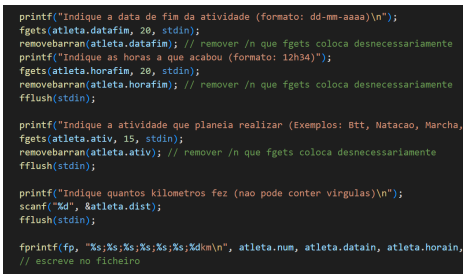



Figura 5 - Função para adicionar planos (Parte 2)

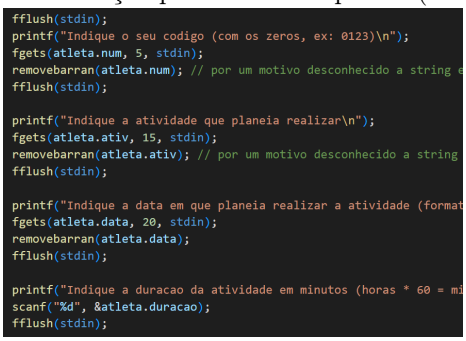



Figura 6 - Função para adicionar atividade (Parte 1)


Figura 7 - Função para adicionar atividade (Parte 2)

Foi apartir deste momento que demos início às funções do enunciado, começando por ordem criando assim a seguinte função:

Para podermos comparar, as datas nós transformamos as datas num *score*, em que o mês é 100x mais importante que o dia e ano é 2000x mais importante.

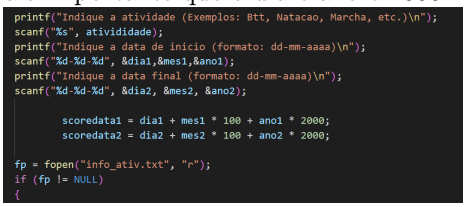



Figura 8 - Função para pesquisa de atletas (Parte 1)

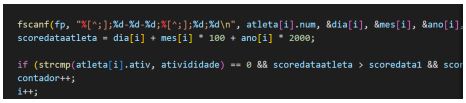



Figura 9 - Função para pesquisa de atletas (Parte 2)

Nós tivemos que trocar o cronograma pois nós tivemos que criar uma função porque por norma a função *fgets* coloca um /n no final da *string*. A função

*removebarran* analisa a string vinda de um *fgets* carater por carater , até encontrar o /n e substituindo-o por /0.


Figura 10 - Função para remover

De volta ao cronograma, codificamos uma função que apresenta o plano de atividades de um determinado tipo, para um determinado período e praticante.

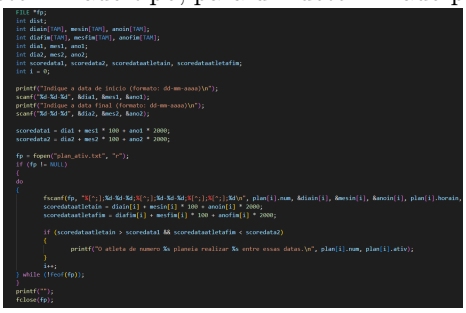



Figura 11 - Função para mostrar plano

De seguida, formulámos o código para calcular a média dos tempos dos praticantes que estiveram envolvidos em atividades fisícas.


Figura 12 - Função para calcular média

Na fase final do trabalho nós codificamos duas tabelas.

A primeira tabela mostra uma compração entre os planos do atleta e a atividade que o mesmo realizou.


Figura 13 - Função para a primeira tabela

Para a segunda tabela são pedidas duas datas, uma inicial e uma final, e na tabela são listados os atletas que realizaram uma atividade entre essas datas, mostrando distância percorrida, a duração e a data em que a atividade foi praticada.

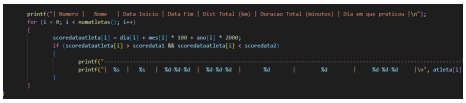


Figura 14 - Função para a segunda tabela

Para concluir o programa, nós codificamos uma função que envia a tabela anterior para um ficheiro binário.

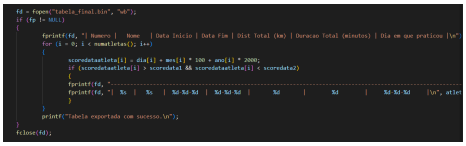


Figura 15 - Função para a exportação da tabela

# Capítulo 3

# Dificuldades Encontradas

Ao longo do trabalho, deparamo-nos com diversas dificuldades sendo as principais os erros de escrita nos ficheiros, onde eram inseridos /n's não pedidos. Na função *planativatleta*, ao listar a informação no terminal a informação tirada da primeira linha do ficheiro de texto não era lida corretamente. A utilização das datas também foi um desafio, visto que nós tivemos que fazer várias comparações entre as mesmas.

# Capítulo 4

# Conclusão

Este trabalho prático, colocou-nos vários obstáculos pela frente tendo que os ultrapassar e aumentando assim os nossos conhecimentos. Aprendemos mais sobre trabalhar em equipa e em programar em C.