Trabalho Prático

As regras que regem o trabalho prático encontram-se descritas na ficha de unidade curricular, chamando-se a atenção para a sua leitura. É importante ter em conta que existem várias condicionantes e incógnitas ditadas pela pandemia COVID-19 e, por conseguinte, existem pormenores relativos a apresentação e defesa do trabalho que podem vir a ser definidos ou alterados posteriormente.

Sistema de Gestão do Espaço Aéreo

O trabalho prático consiste na implementação de vários programas que, no seu conjunto, simulam um sistema de gestão de aeroportos, espaço aéreo, passageiros e aviões que nele circulam. No âmbito deste trabalho, todos os intervenientes serão **processos** a correr **na mesma máquina** e não é necessária comunicação em rede – apenas **comunicação entre processos**. Os programas envolvidos terão interfaces consola ou gráfica, conforme o caso, sendo todos manipulados pelo mesmo utilizador do sistema operativo. Este utilizador simula, através da invocação dos diversos programas, a existência dos diversos elementos do sistema de gestão de espaço aéreo.

Neste enunciado, "sistema" referir-se-á ao sistema de gestão em causa. Sempre que for necessário referir o sistema operativo, serão usadas estas duas palavras explicitamente.

1. Elementos do sistema e lógica geral

O sistema tem os seguintes elementos, aos quais correspondem os programas envolvidos:

- controlador aéreo - programa ***control*** (apenas uma instância em execução),
- avião - programa ***aviao*** (podem existir várias instâncias em execução),
- passageiro - programa ***passag*** (podem existir várias instâncias em execução).

Cada programa desempenha apenas um e um só papel relativo ao elemento que representa. Apenas existirá um controlador aéreo em execução, mas dos restantes tipos podem existir várias instâncias em simultâneo.

Lógica e funcionamento geral

O controlador aéreo é o ponto central do sistema, conhecendo e interagindo com os aviões e os passageiros. Gere o espaço aéreo, que é representado por um "mapa" – um plano cartesiano de duas dimensões (coordenadas *x* e *y*) de dimensão sempre 1000x1000 (a unidade usada não é relevante – pode ser designada por "posição"). O controlador não ocupa nenhuma posição no mapa. Existem vários aeroportos, que estão posicionados no plano em determinadas coordenadas e que são geridos pelo controlador aéreo. Existirão vários aviões que poderão estar em movimento (alteram as suas posições no plano ao longo do tempo, estando sempre em coordenadas *x* e *y* inteiras). Não existem outros elementos nem obstáculos no plano ("mapa"). Quando estão em voo, dois aviões não podem ocupar a mesma posição no plano ao mesmo tempo. Considera-se que os aviões apenas ocupam espaço no plano quando estão em voo. Quando estão parados num aeroporto estão na mesma posição que esse aeroporto, e nesse caso (e apenas nesse) podem ocupar a mesma posição que outros aviões.

2. Utilização e funcionalidade detalhada

O programa ***control*** (controlador aéreo) será o primeiro programa lançado. Este programa permitirá, através da sua interface, adicionar aeroportos (cada um numa determinada posição e com um nome que não se pode repetir). Após o lançamento do ***control***, são lançados os programas que representam os restantes elementos: aviões e passageiros. Cada instância do programa ***aviao*** representa um avião em existência, podendo estar em voo ou parado num aeroporto. Cada instância do programa ***passag*** representa um e só um passageiro, podendo o passageiro estar em trânsito (em voo) ou num aeroporto à espera de vez num avião.

Controlador aéreo

Lançamento: *O programa* ***control*** é lançado pelo utilizador. Não carece de nenhuma indicação em particular, verificando apenas que ainda não estava a correr (se estivesse, a nova instância termina).
Tem conhecimento de todo o espaço aéreo, incluindo a posição dos vários aeroportos, aviões e passageiros (em tempo real), podendo comunicar com estes a qualquer momento.

Funcionalidade principal:

- Controla a informação acerca de aeroportos e aviões que sejam criados no espaço aéreo.
- Permite a criação de aeroportos, mediante indicação pela interface com o utilizador do nome do aeroporto e das suas coordenadas. Não deverá existir nenhum outro aeroporto num raio de 10 “posições” nem nenhum outro aeroporto com esse nome. Não existe o conceito de “remover aeroportos”.
- Controla a informação acerca dos aviões que levantam e aterram.
- Recebe (vê) por parte dos aviões, as atualizações das suas posições (gere a ocupação do espaço aéreo).
    - Pode assumir que existe um número máximo tanto de aeroportos como de aviões. Estas quantidades deverão estar definidas no ***Registry***. Quando os valores máximos são atingidos, os novos aviões que se tentem ligar ao sistema serão ignorados até que este tenha espaço para eles.
- Aceita os novos passageiros que se ligam através do programa ***passag***.

Interface com utilizador: interface gráfica *Win32* que apresenta todo o espaço aéreo e os seus elementos. Os aeroportos e os aviões são representados graficamente de forma distinta. Esta informação estará permanentemente visível e sempre atualizada.

- **Importante**: Numa versão inicial (Meta 1) será aceite uma versão simplificada da interface segundo o paradigma consola. Não é para desenhar uma matriz, mas sim para apresentar mensagens de texto tais como “avião ID na posição x,y”.

A interface deve permitir receber os seguintes comandos:

- Encerrar todo o sistema (todas as aplicações são notificadas).
- Criar novos aeroportos.
- Suspender/ativar a aceitação de novos aviões por parte dos utilizadores.
- Listar todos os aeroportos, aviões e passageiros existentes no sistema (com os detalhes de cada um, por exemplo, destino, no caso dos passageiros).

Aviões

Lançamento: cada instância do programa ***aviao*** representa um avião distinto. Este programa é lançado explicitamente pelo utilizador e efetuará muitas viagens. São indicados por linha de comandos: a capacidade máxima (lotação), a velocidade em "posições" por segundo e o aeroporto inicial em que se encontra.

Funcionamento e interface com utilizador: A interface é do tipo consola, seguindo a lógica de comandos (não são menus). O utilizador atua como piloto, dando instruções explícitas para:

- Definir o próximo destino (depois de iniciada a viagem este não pode ser alterado).
- Embarcar passageiros (todos os que estiverem nesse aeroporto para esse destino) e que caibam no avião.
- Iniciar viagem (pode iniciar viagem sem embarcar ninguém).
- Quando em viagem, o piloto não pode fazer nada a não ser terminar o programa. A viagem decorre em piloto automático até ao aeroporto destino. Não é necessário considerar aspetos irrelevantes para o trabalho tais como combustível.
- O piloto pode terminar o programa a qualquer altura. Se o fizer a meio do voo, considera-se que houve um acidente e os passageiros perdem-se. Se o fizer num aeroporto, considera-se que o piloto se reformou. Em ambos os casos, o avião deixa de existir no sistema.

Funcionalidade principal:

- Interage com o controlador aéreo (***control***) conforme for necessário.
- Movimenta-se no espaço aéreo, partindo sempre de um aeroporto em direção a outro. A lógica desta movimentação não é aqui apresentada, já que será fornecida uma DLL com a sua implementação feita. Se a posição estiver ocupada, o avião deverá lidar com a situação de forma coerente (desvio/caminho alternativo, aguardar, outra estratégia).
- Quando chega a um novo aeroporto, deve comunicar a sua chegada ao controlador aéreo.

Outros aspetos: O controlador aéreo considerará que um avião que não dá sinal de vida durante 3 segundos deixou de existir, quer esteja em voo ou parado no aeroporto. O avião é identificado perante o controlador aéreo através do seu ID de processo. Um avião (***aviao***) comunica com o controlador aéreo (***control***), mas nunca comunica diretamente com passageiros (***passag***).

Passageiros

Lançamento e funcionamento*: O programa* ***passag*** é lançado pelo utilizador. Cada instância representa um novo passageiro, que pretende ir de um aeroporto origem para outro aeroporto destino. É indicado por argumento da linha de comandos o aeroporto de origem, o aeroporto de destino, o nome do passageiro, e, opcionalmente, o tempo (em segundos) que fica a aguardar até embarcar. Os nomes dos aeroportos origem e destino devem existir no sistema, caso contrário o programa termina de imediato.

O passageiro é atribuído ao aeroporto origem, ficando a aguardar que exista um avião disponível para o aeroporto destino. Quando tal avião existir, o passageiro embarca automaticamente e, ao chegar ao aeroporto destino, desembarca e o programa termina. Caso tenha sido indicado um tempo de espera máximo, o passageiro desiste automaticamente de viajar se o tempo indicado passar e não for atribuído a nenhum avião naquele intervalo.

Interação com utilizador: O utilizador é automaticamente informado de quando embarca, da posição em que está quando está em voo, e quando chega. O utilizador pode sempre interagir com esta aplicação para a terminar. Se o fizer, considera-se que o passageiro deixou de existir. Tem uma interface do tipo consola. A interação com o utilizador deve ser orientada a comandos escritos e não a menus. Deve adicionar os comandos que considerar necessários para a utilização do programa tendo em vista a funcionalidade requerida.

Funcionalidades principais: interage com o controlador aéreo e com o utilizador. É importante referir que este programa apenas comunica com o controlador (***control***), não podendo existir comunicação com o avião.

3. Formas de comunicação entre as aplicações

A seguinte descrição refere-se apenas à comunicação entre processos. Se na situação *S* o enunciado refere que deve ser usado o mecanismo de comunicação *M*, então tem mesmo que usar o mecanismo *M* nessa situação *S***.**

- A comunicação entre os passageiros (**passag**) e o controlador aéreo (**control**) é feita exclusivamente por *named pipes*, em ambas as direções.
    - **Importante**: Os passageiros (**passag**) apenas podem comunicar diretamente com o controlador aéreo (**control**).
- A comunicação entre os aviões e o controlador aéreo é feita por memória partilhada. A verificação de ocupação e atualização de posições é feita de forma direta na memória partilhada. Toda a restante comunicação do avião para o controlador é feita através do paradigma de produtor/consumidor (*buffer* circular).

Qualquer fluxo de informação no sistema tem que respeitar estas restrições. Por exemplo, no cenário em que um passageiro tenha de receber uma informação por parte de um avião (e.g., alteração de posição), esta é enviada do avião ao controlador, que depois a encaminhará ao passageiro. Qualquer outro fluxo de informação não especificado no enunciado fica ao critério do aluno, desde que realmente faça sentido.

A identificação e correta aplicação de mecanismos de notificação assíncrona e de sincronização fica a cargo dos alunos considerando os cenários em que são necessários, seguindo a arquitetura e implementação do trabalho. A não aplicação ou o uso incorreto destes mecanismos causa penalizações na avaliação.

Os mecanismos de comunicação e de sincronização devem constar num diagrama claro e inequívoco a incluir no relatório que acompanhará a entrega do trabalho.

4. Aspetos em aberto

Os seguintes aspetos devem ser definidos pelo aluno:

- Texto dos comandos escritos.
- Pormenores gráficos de visualização.
- Formato das mensagens trocadas entre as aplicações.
- Detalhes do modelo de dados para o espaço aéreo.
- Mecanismos de sincronização: quais e onde são necessários.
- Outros aspetos não previstos ou não explicitamente descritos.

Devem ser tomadas decisões autónomas e lógicas quanto a estes aspetos, e que não desvirtuem o sistema pretendido nem evitem os conteúdos que se pretendem ver aplicados. O sistema resultante deve ter uma forma de utilização lógica.

Ao nível do modelo de dados que representa o espaço aéreo, é apresentada a seguinte sugestão: deve ser simplificado ao máximo. Não será dada nenhuma valorização adicional ao uso de estruturas de dados dinâmicas onde soluções mais simples seriam suficientes. A indicação de quantidades máximas referidas no enunciado (número máximo de aviões, de aeroportos, etc.) deve ser aproveitada na simplificação das estruturas de dados.

5. Detalhes adicionais acerca dos requisitos

Ao desenvolver o sistema deve também ter em consideração os seguintes requisitos:

- A interface gráfica da aplicação ***control*** não deve cintilar (deve utilizar a técnica de *double buffering*, abordada nas aulas) e terá o seguinte comportamento adicional:
    - *Mouseover* do rato em cima de um avião em voo mostra informação adicional, nomeadamente o seu ID, o aeroporto de origem, destino e o número de passageiros a bordo.
    - *Click* do rato em cima de um aeroporto mostra informação adicional, como o seu nome, a quantidade de aviões e passageiros que alberga.
    - Terá um menu e uma *dialog box* para o envio dos comandos desta aplicação, permitindo ao utilizador inserir os parâmetros necessários e visualizar as respostas (quando aplicável).
    - A interface gráfica do programa ***control*** apenas é requerida para a meta final. Até lá será suficiente uma interface simplificada com o paradigma consola.
- O uso de más práticas na implementação será penalizado. Alguns exemplos, não exaustivos, são:
    - Uso de variáveis globais.
    - Não utilização de programação genérica de caracteres Char/Unicode.
    - Má estruturação do código.
    - Mau uso de ficheiros *header* (por exemplo, o típico e errado “geral.h” que é incluído em todo o lado, ou a definição de funções no *header* file, em vez de apenas declarações).

6. Biblioteca dinâmica fornecida

Será fornecida uma DLL, juntamente com os ficheiros *.h* e *.lib*, que tem de ser **obrigatoriamente utilizada** para a movimentação dos aviões. O movimento do avião é sempre feito posição a posição. Se o avião se movimentar N posições por segundo, então, a cada segundo são calculadas essas N posições, deslocando-se sempre da atual para a seguinte na linha reta que une o aeroporto origem ao aeroporto destino.

A funcionalidade fornecida por esta biblioteca será o “cálculo” das novas posições dos aviões, quando estão em movimento, sendo dadas as coordenadas atuais e as coordenadas finais, e sendo produzidas as coordenadas da posição “seguinte” na reta que une a posição atual do avião à posição do aeroporto destino.

A DLL tem as seguintes características:

- Não considera velocidade de movimento (terá que ser o avião a chamar mais/menos vezes por segundo a função que obtém a próxima posição).
- Não tem conhecimento acerca do sistema. Não conhece nomes de aeroportos. Apenas trabalha com posições num segmento de linha reta (que une um aeroporto a outro). Assuma que a DLL fornece sempre

a próxima posição e deverá confirmar se esta posição está livre para poder alterar a posição atual do avião que invocou a função.

- Possui apenas a seguinte função: **int move(int cur_x, int cur_y, int final_dest_x, int final_dest_y, int * next_x, int * next_y);**
- A função permite indicar ao utilizador a posição seguinte de um avião (**next_x, next_y**), partindo apenas da sua posição atual (**cur_x, cur_Y**) e da posição final para onde vai (que, neste caso, será a posição de um aeroporto: **final_dest_x, final_dest_y**). A função devolve 0 se o avião chegou à posição final, 1 se fez uma movimentação correta a caminho do destino, e 2 caso ocorra algum erro.

A ligação com a DLL pode ser feita de forma implícita ou explícita, ficando a decisão a cargo dos alunos.

7. Algumas chamadas de atenção

- Não coloque ponteiros em memória partilhada (pelas razões explicadas nas aulas teóricas). Isto abrange ponteiros seus e também objetos de biblioteca que contenham internamente ponteiros (por exemplo, objetos C++ STL String, Vector, etc.).
- Pode implementar o trabalho em C++, se assim quiser, desde que não oculte o API do Windows com *frameworks* de terceiros. Se o objetivo for utilizar polimorfismo, neste trabalho não irá ter grande necessidade. Tenha também em atenção que irá ter de usar uma DLL fornecida pelos docentes que estará feita em C e, portanto, deve garantir a interoperabilidade entre o seu código e essa DLL.
- Se utilizar repositórios *git*, terá que garantir que são **privados**. Se usar um repositório público que depois seja copiado por terceiros, será considerado culpado de partilha indevida de código e terá o trabalho anulado.
- A deteção de situações de plágio leva a uma atribuição direta de 0 valores na nota do trabalho aos alunos de todos os grupos envolvidos, podendo ainda os mesmos estar sujeitos a processos disciplinares.
- Todo o código apresentado poderá ser questionado na defesa e os alunos têm obrigatoriamente que o saber explicar. **A ausência de explicação coerente é entendida como possível fraude ou como falta de conhecimento, que naturalmente se reflete na nota.**

8. Regras e prazos

- **O trabalho é feito em grupos de 2 alunos**. Não são aceites grupos com 3 ou mais alunos. O trabalho foi desenhado e dimensionado para ser realizado em grupo de 2 alunos, sendo que, como exceção, podem eventualmente ser aceites trabalhos individuais. Todavia é aconselhada e encorajada a constituição de grupos de 2 alunos. A avaliação será realizada com os mesmos critérios independentemente de os grupos terem 2 ou, como exceção, 1 aluno.
- O trabalho é composto por duas entregas: a Meta 1 e Meta Final.
- Nas entregas deve enviar o projeto do seu trabalho comprimido no formato **zip**, contendo apenas os ficheiros de controlo do projeto e de código fonte, ou seja, sem os binários (os ficheiros de debug das diretorias *bin* e *obj*) e sem ficheiros *precompiled headers*. Se o seu arquivo zip tiver mais de 10 Mb, poderá não conseguir fazer a submissão no Moodle, sendo esta responsabilidade dos alunos, que devem verificar esta questão atempadamente.

Meta 1 – 16 de Maio

O material a entregar deverá ser o projeto em Visual Studio, compilável e sem erros de execução/compilação, dos seguintes programas com as respetivas funcionalidades:

- Controlador aéreo (**control**) – Nesta meta com uma interface do tipo consola, cria o(s) mecanismo(s) de comunicação e sincronização com os programas que representam os aviões. Atende até um máximo de aviões definido no Registry. Comunica com os aviões em ambos os sentidos. Cria e gere as estruturas de dados a usar pelo sistema.
- Avião (**aviao**) – Desloca-se, evitando colisões em voo com outros aviões, usa a biblioteca DLL fornecida pelos docentes para saber qual será a próxima posição a ocupar na sua trajetória.

É também necessário entregar um relatório sucinto com a descrição dos mecanismos de comunicação e sincronização implementados, assim como as estruturas de dados definidas. Deverão ainda incluir um pequeno manual de utilização de como usar o sistema, para já composto apenas de dois programas.

Meta Final – 13 Junho

O material a entregar deverá ser:

- O trabalho completo, com os programas que constituem o sistema totalmente implementados.
- Um relatório completo, com o máximo de 10 páginas, a explicar os pontos essenciais da implementação de cada um dos programas envolvidos, as estruturas de dados definidas e a sua utilidade, e todos os aspetos que não estejam explicitamente mencionados no enunciado e que tenham sido decididos pelos alunos, e também o diagrama com os mecanismos de comunicação e de sincronização. O relatório deve contemplar uma tabela onde indique quais os requisitos implementados, no formato:

ID	Descrição funcionalidade / requisito	Estado
...	...	implementado / não implementado (neste caso, indicar as razões)

9. Avaliação

O trabalho vale **8 valores**. A nota será atribuída com base nas funcionalidades implementadas, na qualidade das soluções adotadas, na forma como são explicadas no relatório e na qualidade da defesa.

A avaliação ocorre, essencialmente, na Meta Final. Na Meta 1 avalia-se o cumprimento dos objetivos estabelecidos através de uma apresentação obrigatória. Tal como descrito na ficha da unidade curricular, a avaliação será feita da seguinte forma:

- Nota da Meta 1 = fator multiplicativo entre 0,8 e 1,0.
    - A não comparência na apresentação obrigatória da Meta 1 fará com que a mesma não seja considerada. Ou seja, o fator multiplicativo obtido será o mesmo que seria obtido se não entregasse a meta (0,8).
- Nota da Meta Final = valor entre 0 e 100 que representa a funcionalidade e qualidade do trabalho, e a qualidade da defesa.
- Nota final do trabalho = nota obtida na Meta Final * fator multiplicativo obtido na Meta 1.

- A Meta 1 não tem qualquer valor sem a Meta Final. Ou seja, grupos que não entregarem a Meta Final ou que não compareçam na sua defesa terão uma nota final de 0 valores.
- A falta da Meta 1 não impede a entrega da Meta Final. Apenas prejudica a nota final, já que a nota mínima da Meta 1 será automaticamente assumida (0,8).

O trabalho está planeado para ser feito ao longo do semestre e a entrega do trabalho prático é única para todo o ano letivo. Não existirá trabalho prático na época especial ou noutras épocas extraordinárias que os alunos possam ter acesso, sendo o exame sempre cotado para 12 valores.