isec Engenharia	**Sistemas Operativos II** Engenharia Informática – DEIS	**2021/22**

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 à apresentação e defesa do trabalho que podem vir a ser definidos ou alterados posteriormente.

Jogo dos Tubos

O trabalho prático consiste na implementação do "Jogo dos Tubos" (https://classicreload.com/win3x-pipe-dream-1991.html) em versão multiutilizador, com um máximo de dois utilizadores. No âmbito deste trabalho, todos os intervenientes serão **processos** a correr na mesma máquina ou eventualmente em duas máquinas distintas. Os programas envolvidos terão interfaces consola ou gráfica.

O jogo consiste em conduzir um caudal de água de um ponto de origem para um ponto de destino de uma área retangular no monitor. Para esse efeito, o jogador terá de utilizar as peças/tubos mais adequados ao percurso pretendido. A área de jogo está dividida em quadrículas onde o jogador pode colocar cada uma das peças disponíveis. Existem três tipos de peças: tubo horizontal (**━**), tubo vertical (**┃**) e curva a 90º (**┏**). Esta última é passível de ser colocada de quatro formas distintas de acordo com a rotação aplicada (**┏**, **┓**, **┛** e **┗**). A água começa a fluir após um determinado período de tempo, a contar desde o início do jogo, e deve manter-se em movimento até chegar ao destino, ou atingir a última peça que foi colocada pelo jogador. Neste último caso o jogador perde o jogo. Se a peça não estiver posicionada corretamente e for atingida pela água o jogo termina e o jogador perde o jogo.

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

1. Elementos do jogo e lógica geral

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

- Servidor de controlo do jogo – programa **servidor** (apenas uma instância em execução),
- Monitor para acompanhar o jogo – programa **monitor** (normalmente apenas uma instância),
- Cliente para jogar o jogo – programa **cliente** (podem existir até duas instâncias em execução).

Cada programa desempenha apenas um e um só papel. Apenas existirá um **servidor** e uma ou mais instâncias do **monitor** em execução na mesma máquina. Podem existir até duas instâncias em simultâneo do **cliente** na mesma máquina ou em máquinas distintas.

Lógica e funcionamento geral

O **servidor** é o ponto central do sistema, controla todo o jogo e interage com os restantes elementos: clientes e monitor. Gere a área de jogo, que é representada por um mapa bidimensional (coordenadas *x* e *y*) de dimensão pré-definida, com o máximo de 20 x 20 células. Nesta área existem um ponto de origem da água e um ponto de destino que ocupam células limítrofes, normalmente opostas. Em cada uma das células remanescentes pode ser colocada uma peça, de acordo com a jogada efetuada pelo jogador através da aplicação cliente.

O **monitor** é uma aplicação que pode ser executada na mesma máquina onde está a ser executado o servidor e que permite visualizar o estado do jogo de um ou de ambos os jogadores. Permite também gerir alguns aspetos do jogo.

O **cliente** é a aplicação utilizada pelo jogador e que interage com o servidor para indicar as suas intenções e observar o estado do seu jogo.

2. Utilização e funcionalidade detalhada

O programa **servidor** será o primeiro programa lançado. Apenas pode existir uma instância do servidor e isso deve ser acautelado. Os programas **monitor** e **cliente** só podem ser executados caso o servidor já se encontre em funcionamento.

Servidor

Lançamento

O programa **servidor** é lançado pelo utilizador. Deve assegurar-se que ainda não estava a correr (se estiver, a nova instância termina).

Tem conhecimento dos mapas de jogo dos jogadores e gere toda a informação relativa ao jogo. Disponibiliza informação ao programa monitor e interage com os clientes sempre que tal seja necessário.

A aplicação deve permitir a especificação, através da linha de comandos, das dimensões da área de jogo e do período de tempo após o qual a água começa a fluir. Esta informação deve ser armazenada no *Registry* e utilizada em execuções subsequentes do programa, sempre a mesma não seja especificada através da linha de comandos.

Funcionalidade principal

- Controla a informação do mapa de jogo.
    - As dimensões do mapa de jogo e o tempo que a água demora a fluir são passados através da linha de comandos ou encontram-se definidos no ***Registry***.
- Determina de forma aleatória a posição da origem e destino da água. Ambas devem estar localizadas numa das células junto ao bordo do mapa e em quadrantes diagonalmente opostos.
- Recebe comandos do **monitor** e desencadeia as ações necessárias (“fechar a torneira”, etc.).
- Aceita os jogadores que se ligam através do programa **cliente**.
- Recebe por parte dos clientes, as jogadas que pretendem efetuar (colocação de uma peça numa determinada posição) e mantém atualizados os mapas de jogo.

Interface com utilizador

Interface segundo o paradigma consola, seguindo a lógica de comandos (não são menus). A interface deve permitir receber os seguintes comandos:

- Listar os jogadores e respetiva pontuação.
- Suspender e retomar o jogo.
- Encerrar todo o sistema (todas as aplicações são notificadas).

Outros aspetos

Se um jogador não der sinal de vida durante 15 segundos, perde automaticamente o jogo e deixa de existir.

Monitor

Lançamento

O programa **monitor** é lançado explicitamente pelo utilizador. Normalmente apenas existe um monitor, mas nada impede que existam mais instâncias do mesmo.

Funcionamento e interface com utilizador

Interface segundo o paradigma consola que apresenta os mapas dos jogadores que se encontram a jogar em determinado momento. Esta informação estará permanentemente visível e será atualizada em tempo real.

A interface deve permitir receber os seguintes comandos:

- Parar a água durante um período de tempo (especificado em segundos).
- Inserir blocos que representam paredes intransponíveis no mapa.
- Ativar/desativar o modo aleatório para a sequência de peças/tubos (a peça que surge após cada clique pode seguir uma sequência predefinida: ━, ┃, ┏, ┓, ┛ e ┗; ou ser aleatória).

Funcionalidades principais

- Interagir com o **servidor** conforme for necessário para mostrar o estado do jogo de todos os jogadores.
- Modificar o comportamento do jogo através do envio de comandos ao **servidor**.

Cliente

Lançamento e funcionamento

O programa **cliente** é lançado pelo utilizador sempre que pretende jogar. Cada instância representa um novo jogador.

Existem duas modalidades de jogo: individual e competição. No primeiro caso o jogador compete contra o tempo e à medida que vai avançando de nível, o jogo torna-se mais difícil (ex.: a velocidade de água aumenta a cada nível, etc.). No segundo modo de jogo compete contra o tempo e contra um adversário.

Funcionamento e interface com utilizador

Interface gráfica *Win32* que apresenta o mapa do jogo e toda a informação. Esta informação estará permanentemente visível e será atualizada em tempo real.

O programa começa por solicitar ao utilizador o nome e o tipo de jogo que pretende. Tratando-se da modalidade de jogo individual, o jogo começa de imediato. Na modalidade de competição, terá de aguardar a chegada de um adversário para se dar início ao jogo.

No decurso do jogo o utilizador poderá clicar em cima das células do mapa para aí colocar uma peça/tubo. A cada clique do rato na mesma célula a peça muda, percorrendo todas as peças disponíveis.

Funcionalidades principais

- Interage com o **servidor** e com o utilizador.
- Este programa apenas comunica com o **servidor**, não existindo comunicação direta com o **monitor**.

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 o **servidor** e os **clientes** é feita exclusivamente por *named pipes*, em ambas as direções.
    - **Importante**: Os **clientes** apenas podem comunicar diretamente com o **servidor**.
- A comunicação entre o **servidor** e o **monitor** é feita exclusivamente por memória partilhada. A verificação do conteúdo das células e a sua atualização é feita de forma direta na memória partilhada. Toda a restante comunicação do **monitor** para o **servidor** 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 o **monitor** insere barreiras físicas no mapa de um dos jogadores, esta informação é enviada ao **servidor**, que depois a encaminhará para o **cliente**. 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 mapa do jogo.
- 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.

O modelo de dados que representa o mapa de jogo deve ser simplificado ao máximo. Não será dada nenhuma valorização adicional ao uso de estruturas de dados mais complexas onde soluções mais simples seriam suficientes. A indicação de quantidades máximas referidas no enunciado 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 **cliente** não deve cintilar. Deve utilizar a técnica de *double buffering*, abordada nas aulas.
- A interface do **monitor** terá o seguinte comportamento:
    - Mostrar a informação constante do mapa dos jogadores em tempo real.
    - Desencadear ações que alteram o funcionamento do jogo e comunicá-las ao **servidor**.
- A interface do **cliente** terá o seguinte comportamento:
    - Clique com o botão esquerdo do rato em cima de uma célula onde ainda não passou a água permite mudar a peça aí existente. Cada clique do rato nessa célula muda a peça/tubo de forma circular (horizontal, vertical e curva a 90º com as quatro rotações possíveis). Um clique com o botão direito do rato limpa a célula.
    - *Mouseover* sobre a célula onde a água se encontra atualmente suspende temporariamente o seu fluxo, levando a que o jogo entre num modo de pausa até que o cursor saia da célula em causa. Este comportamento é uma ajuda ao jogador, e por isso só acontece no máximo 3 vezes por jogo. Para evitar que este modo de pausa seja ativado acidentalmente, o *mouseover* só é ativado 2 segundos após o cursor se encontrar sobre a célula.
    - O menu principal da janela deve permitir alternar entre 2 conjuntos de *bitmaps* utilizados para representar os vários elementos do jogo. Fica ao critério dos alunos escolher os *bitmaps* que constituem os vários elementos do jogo, não podendo existir *bitmaps* repetidos entre os 2 conjuntos.
- 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. 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 terá grande necessidade.
- 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.**

7. 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 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** (máximo de 10Mb), 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*.

Meta 1 – 15 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:

- **Servidor** – Cria o(s) mecanismo(s) de comunicação e sincronização com o programa **monitor**. Utiliza a informação que se encontra definida no *Registry*. Utilizando um mapa com uma solução para o problema (ligação entre o ponto de origem e destino), implementa o movimento da água nos tubos. Cria e gere as estruturas de dados a usar pelo sistema.
- **Monitor** – Com uma interface do tipo consola permita visualizar o(s) mapa(s) de jogo em tempo real. Interage com o utilizador e permite a receção e envio de informação de/para o **servidor**.

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 – 19 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)

8. 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.