
Relatório do Trabalho Prático

Laboratórios de informática



David Lopes - 26017
Afonso Ferreira - 27981
Paulo Costa - 29851

Professor: Óscar Ribeiro
Curso: Engenharia de Sistemas Informáticos

Índice

1	Introdução	1
2	Divisão de Tarefas	2
3	Descrição do código	3
3.1	Funções Implementadas	3
3.1.1	Importação de Dados	3
3.1.2	Gestão de Dados Nutricionais	3
3.1.3	Análise de Dados Nutricionais	3
3.1.4	Tabela das Refeições Planeadas e Realizadas	4
3.2	Exemplo de código	4
3.2.1	Função para Importar Planos	4
3.3	Makefile	6
4	Github	7
4.1	README.md	7
4.2	Configuração do .gitignore	9
5	Conclusão	10
	Bibliografia	11

Lista de Figuras

4.1	GitHub do trabalho de grupo	7
4.2	README.md do repositório	8

Lista de Códigos

3.1	Função para Importar Planos	4
3.2	Ficheiro do Makefile	6
4.1	Ficheiro .gitignore	9

Capítulo 1

Introdução

Este trabalho tem como objetivo o desenvolvimento de um programa na linguagem C e a sua respetiva documentação.

O programa visa gerir o comportamento nutricional de pacientes de um gabinete de nutrição, controlando e monitorizando a realização de uma deita alimentar saudável. O projeto permitirá carregar, processar e analisar todos os dados nutricionais de cada paciente.

Capítulo 2

Divisão de Tarefas

As tarefas deste projeto foram divididas de forma equitativa e justa.

Todos os membros do grupo ajudaram-se mutuamente, ou seja, todos fizeram um pouco de tudo, desde a criação do repositório, até à concessão deste relatório. Todas as decisões foram de igual forma feitas em grupo, assim como as trocas de ideias, permitindo que o trabalho ficasse mais completo e de acordo com as expectativas de cada um dos membros.

Capítulo 3

Descrição do código

3.1 Funções Implementadas

Neste projeto foi proposto criar um programa com o objetivo de manipular os dados de pacientes de um gabinete de nutrição. Para isso implementamos as seguintes funções:

3.1.1 Importação de Dados

Implementámos várias funções que permitem importar dados de pacientes, dietas e planos de modo a que o programa consiga, a partir de um ficheiro .csv, ler os dados lá armazenados e os poder utilizar.

3.1.2 Gestão de Dados Nutricionais

O programa permite gerir todos os dados nutricionais de todos os pacientes. Os dados nutricionais vão desde planos personalizados para satisfazer as necessidades individuais de cada paciente, definindo num período de tempo, um limite de calorias máximo e mínimo para cada refeição, até à dieta que um determinado paciente realmente consumiu num determinado dia, numa determinada refeição e as calorias consumidas nessa dieta.

3.1.3 Análise de Dados Nutricionais

Para além de guardar todos os dados, o programa também possui algumas funcionalidades que permitem analisar esses mesmos dados. Isto inclui a apresentação do número de pacientes que ultrapassaram determinada quantidade de calorias, num determinado período de tempo, a listagem dos pacientes ordenada por ordem decrescente do número de paciente que realizaram alguma refeição com quantidade de calorias fora do intervalo (de acordo com o plano nutricional) num determinado período de tempo, a listagem do plano nutricional de um paciente para determinada refeição ao longo de um determinado

período de tempo e o cálculo das médias da calorias consumidas por refeição por cada paciente ao longo de um determinado período de tempo.

3.1.4 Tabela das Refeições Planeadas e Realizadas

Para melhor análise dos dados guardados, o programa tem a funcionalidade de mostrar uma tabela com o resumo das refeições planeadas e das refeições já realizadas, comparando as calorias consumidas numa refeição com o plano associado a essa refeição.

3.2 Exemplo de código

3.2.1 Função para Importar Planos

Como exemplo usaremos a função ImportarPlanos que tem como objetivo ler dados de planos guardados num ficheiro .csv e guardá-los numa estrutura de dados. O código lê cada linha do ficheiro, em que cada parâmetro está separado por ";" e traduz as datas de formato "dd/mm/yyyy" para uma variável do tipo "time_t".

```
int ImportarPlanos(Plano planos[], char filename[], Paciente
    pacientes[]) {
    FILE* fp;
    fp = fopen(filename, "r");
    if (fp == NULL) return 0;

    //strings que ser o recebidas para depois converter para
    time_t
    char dataInicioString[N];
    char dataFimString[N];

    //structs tm auxiliares para converter para time_t
    struct tm dataInicio = {0};
    struct tm dataFim = {0};

    int i = 0;
    while (1)
    {

        fscanf(fp, "%d;%[^;];%[^;];%d;%d;%d\n", &planos[i].
            numPaciente, dataInicioString, dataFimString, &
            planos[i].refeicao, &planos[i].minCal, &planos[i].
            maxCal);

        //converte as strings struct tm
        sscanf(dataInicioString, "%d/%d/%d", &dataInicio.
            tm_mday, &dataInicio.tm_mon, &dataInicio.tm_year);
```



```
        sscanf(dataFimString, "%d/%d/%d", &dataFim.tm_mday, &
            dataFim.tm_mon, &dataFim.tm_year);

        //ajusta os valores para os valores esperados na
        struct tm
        dataInicio.tm_mon -= 1; //Para o m s ficar 0-11
        dataInicio.tm_year -= 1900; //Para o ano come ar em
            1900

        dataFim.tm_mon -= 1; //Para o m s ficar 0-11
        dataFim.tm_year -= 1900; //Para o ano come ar em
            1900

        //converter para a vari vel time_t
        planos[i].dataInicio = mktime(&dataInicio);
        planos[i].dataFim = mktime(&dataFim);

        //AssociaPlano(planos[i], pacientes);
        if (feof(fp)) break;
        i++;
    }
    fclose(fp);
    return 1;
}
```

Código 3.1: Função para Importar Planos

3.3 Makefile

O ficheiro *Makefile*[1] é usado na compilação do programa desenvolvido. O ficheiro permite, que a cada vez que compilarmos o código, o mesmo é compilado da maneira mais rápida e eficiente.

Foi configurado para facilitar a compilação, a criação de executáveis e a limpeza de arquivos temporários por definição de comandos específicos.

Resumindo, permite uma rápida compilação e limpeza de arquivos temporários desnecessários na execução final.

```
CC = gcc
CFLAGS = -Wall -Wextra -std=c99

SRC = Source.c Funcoes.c
OBJ = $(SRC:.c=.o)
TARGET = programa

all: $(TARGET)

$(TARGET): $(OBJ)
    $(CC) $(CFLAGS) -o $@ $^

%.o: %.c
    $(CC) $(CFLAGS) -c $< -o $@

# make clean to remove all generated .o files and the executable
clean:
    rm -f $(OBJ) $(TARGET)
```

Código 3.2: Ficheiro do Makefile

Capítulo 4

Github

Neste trabalho usamos o *GitHub*[2], não só como uma maneira fácil de gerir o código, mas também para a colaboração dos membros do grupo.

Para isso criamos um repositório público (<https://github.com/pauloarcosta03/Programacao-Imperativa>) para armazenar todas as alterações feitas.

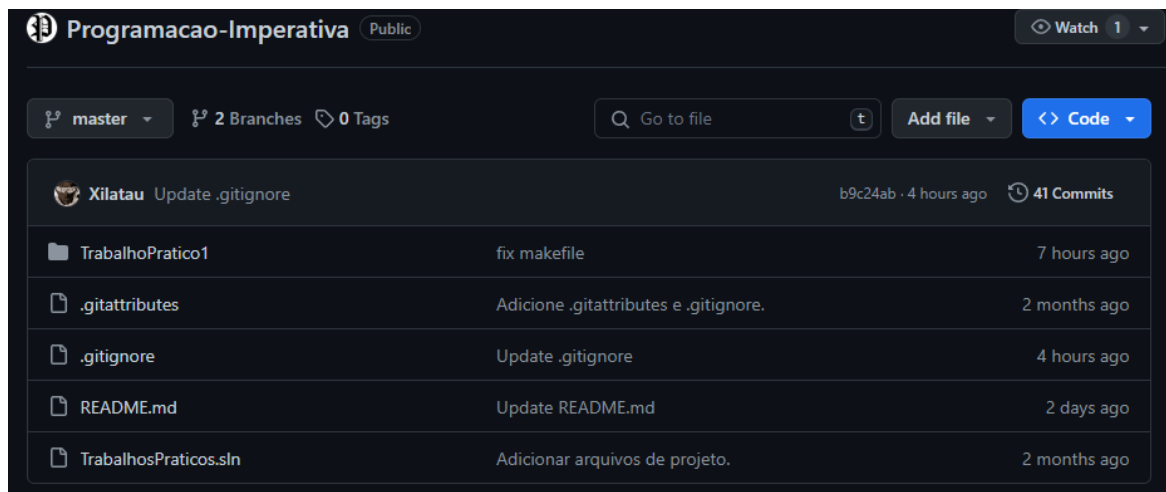
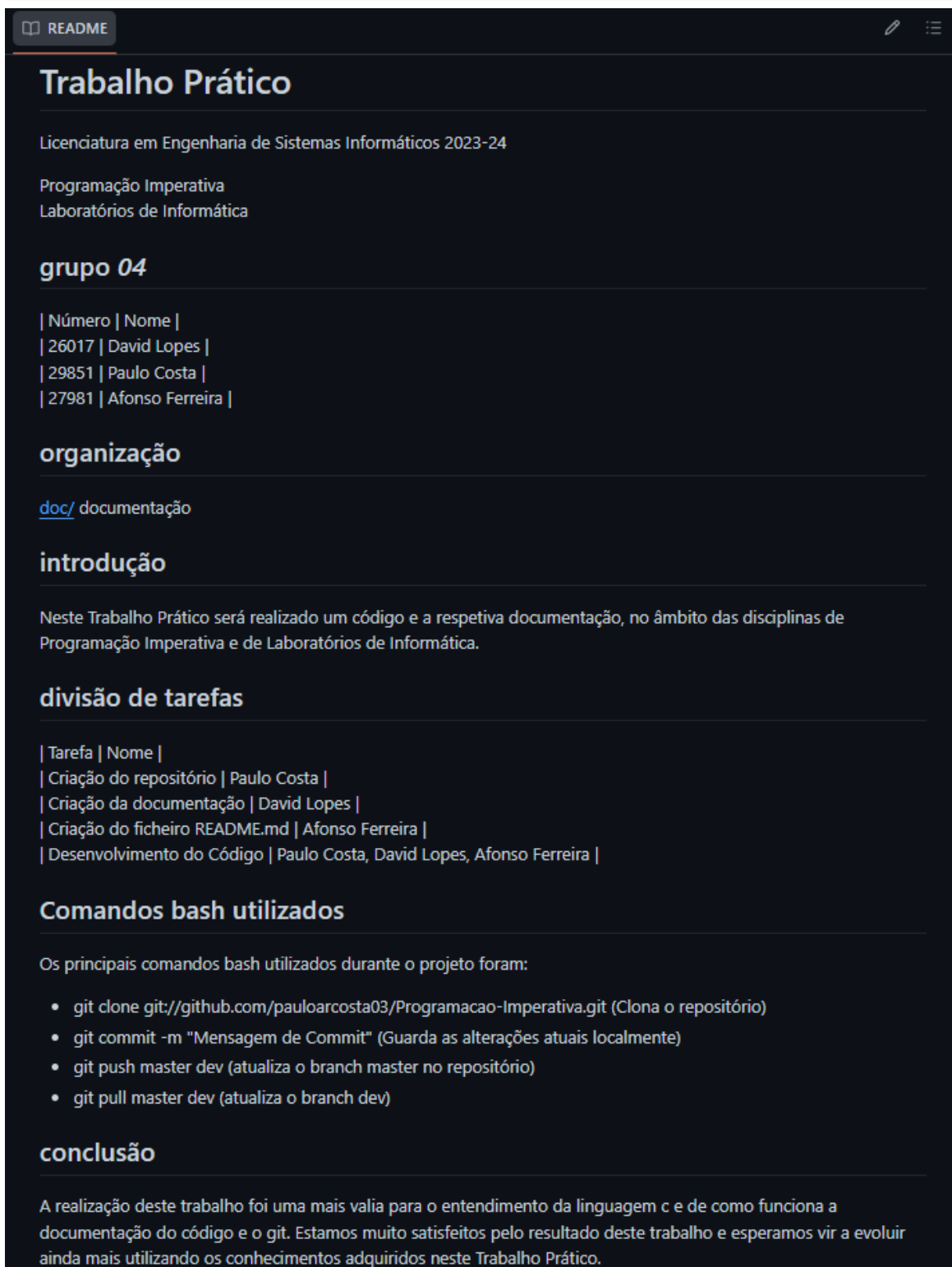


Figura 4.1: GitHub do trabalho de grupo

4.1 README.md

O ficheiro README.md é um arquivo que utiliza a notação *Markdown*[3] e que fornece informações cruciais sobre o propósito, instalação e uso do código, assim facilitando a compreensão e a colaboração para o código do projeto.



The image shows a screenshot of a GitHub repository's README file. The interface is dark-themed. At the top, there's a header with 'README' and some icons. The main content is organized into sections: 'Trabalho Prático', 'Licenciatura em Engenharia de Sistemas Informáticos 2023-24', 'Programação Imperativa', 'Laboratórios de Informática', 'grupo 04', 'organização', 'introdução', 'divisão de tarefas', 'Comandos bash utilizados', and 'conclusão'. The 'grupo 04' section contains a table with student information. The 'divisão de tarefas' section contains a table with task assignments. The 'Comandos bash utilizados' section lists several git commands. The 'conclusão' section provides a summary of the work.

README

Trabalho Prático

Licenciatura em Engenharia de Sistemas Informáticos 2023-24

Programação Imperativa
Laboratórios de Informática

grupo 04

Número	Nome
26017	David Lopes
29851	Paulo Costa
27981	Afonso Ferreira

organização

[doc/](#) documentação

introdução

Neste Trabalho Prático será realizado um código e a respetiva documentação, no âmbito das disciplinas de Programação Imperativa e de Laboratórios de Informática.

divisão de tarefas

Tarefa	Nome
Criação do repositório	Paulo Costa
Criação da documentação	David Lopes
Criação do ficheiro README.md	Afonso Ferreira
Desenvolvimento do Código	Paulo Costa, David Lopes, Afonso Ferreira

Comandos bash utilizados

Os principais comandos bash utilizados durante o projeto foram:

- git clone git://github.com/pauloarcosta03/Programacao-Imperativa.git (Clona o repositório)
- git commit -m "Mensagem de Commit" (Guarda as alterações atuais localmente)
- git push master dev (atualiza o branch master no repositório)
- git pull master dev (atualiza o branch dev)

conclusão

A realização deste trabalho foi uma mais valia para o entendimento da linguagem c e de como funciona a documentação do código e o git. Estamos muito satisfeitos pelo resultado deste trabalho e esperamos vir a evoluir ainda mais utilizando os conhecimentos adquiridos neste Trabalho Prático.

Figura 4.2: README.md do repositório

4.2 Configuração do .gitignore

O ficheiro `.gitignore`[4] foi configurado para ignorar tipos de ficheiros especificados no mesmo, impedindo assim que esses ficheiros não sejam incluídos no repositório.

```
Ignorar ficheiros . txt
*.txt

Ignorar ficheiros da compilacao
.exe
.o
```

Código 4.1: Ficheiro `.gitignore`

Esta parte do ficheiro `.gitignore`, previne que ficheiros do tipo `.txt`, `.exe` e `.o` sejam incluídos no repositório depois de um "commit".

Capítulo 5

Conclusão

A realização deste trabalho foi uma mais valia para o entendimento da linguagem c e de como funciona a documentação do código e o git. Estamos muito satisfeitos pelo resultado deste trabalho e esperamos vir a evoluir ainda mais utilizando os conhecimentos adquiridos neste Trabalho Prático.

Bibliografia

- [1] A Short Introduction to Makefile.
- [2] The GitHub Blog.
- [3] Guia básico de Markdown.
- [4] gitignore Documentation.