Instituto Superior de Engenharia de Coimbra
Departamento de Engenharia Informática e de Sistemas
Engenharia Informática

Programação Orientada a Objectos - 2021/2022

Trabalho Prático

Regras gerais do trabalho

O trabalho prático de POO é constituído por um programa em C++. Este programa deve:

- Seguir os princípios e práticas de orientação a objetos;
- Ser feito em C++, usando corretamente a semântica desta linguagem;
- Usar as classes/bibliotecas usadas nas aulas, onde apropriado, ou outras da biblioteca *standard* C++. Não devem ser usadas outras bibliotecas sem o consentimento prévio dos docentes;
- Deve concretizar as funcionalidades do tema referidas no enunciado.

A interface com o utilizador é feita segundo o conceito de consola não-gráfica, e não requer nenhuma manipulação nem de cores nem de posicionamento de elementos no ecrã.

São aceites os trabalhos que usem a *framework* Qt para construção de interface gráfica. Esta *framework* é feita em C++ e é *cross-platform*, estando alinhada com os objetivos gerais da disciplina (C++). O uso desta *framework* é completamente opcional, mas será recompensado com até 12.5% (mediante a quantidade de uso desta *framework* que é feita), e sem ultrapassar os 100% do trabalho. Significa que pode esta valorização ser usada para compensar outras partes em falta. Todo o trabalho de exploração desta *framework* (que é fácil) fica a cargo dos interessados em a usar.

Este enunciado deixa diversos aspetos do tema e funcionalidades em aberto. Este age como uma situação representativa de cenários de indústria em que existem pormenores que devem ser deduzidos e completados por quem executa o trabalho. Estes aspetos devem ser abordados segundo o bom senso e sem remover dimensão ou complexidade ao enunciado. Não será aceitável não concretizar algo cuja necessidade é obvia apenas porque não foi explicitamente pedido no enunciado. Estes aspetos em aberto incluem:

- Funcionalidades cuja necessidade se torna óbvia por outras que são mencionadas;
- Aspetos de implementação que envolvam o uso correto dos mecanismos e semântica de C++;
- Aspetos de organização segundos as boas práticas de programação e do paradigma de orientação a objetos.

O trabalho deve ser feito em grupos de 2 alunos, podendo ser alunos de laboratórios diferentes.

O trabalho é entregue em duas metas nas datas 21 de novembro e 16 de janeiro. As entregas serão feitas via moodle, e as indicações detalhadas serão dadas na devida altura. Existem defesas que são obrigatórias e influenciam bastante a nota. As regras adicionais relativas ao trabalho prático encontram-se descritas na ficha da disciplina, cuja consulta é obrigatória antes da colocação de questões quanto a estas regras.

As questões omissas que sejam relevantes serão resolvidas e divulgadas pelos docentes no moodle.

Tema e funcionalidades

1. TEMA GERAL

Pretende-se construir em C++ um simulador/jogo (*single-player*) de construção e desenvolvimento. Será atribuída ao jogador a concessão de uma ilha e o jogador deve desenvolver essa ilha, industrializando-a e construindo todo um complexo fabril.

A geografia da ilha consiste num conjunto de zonas adjacentes umas às outras, havendo vários tipos de zona possíveis. O jogador pode edificar edifícios, que ficarão colocados nas zonas que o jogador escolher (no máximo um edifício por zona). O jogador irá gerir uma equipa de trabalhadores que incluirá diversos tipos de trabalhador. O jogador pode movimentar os trabalhadores na ilha para que estes possam desempenhar as suas tarefas. Ao logo do jogo irão ser usados recursos que são produzidos na ilha. A ação do jogo é controlada por ações que o jogador especifica escrevendo ordens ("comandos").

O jogo decorre em dias, em que em cada dia é permitido ao jogador um determinado número de ações. Existem determinados acontecimentos que ocorrem automaticamente no início do dia e outros que ocorrem no final do dia. Considera-se que as ações especificadas pelo jogador ocorrem a meio do dia (depois dos acontecimentos do início do dia, mas antes dos acontecimentos do fim do dia).

O jogo termina quanto o jogador quiser, ou quando fica impossibilitado de continuar por já ter perdido todos os seus recursos/trabalhadores. A pontuação do jogo é a riqueza acumulada pelo jogador no momento em que o jogo termina.

2. ELEMENTOS DO JOGO

Trabalhadores

O jogador gere uma equipa de trabalhadores. Existem diversos tipos de trabalhadores. Os trabalhadores podem ser contratados, mediante um pagamento único inicial (por cada trabalhador) e que varia consoante o tipo de trabalhador. Os trabalhadores podem achar que as condições são muito duras e simplesmente despedirem-se, desaparecendo da ilha (no entanto, avisam quando o fazem). Cada trabalhador tem um identificador único que é gerado automaticamente e tem o formato *n.d* em que *n* é um número crescente começando em 1 e incrementado a cada novo trabalhador contratado, e *d* é o dia (da simulação) em que foi contratado, sendo o primeiro dia o dia 1. Exemplo: 15.7 é o identificador do 15º trabalhador, o qual foi contratado no 7º dia da simulação. Esta numeração é partilhada por todos os tipos de trabalhadores (exemplo, um lenhador contratado a seguir a um mineiro com identificador 10.d terá a identificação 11.d).

Os trabalhadores estarão sempre numa zona da ilha. A presença de um trabalhador pode afetar o que acontece nessa zona ou no edifício que lá esteja, dependendo do tipo de zona, de trabalhador e de edifício. Um trabalhador pode sempre ser colocado numa zona independentemente do número de trabalhadores já lá existentes, do tipo de trabalhador, do tipo de zona, ou até da utilidade que essa colocação possa trazer.

Operário. Bom para operar fundições e centrais elétricas. Custa 15 € para contratar. Tem 5% de probabilidade por dia de se cansar e ir embora, mas só a partir do 10º dia após ter sido contratado.

Lenhador. Só serve para cortar árvores. Custa 20 €. Trabalha 4 dias e descansa 1 (e depois repete). Nos dias em que descansa, é como se não existisse (o que estiver dependente do seu trabalho não funciona / não produz / não tem efeito). Dado o descanso e o trabalho ao ar livre, em princípio não se despede.

Mineiro. Seve para escavar a terra em minas. Custa 10 €. Dada a vida desgraçada que leva tem uma probabilidade de se ir embora de 10% em cada dia e logo a partir do segundo dia na ilha.

Zonas da ilha

A Ilha é retangular e a sua geografia é um quadriculado em que cada quadrícula é uma zona. O tamanho (número de zonas) da ilha é definido pelo utilizador em *runtime*, podendo ser, no mínimo 3x3 zonas e no máximo 8x16 zonas (8 ***linhas*** e 16 ***colunas***). Toda a ilha deve ser visível no ecrã em simultâneo. O jogador tem necessidade de saber várias informações de cada zona e, por isso, recomenda-se que se ocupe, no ecrã, pelo menos 4x4 caracteres para representar cada zona (note que isto tem a ver com *visualização* e não com *dados*). Existem vários tipos de zona, cada um com características diferentes das restantes. A ilha é definida de forma aleatória da seguinte forma:

- Deve existir pelo menos uma zona de cada tipo
- Deve tentar ter uma distribuição *mais ou menos* equilibrada de tipos de zona na ilha: todos os tipos estarem aproximadamente igualmente representados na ilha.

Todos os trabalhadores são inicialmente colocados numa zona do tipo pasto. Se houver vários pastos na ilha, o pasto inicial para colocar os trabalhadores é decidido de forma aleatória.

O objetivo do jogador será o de industrializar toda a ilha, construindo ou adquirindo diverso equipamento que lhe permita produzir bens para venda ou para uso próprio em construções seguintes. Em última análise, toda a ilha poderá ser coberta de construções, uma por zona. O jogador controlará os destinos de todas as suas construções, da sua equipa, e dos recursos acumulados, sendo esse controlo feito através dos comandos que escreve.

Recursos

Alguns dos recursos podem ser obtidos diretamente a partir da ilha, enquanto outros só podem ser obtidos a partir da transformação de outros recursos. Os recursos, depois de obtidos, podem ser armazenados, transformados em outros recursos, vendidos, ou usados em construções.

Os recursos são:

- Ferro. Pode ser obtido do solo da ilha através de uma mina. Pode ser vendido por 1 € por kg.
- Barra de aço. Pode ser obtido através da transformação de ferro e carvão. 1.5 kg de ferro e 0.5 kg de carvão dão origem a um 1 barra de aço. Pode ser vendido por 2 € por kg.
- Carvão. Pode ser obtido a partir da ilha através de uma mina. Também pode ser obtido pela transformação de madeira em carvão: 1 kg de madeira dá origem a 1 kg de carvão. Pode ser vendido por 1 € por Kg.
- Madeira. Extraída das florestas da ilha. Pode ser vendida por 1 € por kg de madeira
- Vigas de madeira. Pode ser obtida por transformação de madeira numa serração. 2 kg de madeira dão origem a 1 viga. Pode ser vendida por 2 € por viga.
- Eletricidade. Obtida por queima de carvão. Pode ser vendida por 1.5 € por KWh. 1 kg de carvão dá origem a 1 KWh de electriidade

Edifícios

O jogador pode ordenar a construção de edifícios para diversos fins. Cada edifício é especializado numa função específica e está sujeito a determinadas restrições. Os edifícios podem exigir determinadas condições para poderem operar: a existência de um trabalhador, ou estarem junto a um outro determinado tipo de edifício, ou outra condição qualquer, dependendo do tipo de edifício (descritos mais adiante). Independentemente de estarem reunidas as condições exigidas, o edifício só "trabalha" se estiver "ligado", e assim que é construído está inicialmente "desligado". Quer isto dizer que o jogador pode ordenar a cada edifício que funcione ou que esteja inativo.

Mina de ferro. Permite obter ferro. Pode ser construída gastando 10 vigas de madeira, em que cada viga pode ser substituída por 10 €. Produz 2 Kg de ferro por dia. Pode ser melhorada até ao nível 5 em que cada nível aumenta a produção em 1 kg de ferro por dia. Cada nível de melhoramento exige 15 € e 1 viga de madeira (não substituível por €). A mina apenas produz se existir um mineiro na zona em que a mina se encontra. Em cada dia a mina tem 15% de probabilidade de desabar. A mina de ferro armazena até 100 kg de ferro, mais 10 Kg por cada nível adicional. Depois disso, pára de produzir.

Mina de carvão. Permite obter carvão. Pode ser construída gastando 10 vigas de madeira, em que cada viga pode ser substituída por 10 €. Produz 2 Kg de carvão por dia. Pode ser melhorada até ao nível 5 em que cada nível aumenta a produção em 1 kg de carvão por dia. Cada nível de melhoramento exige 10 € e 1 viga de madeira (não substituível por €). A mina apenas produz se existir um mineiro na zona em que a mina se encontra. Em cada dia a mina tem 10% de probabilidade de desabar. A mina de carvão armazena até 100 kg de carvão, mais 10 Kg por cada nível adicional. Depois disso, pára de produzir.

Central elétrica de biomassa: Queima madeira, produzindo carvão e eletricidade. A central elétrica transforma 1kg de madeira em 1 kg de carvão mais 1 KWh de eletricidade por dia, desde que se encontre um operário na zona em que se encontra. A central pode armazenar o carvão produzido até 100 kg de carvão. A eletricidade ficará armazenada numa bateria que se encontre numa zona adjacente, caso contrário perde-se. Para funcionar, a central elétrica tem que estar numa zona adjacente a uma zona do tipo floresta, sendo a madeira obtida a partir da madeira que tenha sido cortada e depositada nessa zona de floresta. Custa 15 €.

Bateria: é um edifício que consiste num enorme bloco de lítio que armazena eletricidade. Tem capacidade de 100 KWh. Custa 10 € e 10 vigas. Adquire automaticamente a energia produzida nas centrais elétricas que estejam colocadas em zonas adjacentes. Pode ser melhorado até ao nível 5 por mais 5 € cada nível

Fundição. Permite obter aço a partir de ferro e carvão. Para funcionar é necessário que a zona em que se encontre seja adjacente a uma zona que tenha uma mina de ferro e a uma mina de carvão ou a uma central elétrica (por causa do carvão). Precisa também de ter um operário na sua zona. Custa 10 €.

Edifício-X. Tem as caraterísticas e comportamento que cada grupo entender. Em princípio não há repetição de edifícios-X entre grupos diferentes, pois isso seria mesmo muito estranho.

Zonas

Existem diversos tipos de zona, sendo a ilha inicialmente definida de forma aleatória, garantindo que existe pelo menos uma de cada tipo e que o número de zonas de cada tipo é aproximadamente o mesmo. Consoante as suas características, as zonas podem afetar a construção e operação dos edifícios que nela se encontram, ou até os próprios trabalhadores que nelas se encontrem. Em cada zona pode haver zero ou 1 edifício, e zero, 1 ou muitos trabalhadores, independentemente do tipo de edifício ou de trabalhador.

Para efeitos de zona adjacente a outra: duas zonas são adjacentes apenas se partilharem uma aresta.

Deserto. São dunas de areia. Permite a construção de qualquer tipo de edifícios. No entanto, as minas sofrem uma redução de 50% de produção por causa dos aluimentos (areia solta).

Pastagem. Aceita qualquer tipo de edifício. Os funcionários que se encontrem aqui nunca pedem a demissão dado que a paz da paisagem reduz o stress.

Floresta. Tem um conjunto inicial de árvores, decidido aleatoriamente entre 20 e 40 árvores. A cada dois dias cresce uma nova árvore, até ao máximo de 100. A floresta produz 1 kg de madeira por cada lenhador que se encontra nessa zona. A madeira cortada fica armazenada ao ar livre sem limite de quantidade. Se forem construídos edifícios numa zona do tipo floresta, morrerá uma árvore por dia e não crescem novas árvores.

Montanha. Aceita qualquer tipo de construção, no entanto o preço de construção é o dobro do normal. As minas aumentam a produção em 100%. As montanhas têm a caraterística interessante de produzirem espontaneamente ferro a 0.1 Kg por dia por cada funcionário que lá se encontre, independentemente do seu tipo. Este ferro extra fica armazenado na própria zona e não tem limite (fica ao ar livre). Dado que andam sempre vergados a apanhar pepitas de ferro, a probabilidade destes trabalhadores pedirem demissão aumenta em mais 5%, e até os lenhadores (se lá estiverem) são afetados.

Pântano. Aceita qualquer tipo de edifício, que têm o custo normal e produzem normalmente. No entanto, passados 10 dias, o edifício afunda-se e desaparece. Os funcionários que lá se encontrem pedem a demissão passados esses mesmos 10 dias (se não for antes pelas suas próprias razões, conforme o seu tipo). Um pântano é realmente pouco interessante, mas existe e é tão obrigatório como os outros.

Zona-X. Tem as caraterísticas e comportamento que cada grupo entender. Zonas-X repetidas em grupos diferentes é o mesmo que pedir para ir ver o que se passa nesses grupos com uma lupa.

3. DECURSO DO JOGO

O jogador não está posicionado em nenhuma parte da ilha. Os seus edifícios e os seus trabalhadores é que se encontram posicionados em zonas.

Um edifício só existe estando posicionado numa zona. Não pode ser movimentado.

Um trabalhador tem que estar posicionado num local da ilha qualquer. No entanto, pode ser movimentado para outra zona. O movimento é instantâneo, ou seja, não há lugar a um trajeto que é percorrido, nem a origem e destino precisam de estar próximas. O trabalhador sai da zona origem e entra imediatamente na zona destino. Todos os trabalhadores estarão inicialmente numa zona do tipo pasto.

O jogo decore em dias. Em cada dia existem 3 momentos importantes:

- Amanhecer: acontecem os efeitos dos vários tipos de zona.
- Meio dia: o jogador pode expressar as suas ordens, por exemplo, mandar construir edifícios, vender recursos, ligar/desligar edifícios, movimentar trabalhadores. O jogador pode emitir o número de ordens que quiser por dia, mas cada trabalhador só pode ser movimentado uma vez por dia.
- Anoitecer. Faz-se a recolha dos recursos obtidos (minados, transformados, etc.), já considerando os trabalhadores nas zonas para onde tenham sido movidos nas ordens do meio-dia, assim como os edifícios que tenham sido construídos nessa altura

4. INTERFACE COM O UTILIZADOR

A interface será em consola modo-texto, aplicando-se o que foi dito no início relativamente ao uso de Qt. Sublinha-se novamente que Qt é completamente opcional e a sua valorização só servirá em caso de falha de outras partes (a nota máxima no trabalho é 100%).

A ilha será representada na sua totalidade no ecrã. A representação será sobre a forma de um quadriculado, em que cada quadrícula é uma zona. A quadrícula ocupará vários espaços (NxM caracteres) de forma a poder indicar o que se passa nessa zona: o seu tipo, que edifício contém, que funcionário nela se encontram (quais e quantos), etc. Para efeitos de identificação nas ordens do jogador, as zonas são identificadas por ***linha, coluna***. A primeira linha é a linha 1 e a primeira coluna é a coluna 1, de cima para baixo, da esquerda para a direita. A representação visual deve separar as zonas umas das outras (usando um caracter qualquer a servir de separador). A figura seguinte mostra uma sugestão de interface.

flr LO 2	pas elec	pan	flr	mnt mnF M
pas	dsr bat O 1	mnt	flr L 1	mnt
znX	mnt mnC M 1	pnt	dsr	mnt M 1
mnt	pas	znX	pas OOOO 5	flr

Sugestão de representação (nota: **pode** usar * (ou outra coisa) em vez dos caracteres ┐┌├┬┴─| etc.)

Pode representar de formas diferentes desta, mas:

- A ilha deve estar toda no ecrã
- É necessário indicar o conteúdo das várias zonas. Utilize as abreviaturas e códigos indicados:

pnt –> Montanha dsr -> Deserto pas -> Pastagem flr -> Floresta pnt -> Pântano znZ -> ZonaX	mnF -> Mina de ferro mnC -> Mina de carvão elec -> Central elétrica bat -> Bateria fun -> Fundição

- 1ª linha: tipo de zona.
- 2ª linha: edifício (se houver).
- 3ª linha: trabalhadores na zona (O = Operário, M =Mineiro, L = Lenhador). Nesta representação só cabem 4, mas podem estar mais nessa zona, tal como indicado na 4ª linha.
- 4ª linha número total de trabalhadores na zona.

Para além da ilha em si, será também apresentado um sumário da situação do jogo (dia, quantidade de recursos, trabalhadores existentes, etc.). Será sempre possível ao utilizador solicitar mais informações: sobre o estado do jogo, sobre uma zona em particular, etc.

As ordens do jogador são dadas textualmente, segundo o paradigma dos comandos escritos. Cada ordem é escrita como uma frase em que a primeira palavra é um comando e as palavras seguintes são os parâmetros ou informações adicionais. Por exemplo, a ordem para construir um edifício do tipo mina de ferro na zona 3,5:

```
cons minaferro 3 5
```

A linha correspondente à ordem é escrita na totalidade, só então sendo interpretada e executada pelo jogo. Pode ser assumido que será sempre escrito tudo em minúsculas, mas o programa deve validar tudo o resto (toda a informação necessária foi escrita? Os valores que era suposto serem inteiros são mesmo inteiros? Estão pela ordem certa?)

A aplicação deve disponibilizar os dados do jogo ao fim de cada dia:

- Dia de simulação e riqueza acumulada
- Quantidade de recursos e lista de quais existem
- Quantidade de trabalhadores e lista de quais existem

Comandos

> -> Nota: os caracteres < e > não fazem parte do comando e indicam apenas o significado daquilo que deve ser escrito nesse lugar (exemplo: *<tipo>* deve aparecer no comando como *carvao* ou *central* ou … mas já sem os < >).

Os comandos são:

- ***exec <nomeFicheiro>*** Executa um conjunto de comandos existentes em *nomeFicheiro. Esse ficheiro tem os comandos*, um por cada linha, segundo a mesma sintaxe como quando são escritos diretamente no teclado.
- ***cons <tipo> <linha> <coluna>*** Constrói um edifício de um dado *tipo* na zona posicionada na *linha* e *coluna*. Nesta zona não poderá haver outro edifício e as condições necessária para a construção do tipo de edifício deverão ser observadas na altura de execução do comando. Tipo = minaf | minac | central | bat | fund | edx (abreviaturas baseadas nos nomes dos recursos em minúsculas e sem letras acentuadas).
- ***liga <linha> <coluna>*** Liga o edifício (caso exista) que está construído na zona posicionada na *linha* e *coluna*.
- ***des <linha> <coluna>*** desliga o edifício (caso exista) que está construído na zona posicionada na *linha* e *coluna*.
- ***move <id> <linha> <coluna>*** Move o trabalhador com o identificador *id* para a zona posicionada na *linha* e *coluna*.
- ***vende <tipo> <quanto>*** Vende recursos de um dado *tipo* e de acordo com *quanto* se pretende vender (de acordo com a unidade de cada recurso). Tipo = ferro | aco | carvao | mad | viga | eletr (abreviaturas baseadas nos nomes dos recursos em minúsculas e sem letras acentuadas).
- ***cont <tipo>*** Contrata um trabalhador de um dado *tipo* e este, tal como dito anteriormente, vai para a zona de pasto à espera de ordens do jogador. Tipo = oper | len | miner (abreviaturas baseadas nos nomes dos recursos em minúsculas e sem letras acentuadas).
- ***list <linha> <coluna>*** Obtém a informação do jogo, globalmente se não for indicada nenhuma linha.coluna, ou detalhada acerca de uma zona caso a sua posição *linha coluna* seja indicada.
- ***vende <linha> <coluna>*** Vende o edifício existente na zona indicada na linha e coluna especificadas. Recupera os € gastos na sua construção mas não os recursos adicionalmente usados (exemplo, as vigas na mina). Os recursos armazenados no edifício em questão são automaticamente vendidos e os € resultantes passam para o jogador.
- ***next*** Termina a fase de recolha de comandos e desencadeia as ações necessárias ao seu processamento.
- ***save <nome>*** Grava o estado do jogo em memória, associando-lhe um nome. Esta ação consiste em fazer uma espécie de *savegame* para memória, possibilitando ao jogador manter em memória diversos *snapshots* do jogo, correspondentes a diversos momentos, permitindo-lhe a qualquer

momento recuperar um desses momentos. O jogo continua ativo, mas a cópia feita para memória já não será afetada pelos comandos entretanto escritos a partir deste momento.

- *load <nome>* Recupera um dado estado do jogo em memória, identificado ao nome indicado, e carrega-o. O jogo recuperado passa a ser o que está em efeito: os comandos passam a agir sobre este.
- *apaga <nome>* Apaga um determinado *savegame* de memória.
- *config <ficheiro>* Lê o ficheiro de texto e extrai dele os preços de contratação dos trabalhadores e de compra dos edifícios. São configurados apenas os valores base de € e não os números de recursos adicionais (ex.: número de vigas para a mina). Os valores lidos sobrepõe-se aos indicados no enunciado, que apenas são considerados enquanto não for lido um ficheiro de configuração
    - O ficheiro tem o conteúdo em que cada linha tem duas palavras separadas por um espaço: nome da coisa (edifício ou trabalhador) e preço (apenas aparte dos €). Os nomes são os mesmos usados nos comandos de contratação e de construção. Exemplo

      minac 15
      len 12
- *debcash <valor>* Adicina a quantidade de € especificada em valor. A quantidade pode ser positiva (adicionar) ou negativa (remove. Serve para *debug* e teste.
- *debed <tipo> <linha> <coluna>* Adiciona um edifício a custo zero na zona linha, coluna, respeitando as restrições de não poder haver mais do que um edifício na zona. Serve para *debug* e teste.
- *debkill <id>* Remove o trabalhador com a identificação id. Serve para *debug* e teste.

5. RESTRIÇÕES DE IMPLEMENTAÇÃO

É muito importante que considere o seguinte:

- A ilha deve representada de forma dinâmica, ou seja, não se vai construir um espaço (estrutura de dados) a contar com o tamanho máximo e depois usar apenas uma parte (exceto na meta 1 onde essa simplificação é aceite, mas apenas na meta 1).
- A estrutura de dados da ilha (para guardar as zonas) não deve usar coleções da STL. Serão aceites implementações que contrariem esta restrição, mas perderão alguns pontos (poucos). Esta limitação aplica-se apenas ao armazenamento de zonas da ilha.

6. ACERCA DA IMPLEMENTAÇÃO

É muito importante que considere o seguinte:

- Apenas o essencial foi descrito. Poderá ser necessário complementar os pormenores em falta com a regras do bom senso, sem retirar dimensão nem complexidade.
- Não existe necessariamente a relação uma entidade – uma classe. É muito provável que essa relação exista, mas pode acontecer que a representação de uma entidade/conceito seja distribuída por várias classes, ou apenas como atributo de outra.

- A descrição foca principalmente as propriedades e comportamentos de cada conceito/classe. Os detalhes específicos sobre métodos e dados, incluindo os que não são mencionados explicitamente, devem ser deduzidos e implementados.
- Há mais do que uma estratégia e implementação possíveis.
- O programa deve compilar sem nenhum *warning* ou erro. O jogo deve executar sem nenhum erro ou exceção. O código deve ser robusto e completo.
- É esperado um programa orientado a objetos em C++. Uma solução não usando os conceitos de objetos (por exemplo, na lógica C) terá 0 ou muito próximo disso
- A defesa afeta a nota. Uma má defesa implicará a nota 0 ou próximo disso.

Metas e entregas

Meta 1 – Prazo: 23 de novembro

Programa que implemente as seguintes funcionalidades:

- Leitura do ficheiro de configuração.
- Construção inicial da ilha. Deve ter em atenção que a representação da ilha irá ser melhorada com matéria dada posteriormente e agora só se pretende algo que possa ser representado no ecrã.
    - Pode implementar a ilha (apenas na meta 1) através de uma estrutura de dados de tamanho fixo (por exemplo, considerando um tamanho máximo mesmo que depois a ilha seja mais pequena)
- Considere uma representação simplificada para as zonas sabendo que vai modificar essa parte da implementação quando for dada mais matéria
- Representação visual da ilha e conteúdo incluído nesta meta (ver *bullets* seguintes).
- Nesta meta não são considerados os recursos.
- Implementação da leitura e validação de todos os comandos, seja por teclado, ou seja por leitura do ficheiro de comandos. Os comandos não farão ainda nada, mas devem ser já interpretados e validados.
- Construção de edifício do tipo **minaferro** através de comandos (**cons**), tanto pelo teclado como por ficheiro (comando **exec**)
- Contratação de mineiros (comando **cont**), tanto pelo teclado como por ficheiro (comando **exec**)
- Visualização dos dados do jogo e de zonas (comando **list**)
- O projeto já deverá estar devidamente organizado em .h e .cpp separados.

Relatório – Nesta meta o relatório é simplificado, mas deve incluir a descrição das opções tomadas e a descrição das estruturas usadas. Deve também dar uma indicação da estruturação do trabalho em termos de classes (quais, para que servem / o que representam e qual a relação entre elas)

Meta 2 – Prazo: 16 de janeiro

Objetivos: o programa completo, com relatório.

Projeto a entregar (ambas as metas): CLion
As defesas são obrigatórias sempre que indicadas pelos docentes (meta 1 e meta 2).

7. CASOS OMISSOS:

-> Serão abordados nas aulas ou no *moodle*