Programação Orientada a Objetos - 2024/2025

Trabalho Prático

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;
- Não é permitida uma abordagem baseada na mera colagem de excertos de outros programas ou de exemplos. Todo o código apresentado terá de ser demonstradamente entendido por quem o apresenta e explicado em defesa, caso contrário não será contabilizado.

Tema geral

Pretende-se construir em C++ um simulador de viagens no deserto. O utilizador possui caravanas que conduz a lugares espalhados pelo deserto, transportando mercadorias, encontrando tesouros e combatendo bárbaros.

A simulação decorre num mapa que é constituído por zonas de deserto, que podem ser percorridas, e montanhas, que são intransponíveis. Haverá cidades onde as caravanas podem parar para reabastecer, comprando e vendendo itens. O mapa é constituído por uma grelha retangular. A representação visual do mapa deve mostrar que zonas são deserto (em princípio, a maior parte), montanhas, cidades e onde estão as caravanas. No final do enunciado é dada uma indicação dos caracteres a usar.

A simulação decorre por turnos: em cada turno o jogador pode indicar ações a realizar escrevendo comandos específicos para cada ação. Os comandos são escritos segundo o paradigma da “linha de comandos”, ou seja, todo o comando (palavras-chave e valores adicionais) são escritos de uma só vez numa linha.

O simulador inclui componentes que exibem comportamento autónomo ao longo de um tempo virtual. Isto significa que o simulador tem a noção de passagem de tempo. O tempo é contado numa unidade arbitrária chamada “instante” e haverá um comando para simular a passagem de 1 ou vários instantes.

Interação com o utilizador

O simulador efetuará toda a interação com o utilizador em modo consola (modo texto), sem recurso a qualquer grafismo ou cores. Não será utilizada nenhuma biblioteca auxiliar para posicionamento de caracteres no ecrã. O utilizador dará indicações sob a forma de comandos escritos, sendo cada comando totalmente escrito numa única

linha, ou seja, o comando e respetivos dados adicionais são lidos numa única linha e não será usada uma lógica de menus e submenus.

A cada novo instante, o simulador desencadeia as ações dos vários elementos do simulador, apresentando na consola (ecrã) a situação resultante e então aguardando os comandos do utilizador. Esta ideia corresponde a uma simulação por turnos, em que o turno é a oportunidade do utilizador dar indicações (comandos). Pode dar uma ou mais indicações e o simulador só avança para um novo instante por ordem do utilizador.

A apresentação da situação (o mapa e os elementos nele existentes, assim como informações adicionais) obriga ao controlo preciso da posição de caracteres no ecrã, o que não está previsto na biblioteca C++ habitual. De forma a rodear a dificuldade de posicionamento de caracteres em posições específicas no ecrã, as informações serão escritas primeiro num buffer de caracteres em memória, que depois será transcrito de uma só vez para o ecrã. O buffer deverá ser configurado com dimensão igual ou inferior à consola, desde que suficientes para apresentar os dados do simulador. De cada vez que o seu conteúdo é transcrito para a consola, o conteúdo já existente nesta será, naturalmente, empurrado para cima (*scroll up*). Este buffer será encapsulado numa classe com a funcionalidade necessária para pôr caracteres e *strings* em posições arbitrárias, e a sua elaboração faz parte do trabalho, sendo este o primeiro requisito descrito neste enunciado.

Buffer em memória

Deve existir uma ou mais classes para armazenar caracteres em memória com a finalidade de simular "um ecrã em memória" que possa depois ser transcrito para o ecrã (a consola / o terminal). Esta ideia corresponde ao conceito de buffer. Este buffer simula um ecrã, e, por conseguinte, terá uma aparência de matriz bidimensional de caracteres. No entanto, a implementação poderá ou não recorrer a *arrays*, de uma ou mais dimensões, sendo este um aspeto de implementação em aberto, desde que sejam respeitados os seguintes requisitos:

- A dimensão em número de linhas e número de colunas deve ser definida em *run time*.
- Não deve ser usado mais espaço de armazenamento do que aquele que efetivamente é necessário para os caracteres definidos no ponto anterior.
- O armazenamento dos caracteres não pode recorrer a nenhuma classe de biblioteca. Mais especificamente: nem string, nem ostringstream, nem nenhuma classe de coleção (vector, set, etc.).
- Considerando que esta funcionalidade será representada por uma classe, não pode ser assumido que apenas existirá um objeto desta classe no programa.

Este buffer destina-se exclusivamente a armazenar caracteres que depois serão impressos na consola, ou seja, serve para apoio a operações de output, sendo totalmente alheio a assuntos de leitura de dados via teclado.

A funcionalidade mínima esperada é a seguinte:

- Esvaziar o buffer (fica só com espaços);
- Transcrever o conteúdo do buffer para a consola (terminal / ecrã). Corresponde a imprimir na consola o conteúdo do buffer, recorrendo para isso aos mecanismos habituais vistos nas aulas;
- Suporta o conceito de cursor, que corresponde às coordenadas linha, coluna onde será "impresso" (armazenado) o próximo caracter;
- Mover cursor virtual para uma determinada linha e coluna;
- Imprimir os seguintes tipos de dados na posição atual do cursor: caracter, *string*, inteiro. Esta operação desloca o cursor para caracter seguinte ao último impresso;
- As operações listadas acima serão mais valorizadas se forem suportadas também pelo operador <<. Do lado esquerdo estará o objeto que representa o buffer e do lado direito estará os dados ou objetos necessários à funcionalidade;

- Havendo classes específicas relativas ao assunto do trabalho (ver mais adiante), será mais valorizada a solução que suportar a transcrição dos dados que os objetos dessas classes representam diretamente para o buffer com o operador << (na forma: objeto-buffer << objetos-de-etc).

Sublinha-se que a funcionalidade do **buffer é apenas para output e tem como objetivo facilitar o posicionamento de caracteres em posições específicas**. **A leitura de dados é feita da forma habitual vista nas aulas**. Paralelamente ao uso do buffer pode também usar-se a forma habitual de output igualmente vista nas aulas. A construção deste buffer **faz parte do trabalho**, a classe correspondente deve ser completa e correta; no entanto, a forma como é usado (se for usado) fica ao critério do aluno. **Podem** ser acrescentadas as funcionalidades adicionais que forem consideradas úteis (por exemplo, desenhar molduras, etc.)

Lógica geral de alto nível do simulador

O utilizador dispõe de um conjunto inicial (**configurável**) de moedas com as quais pode comprar as suas caravanas iniciais. Cada caravana vem com uma tripulação inicial. O utilizador usa as suas caravanas para caçar, ir de cidade em cidade e vender mercadorias. As receitas obtidas com as vendas podem ser usadas para comprar mais caravanas e mais tripulação.

O simulador inclui caravanas bárbaras que não pertencem ao utilizador. Se uma caravana do utilizador se encontrar numa célula do mapa adjacente a uma caravana bárbara, haverá um combate. Os combates podem durar vários instantes (o que se traduz em vários "turnos") e terminam quando uma caravana é destruída, ou quando uma caravana se retira, deixando de estar numa célula adjacente à outra caravana. Por adjacência entende-se apenas esquerda, direita, acima, abaixo, e os cantos não contam como adjacência.

A simulação termina ou quando o utilizador fica sem caravanas e sem dinheiro suficiente para comprar novas caravanas, ou quando escrever o comando para terminar. No final da simulação deverá ser indicado o número de instantes decorrido, o total de combates que venceu, e o número de moedas que ainda tinha.

O mapa inicial e alguns parâmetros são lidos de um ficheiro de texto (descrito mais adiante). Não é necessário gerar mapas aleatoriamente. Os parâmetros configuráveis estão assinalados no enunciado com "(**configurável**)".

Deserto

O deserto corresponde à totalidade da área geográfica onde a ação do simulador decorre. Conceptualmente, corresponde à ideia de uma grelha retangular de posições. Cada posição pode corresponder a uma montanha, a uma cidade, ou estar vazia (é o deserto). As montanhas, em princípio, ocorrem em grupos de várias posições contíguas. As cidades, estarão isoladas e nunca totalmente rodeadas de montanha (seriam inacessíveis).

Considera-se que o deserto é esférico, o que significa que se se estiver no topo e se se subir, passa-se para a posição mais ao fundo. Se igual forma, se se estiver no extremo esquerdo e se se deslocar para a esquerda, passa-se para o extremo direito. Este conceito afeta apenas os movimentos, sendo a visualização completamente indiferente quanto a isto (em caso de dificuldade: esta característica não é muito valorizada).

As caravanas (do utilizador e as bárbaras) podem apenas estar em posições de deserto ou de cidade. Quando uma caravana se encontra numa posição de deserto é visível; quando se encontra numa cidade, deixa de ser visível diretamente, tendo que se listar o conteúdo da cidade para se perceber que a caravana existe. Numa posição de deserto apenas poderá estar uma caravana de cada vez, no entanto, numa cidade pode haver muitas caravanas. As posições de montanha não podem ser atravessadas. Pode também haver itens no deserto (descrito mais adiante).

Pode haver tempestades de areia no deserto. As tempestades são sempre provocadas a pedido do utilizador, especificando este a região quadrada afetada indicando o centro (linha e coluna) e o "raio" - dimensão (quantas posições em redor do centro). A tempestade tem a duração de um instante, afetando o que existir dentro do quadrado definido pelo centro e dimensão especificados.

Cidade

A cidade ocupa apenas uma posição e deve ter pelo menos um lado acessível (deserto). A cidade permite a entrada de caravanas. São neutras, nunca havendo combates entre caravanas e cidades. Estão sempre disponíveis para fornecer novos tripulantes, comprar e vender mercadorias. Quando uma caravana entra na cidade, deixa de se visualizar essa caravana, tendo o utilizador que inspecionar o conteúdo da cidade para ver que a caravana lá se encontra. Todas as cidades têm um nome, que deve ser único, e é simplesmente uma letra. O número de cidades não é fixo e são tantas quantas estiverem no ficheiro com o mapa inicial.

Itens

Periodicamente, de forma **configurável**, aparecerão itens aleatoriamente no deserto, sempre em posições vazias (ou seja, nem montanhas, nem caravanas, nem cidades – só deserto). Os itens permanecem 20 instantes (**configurável**) desaparecendo de seguida. Enquanto existem podem ser apanhados por uma caravana que entre numa posição adjacente (acima, abaixo, esquerda, direita). O item é apanhado de forma automática pela caravana, não sendo uma opção do utilizador. Os itens são mágicos, fazendo efeitos diversos consoante o seu tipo ao serem apanhados. Os tipos no trabalho são: Caixa de pandora, Arca do tesouro, Jaula, Mina, Surpresa. Todos os itens têm a mesma representação visual, e o utilizador não tem forma de saber o que o espera. Não deve haver mais do que 5 itens (**configurável**) em simultâneo. Novos itens são colocados de 10 em 10 instantes (**configurável**) e o seu tipo é sorteado em igual probabilidade.

- **Caixa de Pandora -** Contém doenças de todo o tipo e feitio, matando 20% da tripulação da caravana que a apanha.
- **Arca do tesouro -** Tem ouro aos montes. Acrescenta mais 10% às moedas do utilizador.
- **Jaula -** Contém prisioneiros de uma guerra há muito esquecida, que são libertados e, por gratidão, juntam-se à tripulação da caravana, sem, no entanto, exceder o seu máximo.
- **Mina -** Resquício da WW2. Explode e destrói a caravana.
- **Surpresa -** Os autores do enunciado não sabem o que é que este item faz. Os alunos devem surpreender os docentes definindo este tipo. Se dois trabalhos diferentes tiverem itens surpresa iguais, a surpresa não será assim tão grande e será assumido que os trabalhos tiveram origem partilhada, cabendo aos alunos demonstrar que assim não foi (se não tiver sido).

Caravanas

As caravanas têm diversos atributos e comportamentos. São identificadas por um número. Existem vários tipos de caravanas no simulador. Com algumas diferenças entre si, todas conseguem transportar tripulantes e mercadoria. Algumas caravanas têm características mais relacionadas com transporte de mercadoria, outras estão mais vocacionadas para a missões de ataque a outras caravanas.

O deslocamento de uma caravana corresponde a movê-la para uma das 8 posições adjacentes à posição em que se encontra (não esquecer que, se estiver no limite do mapa, pode passar para o extremo oposto).

O movimento das caravanas acontece por ordem do utilizador, ou por comportamento autónomo, que varia conforme o tipo de caravana, ou por estar sem tripulantes, que também depende do tipo de caravana.

- **Por ordem do utilizador** - O utilizador especifica em cada turno a direção do deslocamento, devendo a caravana deslocar-se uma posição apenas nessa direção. Esta ordem pode ser dada uma ou mais vezes por turno, dependendo do tipo de caravana.
- **Deslocamento autónomo** - A caravana desloca-se autonomamente, exibindo um comportamento dependendo do seu tipo.
- **Deslocamento quanto está sem tripulante nenhum** - A caravana que perdeu toda a sua tripulação move-se de forma que varia conforme o seu tipo.

As caravanas podem transportar mercadoria até uma certa quantidade máxima. A unidade de carga é "tonelada". A tripulação precisa de água para sobreviver e cada caravana transporta água, que vai diminuindo em função dos instantes e da tripulação, e quando chegar a zero a tripulação sofre. Quando entra numa cidade, a caravana pode vender a sua carga (mercadoria) e reabastecer-se de água (o reabastecimento de água é automático, através do poço da cidade, ficando a caravana com o seu depósito de água a 100%).

Quando uma caravana se encontra numa cidade pode comprar ou vender mercadorias (só existe um tipo de mercadoria, designado de "mercadoria"). O preço de compra e venda de mercadorias é **configurável**, mas, por omissão, é: compra, 1 por tonelada; venda, 2 por tonelada (as viagens dão sempre lucro).

Neste simulador existem as caravanas do tipo: comércio, militar, secreta.

Caravana de comércio

- Lenta, e fraca, mas com uma capacidade de carga enorme. Pode deslocar-se uma ou duas posições em cada turno. Tem inicialmente 20 tripulantes e leva até 40 toneladas de carga. Tem capacidade para 200 litros de água. Gasta 2 litros de água por instante, ou 1 se tiver menos de metade dos seus tripulantes, 0 se não tiver nenhum tripulante
- Se estiver a mover-se de forma autónoma, tenta manter-se ao lado de uma outra caravana do utilizador, para ter proteção. Se existir algum item a uma distância de 2 linhas e/ou colunas, desloca-se para a apanhar o item.
- Se for apanhada por uma tempestade de areia tem 50% de hipóteses de ser destruída se tiver mais do que 50% da sua capacidade de carga ocupada, 25% se tiver menos. Se sobreviver, perde 25% da carga.
- Se ficar sem tripulantes move-se de forma aleatória e passados 5 instantes a caravana desaparece.

Caravana militar

- Rápida e forte, mas não transporta quase nenhuma carga. Transporta até 40 tripulantes, tem capacidade para 5 toneladas de carga e tem capacidade para 400 litros de água (expedições longas). Pode deslocar-se até 3 vezes por turno. Gasta 3 litros de água por instante, 1 litro apenas se tiver menos que metade dos tripulantes (inclusive se não tiver tripulante nenhum)
- Em movimento autónomo fica parada. Mas, se aparecer alguma caravana bárbara a 6 posições de diferença (linha e coluna), persegue-a.
- Numa tempestade de areia perde sempre 10% dos tripulantes e tem 33% de hipótese de ser destruída.
- Se ficar sem tripulantes, a caravana desloca-se sempre na mesma direção do último movimento que fez quando tinha tripulantes. Mas, passados 7 instantes, acaba por desaparecer.

Secreta

- Esta caravana tem propriedades que, de momento, são secretas. Cada grupo define as características e comportamento desta caravana. Se aparecerem dois trabalhos com as caravanas secretas idênticas,

então as caravanas não eram assim tão secretas e será assumido que os trabalhos foram partilhados, cabendo aos alunos demonstrar que não foi o caso (se não tiver sido).

As caravanas bárbaras têm sempre as características indicadas abaixo. Este tipo de caravana não está disponível ao utilizador.

- Está sempre em modo de deslocamento "autónomo". Anda aleatoriamente para uma posição adjacente, (1 posição por turno). Se vir uma caravana do utilizador a uma distância de 8 posições (em linha e/ou coluna). Nesse caso desloca-se (1 posição por turno) na direção dessa caravana tentando apanhá-la.
- Tem inicialmente 40 bárbaros armados até aos dentes. Os bárbaros não fazem uso nenhum de água, nem sequer para beber e por isso podem andar o tempo que quiserem pelo deserto.
- As caravanas bárbaras também apanham itens. Os efeitos são os mesmos (ganhar ou perder tripulantes da caravana) e, até a arca do tesouro é igual: quem ganha as moedas continua a ser o utilizador.
- Numa tempestade de areia perde sempre 10% dos tripulantes e tem 25% de hipótese de ser totalmente destruída. Se ficar sem tripulantes a caravana desaparece imediatamente.
- Os bárbaros aborrecem-se passados 60 turnos (**configurável**) e desaparecem.

Quando uma caravana fica sem água, perde um tripulante a cada turno (exceto as dos bárbaros, porque esses não fazem uso nenhum de água).

As caravanas bárbaras aparecem em posição aleatória (deserto) de 40 em 40 turnos (**configurável**).

Compras e vendas

Quando uma caravana se encontra numa cidade, o utilizador pode:

- Comprar tripulantes para essa caravana por 1 moeda cada um;
- Vender mercadoria existente na caravana (preço **configurável**, podendo ser assumido 2 moedas por tonelada), esvaziando a caravana.
- Comprar mercadoria para a caravana (preço **configurável**, podendo ser assumido 1 moeda por tonelada), não podendo exceder a capacidade de carga.

As caravanas podem ser compradas pelo utilizador. Inicialmente, existe um conjunto de caravanas estacionadas em cidades, à espera de serem compradas. Quando todas as caravanas forem compradas não haverá nenhum mecanismo de substituição. O número inicial de caravanas prontas para serem compradas é muito simples: uma caravana de cada tipo por cada cidade. Após a compra de uma caravana, a caravana passa para o controlo do utilizador, ficando no interior da cidade até que receba ordens de movimento. As caravanas compradas surgem na cidade onde foram compradas, com tripulação completa e depósito de água cheio.

As caravanas podem ser compradas a qualquer instante, desde que o utilizador tenha moedas e ainda haja caravanas disponíveis (será necessário especificar qual a cidade, que deverá ter uma caravana do tipo indicado disponível). Cada caravana custa 100 moedas (**configurável**).

Combates

Ocorrem sempre que duas caravanas utilizador vs. bárbaro se encontrem em posições adjacentes. É sorteado um número para cada caravana que vai de 0 até ao número de tripulantes dessa caravana. A caravana que tiver o maior número ganha o combate. A caravana que ganha perde 20% da tripulação e a caravana que perde fica sem o dobro desse número (em ambas as caravanas, a perda de tripulantes é definida em função do número de tripulantes da caravana vencedora).

Cada caravana é livre de fugir (no instante/turno seguinte). Se permanecerem paradas, no turno seguinte voltará a haver combate, e só termina quando uma caravana foge ou quando uma caravana (ou ambas) perdem. Se a caravana ficar sem tripulantes no contexto de um combate é eliminada de imediato (não fica com o movimento descrito acima na secção sobre caravanas). Se uma caravana for destruída em combate, a sua água passa para a caravana vencedora (sem exceder a sua capacidade máxima). Havendo perda de tripulantes, o número de tripulantes restantes é sempre um inteiro, sendo o resultado do arredondamento para baixo (idem quanto a tempestades de areia, caixas de pandora, etc.).

Se houver várias adjacências de caravanas inimigas em simultâneo (um engarrafamento de caravanas?), haverá em cada turno todos os combates relativos aos pares de adjacências inimigas existentes (por exemplo, se uma caravana tiver uma caravana inimiga em ambos os lados, essa caravana lutará com ambas em cada turno e, provavelmente, a coisa não lhe correrá lá muito bem).

Inicio da simulação

De uma forma geral, a simulação é constituída por duas fases: a primeira, muito curta e simples, em que apenas é permitido especificar o nome do ficheiro de configuração a ler. Esse ficheiro define o tamanho do mapa do deserto, o mapa propriamente dito, e alguns parâmetros gerais de configuração (moedas iniciais, número de cidades, preços, etc.), correspondendo a todos os valores assinalados no enunciado como "**configurável**"). Esta informação é obtida através da leitura de um ficheiro de texto (ver exemplo mais adiante). Após o ficheiro lido, passa-se automaticamente à fase seguinte:

Desenrolar da simulação

A segunda fase da simulação consiste na simulação propriamente dita, que decorre nos moldes já anteriormente referidos: o utilizador escreve os comandos que entender e haverá comandos que permitem fazer avançar a simulação 1 ou mais instantes/turnos. Existem os seguintes momentos distintos e em sequência em cada turno:

1. Apresentação: É colocado na consola / ecrã a informação e mapa relativos ao ponto da situação atual.
2. Leitura e execução de comandos: O utilizador indica todos os comandos que entender. Pode consultar informação adicional através de comandos para esse efeito ou dar diversas ordens às suas caravanas, comprar e vender mercadorias, etc. Esta etapa termina com o comando *prox*.
3. Execução de comportamentos automáticos: as caravanas efetuam os seus movimentos autónomos (se estiverem nesse modo), podendo apanhar itens, etc.
4. Bárbaros: nesta fase podem surgir caravanas bárbaras. As que já existirem movimentam-se.
5. Combates: são considerados apenas no final de todos os outros tipos de comportamentos automáticos.

Todos os acontecimentos que se produziram (itens apanhados, combates realizados, etc.) devem ser registados e ser apresentados juntamente com o mapa do deserto no início do turno seguinte.

Comandos do utilizador

Cada comando é uma linha de texto, formada pelo nome do comando e 0 ou mais parâmetros, separados por espaços. Os parâmetros especificam qual a caravana à qual se está a dar a ordem, ou qual a cidade, ou qual a quantidade de uma certa coisa que se está a fazer.

As caravanas são identificadas por números e as cidades são identificadas por uma letra (que não se deve repetir). Os caracteres < e > não serão escritos pelo utilizador, obviamente, e o que está entre esses < e > são valores ou nomes consoante o seu significado no contexto do comando.

Os comandos são case-sensitive, **e**xec e **E**xec são dois comandos diferentes, e, portanto, devem ser implementados exatamente como indicados de seguida.

Comandos da fase 1

- **config <nomeFicheiro>** - Lê as características do mapa do deserto e alguns parâmetros do ficheiro de texto designado por nomeFicheiro
- **sair** – sai do programa

Comandos da fase 2

- **exec <nomeFicheiro>** - Lê comandos do ficheiro de texto designado por nomeFicheiro, um por linha, e executa-os. Este comando é **muito útil** para executar um conjunto de comandos predefinido, previamente gravados num ficheiro. Permite, assim, poupar imenso tempo durante os testes, carregando uma determinada sequência de comandos reproduzível. Cada linha do ficheiro tem um comando com **a mesma estrutura e texto dos comandos que se digitam através do teclado**, portanto com a mesma interpretação e processamento, partilhando o código associado a estas tarefas e diferindo apenas de onde vêm os caracteres que o compõem (do teclado *vs.* de um ficheiro).
- **prox <n> –** Termina a fase de indicação dos comandos e vai ter início a fase de execução de comandos pendentes e comportamentos automáticos. n é opcional, e maior que 0, e indica o número de instantes a avançar. Se avançar mais do que um instante, deverá ser apresentado o mapa e outras informações em cada um dos instantes.
- **comprac <C> <T> -** Compra, na cidade C uma caravana do tipo T. T indica o tipo da caravana: C – Comércio, M – Militar, S – Secreta.
- **precos** – Lista os preços das mercadorias (igual em todas as cidades).
- **cidade <C> -** Lista conteúdo da cidade C (caravanas existentes)
- **caravana <C> -** mostra a descrição da caravana C (todos os detalhes)
- **compra <N> <M>** - Compra M toneladas de mercadorias para a caravana N, o qual deverá estar numa cidade nessa altura.
- **vende <N>** - Vende toda a mercadoria da caravana N, o qual deverá estar numa cidade.
- **move <N> <X>** - Move a caravana com o número N uma posição na direção X: D (direita), E (esquerda), C (cima), B (baixo) e as diagonais, na mesma lógica: CE (cima-esquerda), CD (cima-direita), BE (baixo-esquerda), BD (baixo-direita).
- **auto <N>** - Coloca a caravana N em “auto-gestão”, ou seja, a caravana passa a ter o seu comportamento automático, seja ele qual for, mesmo que seja “ficar parado”.
- **stop <N>** - A caravana com o número N pára o comportamento automático.
- **barbaro <l> <c>** - Cria uma caravana barbara nas coordenadas (l, c).
- **areia <l> <c> <r>** - Cria uma tempestade de areia na posição l, c com raio r posições (é um quadrado de posições).
- **moedas <N>** - Acrescenta N moedas ao jogador (pode ser um valor negativo). Isto serve para testes.
- **tripul <N> <T>** - Compra T tripulantes para a caravana N desde que essa caravana esteja numa cidade.
- **saves <nome>** - Faz uma cópia do estado do buffer (**aspeto visual apenas – não os dados do simulador**) usado para apresentar a informação apresentada no turno atual. A cópia é armazenada em memória associando-o ao nome indicado.

- **loads <nome>** - Recupera a cópia do buffer previamente armazenado em memória com o nome indicado, permitindo assim ver um determinado instante anterior (para comparação ou que for). Esta operação não altera o estado da simulação atual.
- **lists -** Lista os nomes das cópias do buffer existentes.
- **dels <nome>** - Apaga a cópia do buffer em memória associada ao nome indicado.
- **terminar** – Termina a simulação, sendo apresentada a pontuação. O simulador regressa à fase 1, onde o utilizador pode iniciar nova simulação, ou sair do programa.

Configuração

O estado inicial do mapa e alguns valores iniciais são definidos num ficheiro de texto. O formato é o seguinte:

- Primeiras duas linhas: indicam a dimensão do deserto.
- Segue-se um conjunto de linhas, todas com o mesmo tamanho, com quantos caracteres quantas as colunas do deserto e tantas linhas como o deserto. Cada caracter corresponde ao que se encontra nessa posição do deserto: '.' -> uma posição vazia; '+' -> montanha; letras minúsculas -> cidades, algarismos de 0 a 9: caravanas (inicialmente haverá apenas 9 caravanas no máximo), '!' -> uma caravana bárbara.
- Segue-se a configuração dos valores configuráveis com a sintaxe nome-da-coisa valor-dela.

O exemplo abaixo indicado mostra um mapa e os valores configuráveis. Os nomes dos valores configuráveis devem ser respeitados (são os usados neste exemplo)

```
linhas 10
colunas 20
..............+++++
.........++++++++++
.........a+++++++++
...1.........+++++++
........2.....++++++
....+c...........b++
...++++..........+++
....++............++
.....3........!.....
..........4.........
moedas 1000
instantes_entre_novos_itens 10
duração_item 20
max_itens 5
preço_venda_mercadoria 2
preço_compra_mercadoria 1
preço_caravana 100
instantes_entre_novos_barbaros 40
duração_barbaros 60
```

As cidades, caravanas e bárbaros iniciais são deduzidos pelo conteúdo do mapa.

Restrições quanto à implementação

- O programa deve compilar sem nenhum *warning*. O programa 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.
- **Não pode recorrer a vectores de vectores em nenhum local do programa (vector<vector<X>>).**

Há mais do que uma estratégia de implementação possível.

Acerca do enunciado

- É inevitavelmente longo dado que descreve características dos elementos do simulador e tenta responder a questões típicas. Extensão de texto não significa complexidade ou trabalho adicional.
- O enunciado representa 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, matéria ou complexidade ao enunciado (em caso de dúvida convém perguntar ao professor atempadamente). Não será aceitável não concretizar algo cuja necessidade é óbvia apenas porque não foi explicitamente pedido no enunciado.

Regras gerais do trabalho

As que estão descritas na FUC e nos *slides* da primeira aula teórica:

- Grupos de dois alunos no máximo. Grupos de 1 devem ser previamente justificados junto do professor.
- Defesa obrigatória. Ambos os alunos devem comparecer ao mesmo tempo e quem faltar fica sem nota.
- A defesa afeta bastante a nota, e os alunos do mesmo grupo podem ter notas diferentes.

Entregas e defesa

Data – 22 de dezembro

Requisitos: projecto CLion pronto a compilar e executar, com relatório detalhado e completo e outros ficheiros auxiliares (configuração, comandos etc.). O projeto deve estar pronto para compilar em qualquer computador e não apenas no do aluno. No relatório deve incluir uma lista com os requisitos implementados, parcialmente implementados e não implementados (com indicação da razão dos não implementados).
-> **Arquivo** zip com o nome: poo_2425_ *nome1_numero1_nome2_numero2***.zip**

Importante:

- **Apenas um dos alunos** do grupo faz a submissão e **associa obrigatoriamente a submissão ao colega de grupo.**
- **Os alunos têm de estar inscritos numa turma (podem ser turmas diferentes no mesmo grupo).**

Defesas

- Serão marcadas depois da entrega.
- São obrigatórias. Ambos os alunos (“cada um dos alunos”) deverão trazer um computador onde possam demonstrar o trabalho. A defesa é individual, mas ambos os alunos devem comparecer ao mesmo tempo.