UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática

Princípios de Programação Procedimental

Projeto 2017/18

Quadro de Tarefas - Kanban

Projeto realizado por:

Maria Carolina Gonçalves-2017247573

Ricardo Martins- 2017246268

TP8

INTRODUÇÃO

O objetivo do projeto era criar um quadro de *Kanban*, que permite a gestão de tarefas em processos baseados em pipeline, usando listas ligadas e ficheiros.

Para a execução do trabalho usamos ficheiros para guardar informação ao fechar o programa e recuperar os dados ao iniciar. Relativamente às listas optamos por usar uma lista de pessoas, uma lista de tarefas, que contém todas as tarefas ordenadas por data de criação e três listas de apontadores para tarefas, cada uma delas representa uma fase do pipeline (“to do”, “doing” e “done”). Optamos pelo uso de três listas diferentes para representar as fases do pipeline, devido ao facto de cada fase ser ordenada de forma diferente, decidimos também usar listas de apontadores para tarefas para representar as diferentes fases do pipeline de forma a facilitar as trocas entre fases, ou seja, para evitar copiar informação. Numa tentativa de minimizar os erros causados na introdução de dados por parte do utilizador, o id da tarefa é gerado automaticamente e sequencialmente, a começar em 1. Para simplificar o programa as tarefas do responsável estão sempre na fase “doing”, ou seja, as tarefas que não estão na lista “doing” não aparecem nas tarefas do respetivo responsável, mesmo que tenha sido este a termina-la, assim as tarefas que se encontram acabadas não entram na contagem das tarefas atribuídas a uma pessoa. O utilizador tem acesso aos ids gerados ao imprimir as tarefas, no entanto, não as pode alterar em nenhum momento. Este projeto segue também o pressuposto que o utilizador não vai alterar os ficheiros de texto relativos às tarefas.

Segue-se uma breve explicação das estruturas e funções utilizadas para o desenvolvimento do projeto, bem como o funcionamento do mesmo.

Estruturas utilizadas

Em primeiro lugar, temos uma lista com todas as tarefas, sendo que cada tarefa, tem uma estrutura, cartão, com a informação relativa à mesma e tem um apontador para o responsável dessa tarefa;

```
typedef struct {
    int dia, mes, ano;
}Data;

typedef struct {
    int id, prioridade, estagio, id_responsavel;
    char descricao[MAX_DESC];
    Data dt_criacao, prazo, dt_fim;
}Cartao;

typedef struct tarefas *List_tarefas;
typedef struct responsavel *List_pessoas;
typedef struct tarefas {
     Cartao info;
     List_pessoas resp_tarefa;
     List_tarefas next;
}Tarefas;
```

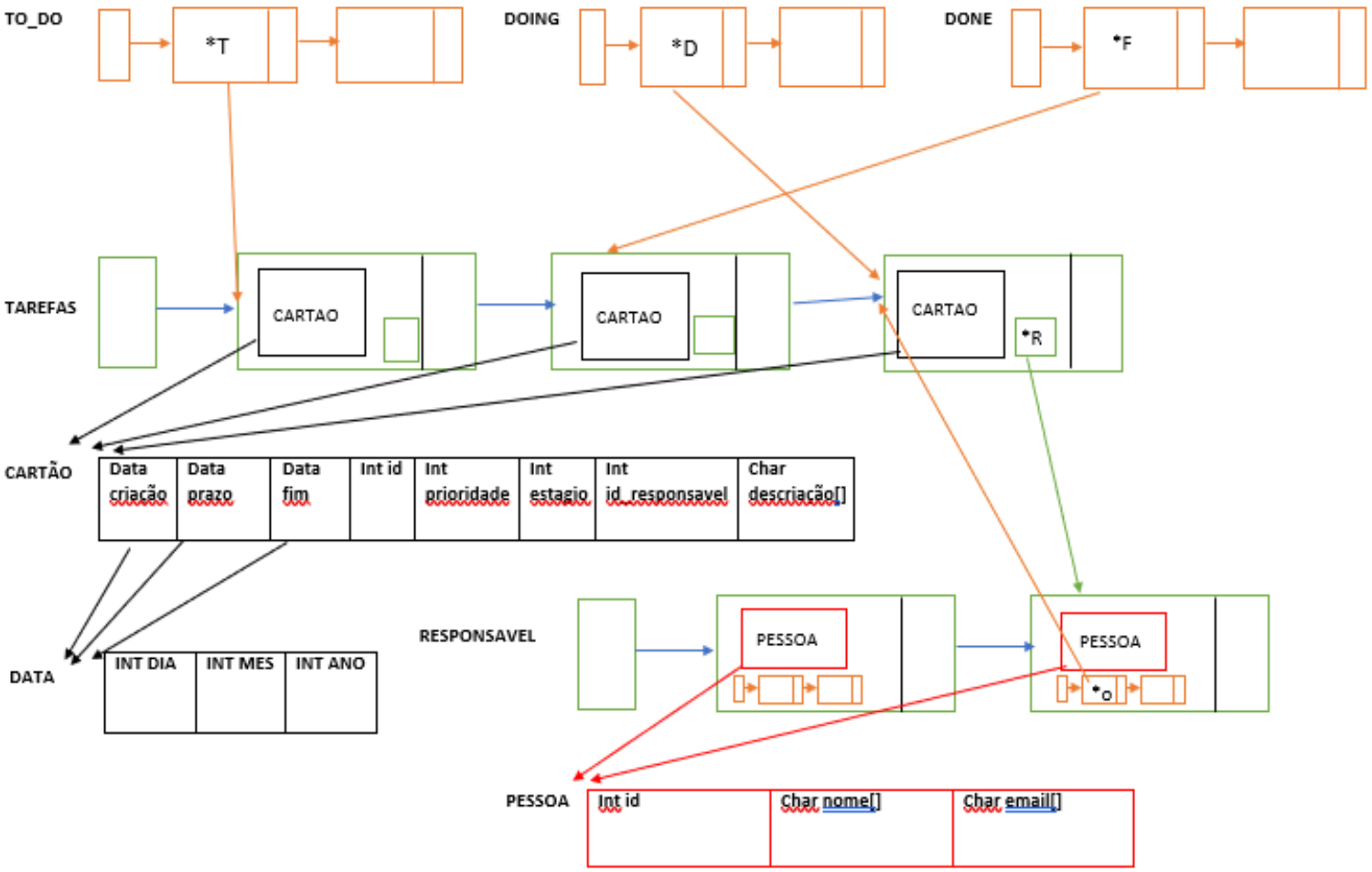
De seguida, uma lista com todas as pessoas, dado que cada pessoa tem uma estrutura com toda a sua informação. Para além disso, cada pessoa contém uma lista de apontadores que apontam para as tarefas pelo qual estão responsáveis;

```
typedef struct {
   char nome[MAX_NOME], email[TAM_EMAIL];
   int id;
}Pessoa;
typedef struct responsavel{
    Pessoa info;
    List_apont_tarefas tarefas_correspondentes;
    List_pessoas next;
}Responsavel;
```

Por fim, as nossas listas, to_do, doing e done são listas ligadas que contêm um apontador para uma lista de tarefas.

```
typedef struct lista_ponteiros* List_apont_tarefas;
typedef struct lista_ponteiros{
    List_tarefas ponteiro;
    List_apont_tarefas next;
}List_apontadores;
```

LISTAS_APONTADORES:



Estrutura geral do programa:

Para o programa recorremos às seguintes bibliotecas:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

Constantes e variáveis globais:

```
#define MAX_TAREFAS_DOING 5
#define MAX_DESC 200
#define MAX_NOME 30
#define TAM_EMAIL 30

int contador_id;
int max_tarefas;
int contador_tarefas_doing;
```

O nosso programa está dividido em 5 ficheiros:

- O main.c

Funções (PARAMETROS){	**Utilidade**
Int main()	Criação das listas de tarefas, apontadores e de pessoas, é a função onde esta o loop que permite o funcionamento do programa, chama o menu e destrói as listas antes de terminar o programa.
Int menu_inicial (List_tarefas* tarefas, List_pessoas* pessoas, List_apont_tarefas* to_do, List_apont_tarefas* doing, List_apont_tarefas* done)	Apresenta todas as opções que o utilizador pode selecionar, chamando posteriormente, as funções apropriadas à realização da opção.
int alterar_max_tarefa()	Altera o máximo de tarefas permitidos por pessoa.

- Header.h – Neste ficheiro declaramos todas as funções que usamos no programa.

- func_listas.c

Funções(PARAMETROS)	**Utilidade**
List_apont_tarefas criar_todo (List_apont_tarefas to_do, List_tarefas novo)	Cria um nó com um apontador para determinada tarefa (novo) e de seguida insere na lista to_do.
List_apont_tarefas inserir_to_do (List_apont_tarefas lista, List_apont_tarefas novo)	Insere o nó (novo) por ordem de prioridade na lista de apontadores para tarefas (lista).
List_apont_tarefas criar_doing (List_apont_tarefas doing, List_tarefas novo, List_pessoas pessoas)	Cria um nó com um apontador para determinada tarefa(novo) e de seguida insere na lista doing.
List_apont_tarefas inserir_doing (List_apont_tarefas lista, List_apont_tarefas novo, List_pessoas pessoas	Insere o nó (novo) por ordem alfabética na lista de apontadores para tarefas (lista). A função começa por pedir um prazo para a tarefa, seguido de um responsável para atribuir a tarefa a começar, após obter a confirmação de que o responsável escolhido pode receber a tarefa esta função vai inserir a tarefa na lista por ordem alfabetica relativa aos responsáveis. Através do uso da função ordenar_alfabetica.

int verificar_responsavel (List_apont_tarefas novo, List_pessoas resp)	Verifica se o responsável escolhido esta na lista de pessoas, e no caso de existir vê se este não excedeu o número max_tarefas e se o prazo da respetiva tarefa tem uma diferença de 7 dias das tarefas desse responsável. Retorna 1 se a pessoa escolhida reunir as condições necessárias para a tarefa, caso contrario a função retorna 0.
List_apont_tarefas ordenar_alfabetica (List_apont_tarefas lista, List_apont_tarefas novo)	Insere um nó com um apontador para uma tarefa, por ordem alfabética com a ajuda da função comparar_strings.
int comparar_strings (char a[],char b[])	Compara as strings e devolve um determinado valor de x. (PRECISO DE SABER OS VALORES DE X)
List_apont_tarefas criar_done (List_apont_tarefas done, List_tarefas novo)	Tem a mesma funcionalidade de criar_todo e criar_doing, mas neste caso cria um nó na lista done.
List_apont_tarefas inserir_done (List_apont_tarefas lista, List_apont_tarefas novo)	Insere um nó com um apontador para uma tarefa (novo) por ordem de data de fim numa lista (lista).
void alterar_responsavel (List_apont_tarefas tarefa, List_pessoas pessoas)	Altera o responsável de uma tarefa utilizando também as funções pesquisa_pesso, verificar_responsavel, remover_tarefa e inserir_to_do, visto que queremos remover o nó do antigo responsável e inseri-lo na lista tarefas da pessoa a quem queremos reatribuir a tarefa.
List_apont_tarefas remover_tarefa (List_apont_tarefas* lista_actual, int id_tarefa)	Remove o nó do tipo List_apont_tarefas da lista indicada no primeiro parâmetro mantendo a continuidade desta.
List_tarefas alterar_prioridade (List_tarefas novo)	Altera prioridade de uma tarefa.
List_tarefas criar_lista (List_tarefas* lista)	Cria nó do tipo List_tarefa preenchendo-o com a ajuda da função criar_tarefa chamando posteriormente a função inserir_ordenado_data.
List_tarefas inserir_ordenado_data (List_tarefas al, List_tarefas novo)	Vai inserir um nó com uma tarefa na(novo) na lista de tarefas (al) por ordem de data de criação.
void criar_tarefa (Cartao *n)	Serve para preencher a informação essencial da tarefa.
List_apont_tarefas pesquisa_lista_apont (List_apont_tarefas lista, int id)	Procura e retorna, se encontrar, um nó numa lista de apontadores para tarefas.
List_pessoas pesquisa_pessoa (List_pessoas lista,int id)	Procura e retorna, se encontrar, um nó numa lista de pessoas.

List _tarefas pesquisa_lista (List_apont_tarefas lista, int id)	Procura e retorna, se encontrar, um nó numa lista de tarefas.
void imprimir_pessoa (List_pessoas lista)	Imprime a lista de pessoas (lista).
void imprimir_apontadores (List_apont_tarefas lista)	Imprime lista de apontadores.
void imprimir_tarefas (List_tarefas lista)	Imprime lista de tarefas.
List_tarefas destroi_lista(List_tarefas lista)	Liberta a memória alocada para as listas ligadas do tipo List_tarefas.
List_apont_tarefas destroi_apontadores(List_apont_tarefas lista)	Liberta a memória alocada para as listas ligadas do tipo List__apont_tarefas.

- Ficheiros.c

Funções (PARAMETROS)	**Utilidade**
List_tarefas carregar_listas (List_tarefas lista, List_pessoas pessoas, List_apont_tarefas* to_do, List_apont_tarefas* doing, List_apont_tarefas* done)	Vai ler do ficheiro que contém todas as tarefas e de seguida chama a função inserir_ordenado_data para preencher a lista que tem todas as tarefas. Esta função coloca a tarefa na lista de apontadores correspondente, ou chamando a criar_todo, ou inserir_doing_ficheiro, ou inserir_done_ficheiro.
List_apont_tarefas inserir_doing_ficheiro (List_apont_tarefas lista, List_tarefas novo, List_pessoas pessoas)	Cria um nó com um apontador para a tarefa, cria as ligações responsável tarefa e vice versa colocando um apontador na lista de tarefas_correspondentes do responsável, e insere o nó na lista doing
List_apont_tarefas inserir_done_ficheiro (List_apont_tarefas lista, List_tarefas novo, List_pessoas pessoas)	Cria um nó com um apontador para a tarefa, cria a ligação da tarefa com o responsável e insere o nó na lista done.
List_pessoas criar_pessoas (List_pessoas lista)	Lê as informações de cada pessoa a partir do ficheiro.
List_pessoas inserir_pessoa(List_pessoas lista, List_pessoas novo)	Insere um nó (novo) na lista de pessoas
void guardar_lista (List_tarefas lista)	Guarda a lista de tarefas num ficheiro auxiliar.

void colocar_ficheiro (List_apont_tarefas to_do, List_apont_tarefas doing, List_apont_tarefas done)	Esta função vai colocar, num ficheiro de texto, o quadro que tem todas as tarefas criadas dividas pela fase do pipeline me que se encontram, recorrendo á função imprimir_apontadores_ficheiros.
void imprimir_apontadores_ficheiros (List_apont_tarefas lista, FILE *fp)	Imprime no ficheiro a informação da lista de apontadores.

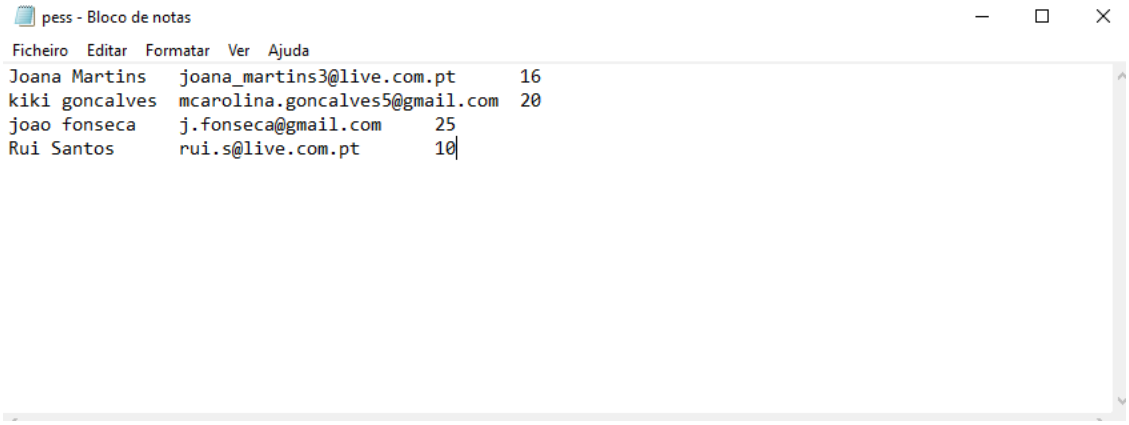
- Datas.c

Funções (PARAMETROS)	**Utilidade**
void preencher_data (Data* dt)	Garante que a data inserida é válida
int diferenca_dias (Data *d1, Data *d2)	Esta função utilizada para ver se duas datas têm diferença de 7 dias, função utilizada na inserir_doing, Retorna 1 se a diferença de dias for maior ou igual a 7, caso contrario retorna 0.
int comparar_datas (Data *d1, Data *d2)	Verifica se a 1ªdata é superior (retorna 1), igual (retorna 0) ou anterior à 2ªdata (retorna -1).

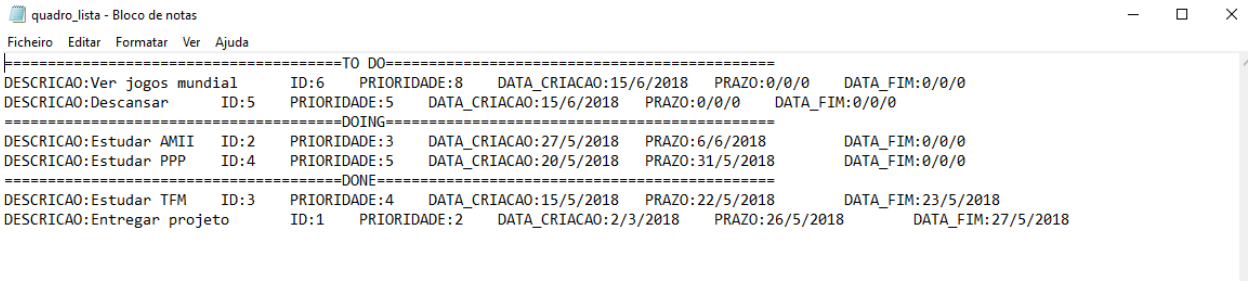
Estrutura dos ficheiros de texto:

Em alternativa ao uso de bases de dados usamos diversos ficheiros auxiliares que guardam as informações das tarefas já criadas, dos dados das pessoas e dos valores necessários para o funcionamento correto do programa. Cada tarefa/pessoa corresponde a uma linha e os diferentes dados de cada estrutura estão separados por um **tab**.

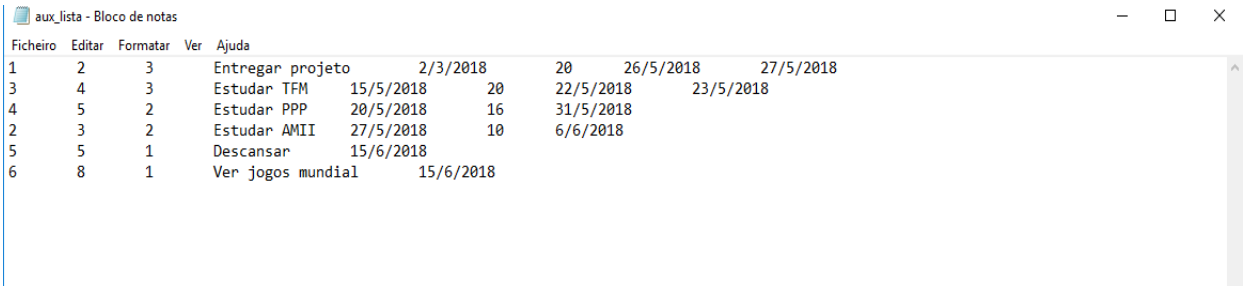
Um dos ficheiros contém as pessoas e a sua informação, é necessário salientar que este é o único ficheiro de texto que pode ser alterado pelo utilizador. Uma vez que não existe forma de adicionar ou remover pessoas durante a execução do programa, o utilizador pode, respeitando o formato prévio, adicionar novas pessoas e alterar a informação das pessoas já existentes (com exceção do identificador pessoal).



De seguida, criamos outro ficheiro, este contém o último quadro de tarefas guardado, este ficheiro serve apenas para o utilizador poder visualizar o quadro e poder comparar.



Para além desse quadro, existe um ficheiro que guarda todas as tarefas de um modo simples, uma tarefa em cada linha, o estagio de pipeline de cada tarefa é representado por um numero neste ficheiro, o 1 representa a lista “to do”, o 2 representa o “doing” e finalmente o 3 representa o “done”. Este ficheiro serve apenas para guardar informação vital para o programa.



Por fim guardamos o máximo de tarefas num ficheiro, pois sendo possível alterar esse valor, não o queremos perder após terminar a execução, e ao voltar a abrir o programa, voltamos a carregar este valor evitando assim conflitos de dados.

Execução do programa:

Ao iniciar o programa, criamos a lista das tarefas, a lista das pessoas e a lista de apontadores to_do, doing, done. Estas são carregadas com informação através das funções carrega_listas e cria_pessoas.

De seguida, o programa mostra o menu e as suas opções (total de 13 opções).

- Ao selecionar a primeira opção, o programa corre a função imprime_pessoas, permitindo ao utilizador verificar todas as informações de cada pessoa (Nome, e-mail e identificador pessoal);

- Se o utilizador selecionar a segunda opção, pode alterar o número máximo de tarefas que podem ser atribuídas a uma única pessoa. Para realizar esta operação recorremos apenas à função alterar_max_tarefa;

- Na terceira opção do menu, vamos criar um novo nó com uma tarefa na lista de tarefas e, simultaneamente, criar um nó na função to_do, que contém um apontador para a tarefa criada. Para isso, o programa utiliza a função criar_lista. Após fazer a atribuição do número do estágio, a função criar_todo é utilizada;

- A quarta opção permite alterar a prioridade de uma tarefa específica. Para realizar esta operação é pedido o id da tarefa à qual quer alterar a prioridade ao utilizador. Após ler o valor de id, vamos procuramos essa tarefa através da pesquisa_lista e alteramos a sua prioridade, com a ajuda da função alterar_prioridade. Se esta tarefa estiver na lista to_do, iremos retirá-la dessa lista e voltá-la a introduzir, usando, respeticamente, as funções remover_tarefa e inserir_to_do para garantir que a lista continua ordenada, apenas se realiza esta parte se a tarefa estiver na lista to_do porque esta é a única que lista ordenada por prioridade

- Ao escolher a quinta possibilidade, o utilizador pode começar uma tarefa, ou seja, passar da lista to_do para a lista doing. É pedido, ao utilizador, o id da tarefa que pretende começar e o responsável que queremos atribuir a tarefa. De seguida é retirada da lista to_do através da função remover_tarefa e imprimimos essa tarefa para a pessoa verificar se é a tarefa certa. Para inserir uma tarefa na lista doing é necessário verificar uma série de condições como por exemplo, a lista doing não pode estar cheia e o responsável tem de estar disponível para receber a tarefa. Apenas depois destas verificações podemos colocar o nó retirado do to_do na lista doing recorrendo a inserir_doing, ainda nesta fase temos de atribuir o responsável pela tarefa, e adicionar um nó de apontadores para tarefas (List_apont_tarefas), às tarefas do responsável para assegurar a ligação entre responsável e tarefa, para esta ultima fase usamos a função criar_todo, que nos permite criar um nó para inserir uma tarefa na lista de tarefas do responsavel e também garante que as tarefas ficam ordenadas por ordem de prioridade.

- A opção seis permite parar uma tarefa em execução, ou seja, passar a tarefa de doing para to_do. É novamente pedido o id da tarefa que se pretende devolver para a lista to_do ao utilizador. Para trocar o estagio desta tarefa temos de confirmar que ela de facto de encontra na lista doing, usamos a função remover tarefa, removemos as ligações entre tarefa e o respetivo responsável para libertar o espaço de tarefas do responsável, depois usamos a função inserir_to_do para colocar corretamente a tarefa na lista to_do. Finalmente temos de reduzir em um a variável global contador_tarefas_doing (-1), para libertar o espaço da tarefa na lista doing uma vez que este tem um máximo de 5;

- Se o utilizador escolher a sétima opção do menu, poderá alterar o responsável de uma tarefa que esteja na lista doing. Para isso, é mais uma vez pedido ao utilizador a id da tarefa que pretende alterar. Após verificar que esse tarefa existe e se encontra na lista doing, removemos o nó da lista usando a função remover_tarefa, alteramos o responsável com a função alterar_responsavel, que recorrendo à função verificar_resonsavel nos permite ter a certeza que o novo responsável está disponível para receber a tarefa, se verificarem todas as condições necessárias para alterar o responsável, voltamos a inserir a tarefa na lista doing, usamos a função ordenar_alfabetica que nos permite colocar a tarefa numa lista ordenada alfabeticamente pelo nome dos responsáveis(critério de ordenação da lista doing);

- A oitava opção permite terminar uma tarefa, ou seja, mudar uma tarefa da lista doing para a lista done. É pedido o id da tarefa ao utilizador. Procuramos e removemos, se existir, a tarefa da lista doing com a função remover_tarefa, e posteriormente recorremos à função inserir_done para colocar a tarefa no estagio corerspondente. Adicionalmente, retiramos esta tarefa da lista de tarefas da pessoa responsável, utilizando a função remover_tarefa e eliminando o nó devolvido libertamos espaço nas tarefas correspondentes do responsável;

- Na nona opção possibilitamos a reabertura de uma tarefa, ou seja, passar uma tarefa já terminada para a lista das tarefas por fazer, passar de done para to_do. Após o utilizador indicar o id da tarefa que pretende reabrir. Removemos a tarefa da lista done, remover_tarefa, o ponteiro que corresponde ao responsável da tarefa passa para NULL, mudamos a variável estagio da tarefa para 1 para esta ser guardada corretamente nos ficheiros e inserimos este nó na lista to_do, recorrendo ao uso da da função inserir_to_do, reintroduzimos a tarefa na lista to_do garantindo que a lista continua ordenada;

- A décima opção do menu imprime um quadro que mostra detalhadamente as tarefas exitentes assim como o estágio do pipeline em que se encontram

(Este quadro é igual ao que será guardado no ficheiro texto disponível para o utilizador). Para isto, chamamos a função imprimir_apontadores três vezes, uma para cada lista de apontadores, to_do, doing e done, fazendo uma separação clara do estagio em que se encontram as tarefas;

- Quando o utilizador escolhe o numero 11, é-lhe pedido o id de uma pessoa. Usando as funções pesquisa_pessoa e imprime_apontadores, mostramos ao utilizador todas as funções atribuídas á pessoa escolhida.

- Se o utilizador selecionar 12, o programa só chama a função imprimir_tarefas, mostrando a lista completa de tarefas por ordem de criação;

- A última opção serve para terminar o programa, no entanto antes de sair é perguntado ao utilizador se pretende guardar as alterações feitas. Se o utilizador decidir guardar, a informação que estava na lista de tarefas é carregada para o ficheiro "aux_lista.txt" através da função guardar_lista e o quadro de tarefas, ou seja a informação que está em cada lista de apontadores, é guardada num ficheiro de texto escolhido pelo utilizador através da função colocar_ficheiro. Após esta opção o loop que foi iniciado na função main vai terminar e todas as listas ligadas utilizadas vão ser destruídas.