DEPARTAMENTO DE CIÊNCIA DE COMPUTADORES FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DO PORTO	**LABORATÓRIO DE PROGRAMAÇÃO** Projeto 2020/21 **Quadro de *Kanban***
Nota importante: A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante do ensino superior e futuro profissional. Qualquer tentativa de fraude pode levar à reprovação na disciplina tanto do facilitador como do prevaricador, para além de outras consequências previstas na lei.	

Objetivo

Pretende-se com este projeto desenvolver um quadro de *Kanban*.

Competências

1. Escrita de programas em C. Domínio do ambiente de desenvolvimento.
2. Escrita de código modular, corretamente formatado e indentado.
3. Definição de novos tipos de dados.
4. Domínio de estruturas de dados dinâmicas.
5. Utilização de listas ligadas.
6. Utilização de soluções eficientes de ordenamento.
7. Acesso a ficheiros.
8. Proteção de *input* de dados.

Descrição do Problema

Um quadro de Kanban serve para a gestão visual de processos baseados em pipelines[1], sendo que os principais elementos são apresentados na Figura 1.

[1] Detalhes e exemplos estão disponíveis em https://en.wikipedia.org/wiki/Kanban_(development)

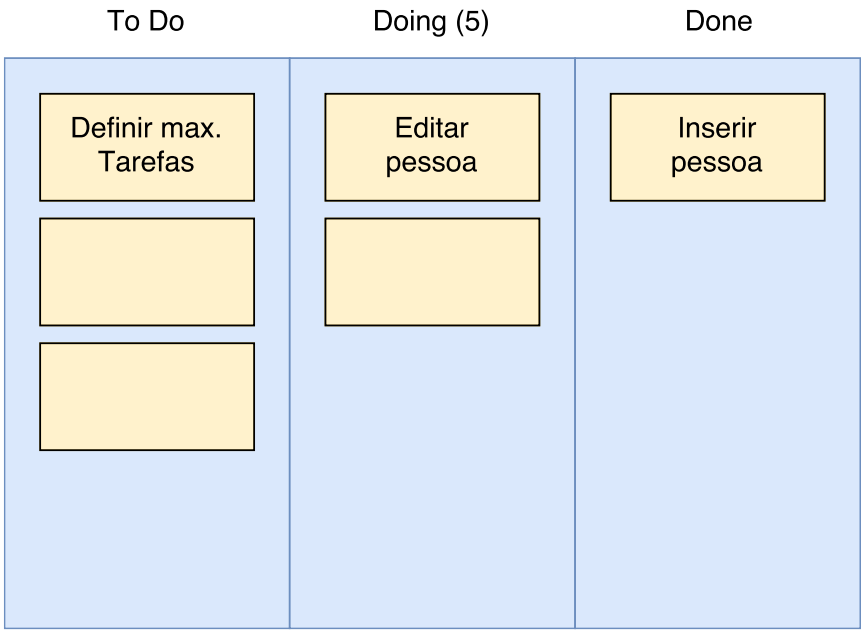

Figura 1. Representação visual do quadro a implementar (exemplo).

Existe uma coluna por cada fase do pipeline. As tarefas são representadas por cartões que vão sendo deslocados pela pipeline até à conclusão. A coluna "Doing" tem um número máximo de cartões que pode conter em cada instante (entre parênteses). Os cartões apenas se podem deslocar uma coluna de cada vez. Cada cartão tem sempre um identificador único, uma prioridade, uma data de criação e uma descrição textual da tarefa. Poderá também ter uma pessoa atribuída, um prazo máximo de conclusão (para as tarefas a realizar ou em curso) e uma data efetiva de conclusão (para as tarefas concluídas), dependendo das fases do pipeline.

O projeto deve permitir as seguintes ações:

1. Inserir uma nova tarefa na lista "To Do", definindo a prioridade para cada tarefa nova (de 1 a 10, sendo 10 a prioridade mais elevada);
2. Mover cartões de "To Do" para "Doing" e vice-versa. Quando se move uma tarefa para "Doing" é necessário atribuí-la a uma pessoa (nome) específica, sendo também necessário indicar um prazo;
3. Alterar a pessoa responsável por um cartão em "Doing";
4. Fechar tarefa: terminar cartão enviando-o para "Done";
5. Reabrir tarefa: enviar cartões de "Done" para "To Do";
6. Visualizar o quadro, sendo que esta tarefa envolve a visualização das três fases. A visualização das tarefas em "To Do" deve ser ordenada por prioridade primeiro e em segundo lugar pela data de criação. As tarefas em "Doing" deverão ser ordenadas pelo nome da pessoa, enquanto que as tarefas em "Done" deverão ser ordenadas por data de conclusão.
7. Visualizar todas as tarefas de uma pessoa. Deve separar a visualização de acordo com a fase do pipeline.
8. Visualizar todas as tarefas ordenadas por data de criação.

A interação com o utilizador deverá ser realizada através de uma consola em modo de texto para permitir a entrada de dados e a apresentação de resultados. Devem implementar as devidas proteções de *input*, para evitar potenciais falhas na interação com o programa. *O*s resultados (ex.: listagens e pesquisas) deverão, também, poder ser armazenados em ficheiros de texto (a pedido do utilizador).

Implementação

A aplicação deve ser implementada na linguagem C e deverá ser baseada em listas ligadas. Mais concretamente, deverá usar listas ligadas em memória para manter os dados das tarefas e das pessoas. As relações entre estas deverão depois ser mantidas com novas listas ligadas de apontadores. Para a visualização das listas ordenadas segundo diferentes critérios, a utilização de estruturas de dados auxiliares com apontadores para os registos reais em vez de estruturas com os próprios registos (para evitar duplicação em memória) será valorizada.

Algumas destas listas deverão ser ordenadas, por exemplo, a das tarefas dentro de cada fase, de acordo com a listagem acima, de forma a permitir uma visualização rápida. Note que deverá ter o cuidado de manter estas listas num estado consistente, mesmo quando apagar dados do programa.

Utilize ficheiros para armazenar os dados sempre que sair da aplicação, para os recuperar, sempre que reentrar, e para os atualizar, sempre que se justifique, durante a execução do programa.

Composição dos grupos

O trabalho deverá ser realizado por grupos de 2 ou 3 elementos pertencentes a alunos da mesma turma.

Material a entregar

Cada grupo deve entregar obrigatoriamente:

1. Upload de zipFile no Moodle com:
   - Todo o código fonte (.c e .h).
   - Ficheiros de dados para teste.
   - Relatório (em formato pdf).
2. O relatório deve incluir os seguintes aspetos:
   - Esquema com as estruturas de dados utilizadas e como se relacionam entre si.
   - Estrutura geral do programa.
   - Estrutura dos ficheiros (que dados são armazenados em cada ficheiro e como).
   - Breve explicação de como o programa se executa.

Entregas com atraso serão penalizadas de acordo com a seguinte fórmula:

$$5\% * 2^{MD-1}$$

Em que MD significa cada meio dia de atraso. Como exemplo, um trabalho entregue na tarde seguinte ao dia de entrega terá **1 dia de atraso**. Assim, se fosse avaliado com **16**, teria **14.4** (10% de penalização).

Defesa final do trabalho

O trabalho deverá ser defendido através de uma prova oral com todos os elementos do grupo. Os estudantes que não comparecerem à defesa do trabalho terão a classificação de **zero** valores no projeto.

A nota do projeto será atribuída em função da avaliação do trabalho entregue e da sua defesa.

Apesar do trabalho ser essencialmente um trabalho de grupo as notas são individuais, podendo ser atribuídas notas diferentes a cada elemento do grupo, sempre que tal se justifique.

As defesas decorrerão após a entrega dos trabalhos, em data a anunciar. Será disponibilizado um mapa de defesas onde os grupos se deverão inscrever.