Programação Orientada a Objectos - 2022/2023

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 que 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 uma reserva natural povoada por diversos animais. Estes animais têm comportamento autónomo e variado, deslocando-se pela reserva. O utilizador interage com o simulador, visualizando a reserva e o seu conteúdo. O utilizador pode especificar ordens que alteram o que acontece na reserva, afetando diversos aspetos da simulação e dos animais.

A interface com o utilizador será simples, em modo texto, sendo necessário posicionar o cursor em *linha,coluna*. Dado que a impressão via *cout* não tem essa funcionalidade, **será fornecida uma biblioteca** com essa capacidade. O seu uso e configuração do projeto é descrito em documento também a fornecer.

Em termos de aparência, trata-se de um programa que apresenta alguns caracteres no ecrã, alguns dos quais se movimentam quando o utilizador manda avançar para o passo seguinte da simulação.

Conceitos envolvidos

Simulador

Lida com a reserva, representando-a visualmente, e permite ao utilizador interagir com a reserva, animais e outros elementos que lá estejam. A interação é feita segundo a lógica de consola, o que significa comandos escritos (não haverá espaço no ecrã para apresentar menus e o seu uso é vedado pelo enunciado). Os pormenores de posicionamento exacto dos elementos visuais ficam em aberto desde que sejam respeitados os seguintes requisitos:

- Deve existir uma área onde é apresentada a reserva. Dado que a reserva poderá ser maior que o que cabe nessa área do ecrã, o simulador apresenta uma parte da área, permitindo ao utilizador deslizar a zona visível da reserva (cima, baixo, esquerda, direita) de forma a ver o resto do ecrã. “Deslizar” significa passar a área visível um pouco mais para o lado/cima/baixo.
- Os elementos que se encontram na reserva são representados por uma letra.
- Deve existir uma zona do ecrã onde é possível ver detalhes dos animais e outros elementos do simulador sem interferir como que se encontra na área de visualização da reserva.
- Deve existir uma zona do ecrã destinada à leitura de comandos escritos pelo utilizador. Esta zona não interfere com as outras duas referidas atrás.

O funcionamento do simulador é relativamente simples:

- Toda a simulação decorre num tempo simulado em “instantes”. Na passagem de um instante para o seguinte o simulador faz os elementos simulados desempenharem as suas tarefas, sendo modificado o estado interno da reserva e demais elementos, após o que a área visível da reserva é novamente representada no ecrã.
- Apenas se avança para o instante seguinte por ordem do utilizador.
- O utilizador é livre de mandar executar as ações que entender antes de avançar para o ecrã seguinte. As consequências visíveis dos seus comandos devem ser imediatamente representadas visualmente, excepto se se reportarem a uma área da reserva que não esteja visível.

Reserva

A reserva é um espaço plano, sem qualquer característica geográfica especial (não há “tipos de terreno”). A sua geometria tem a seguintes características:

- Corresponde a uma região retangular de NL linhas e NC colunas. Os valores NL e NC são conhecidos apenas em *runtime*, podendo estar entre 16 e 500.
- Corresponde a um espaço finito mas sem fronteiras: quando um animal se encontra numa extremidade e se desloca para o exterior, é posicionado na extremidade oposta ("deu a volta"). Isto acontece em qualquer direção: cima, baixo, esquerda, direita.

Na reserva existem essencialmente animais e alimentos. De forma a poder mais facilmente identificar e referir estas coisas na interface de utilizador, todos estes elementos partilham um ID único que é um número que começa em 1 e aumenta automaticamente em cada novo animal que nasce ou alimento que aparece (o contador é partilhado entre ambos).

Dizer que a reserva tem posições que por sua vez têm coisas nelas, ou dizer que a reserva é apenas um conceito e cada coisa sabe em que coordenadas está é, essencialmente, um assunto de implementação e o enunciado não se vai pronunciar sobre isso.

Independentemente da implementação, a reserva representa um terreno com diversas coisas que nela existem, e portanto, tudo o que existir na reserva partilha os acontecimentos gerais que ocorrem à reserva como um todo.

Animais

O conceito de animal não necessita de grandes explicações. São simulados os seguintes comportamentos genéricos dos animais

- **Nascimento**. Podem aparecer espontaneamente novos animais como resposta a um qualquer acontecimento na reserva ou por ordem do utilizador.
- **Morte**. Os animais podem morrer de forma natural ditada pelas suas características, por ação de outros animais ou de algum acontecimento que tenha ocorrido na reserva, ou por ordem do utilizador.
- **Movimento**. Os animais possuem a capacidade de se movimentarem segundo determinadas regras específicas para cada animal. Encontram-se sempre numa determinada posição da reserva. A questão acerca de isto significar que a reserva tem um animal numa determinada posição ou se é o animal que sabe que está numa certa posição é um assunto para um debate interminável no qual o enunciado não entrará.
- **Alimentação**. Os animais precisam de ser alimentados. Podem ser alimentados por ordem específica do utilizador, mas também podem procurar e alimentar-se por iniciativa própria, existindo diversas fontes possíveis de sustento tais como: objetos que se encontrem dentro da reserva, e até outros animais. Animais diferentes podem reagir de forma diferente a diversos tipos de sustento.
- **Interação com o meio ambiente**. Os animais apercebem-se do que os rodeia até uma certa distância, interagindo ou reagindo com o que "vêem" dentro dessa distância.

Outras características. Como ser vivos que são, os animais possuem as características habituais de seres vivos: possuem uma certa quantidade de massa ("peso"), estão mais ou menos saudáveis, deslocam-se de uma determinada forma, reagem de formas diferentes a alimentos, possuem mais ou menos fome, são mais ou menos agressivos, vivem durante determinado tempo, reproduzem-se, etc. Existem diversas espécies de animais, havendo características diferentes inerentes a cada espécie, mas mesmo dentro da mesma espécie os animais não são todos iguais.

Como conceito genérico, sabe-se que um animal:

- Só faz alguma coisa se estiver vivo (obviamente). Quando nascem, estão vivos.
- Exibe um comportamento como se possuísse massa (kg), tivesse uma quantidade de fome (ou fome nenhuma), tivesse saúde (ou saúde nenhuma, caso em que morrem), mais ou menos fome (um identificador numérico, um nome próprio, tivesse um nome de espécie, aparenta estar numa determinada posição da reserva).
- Tem a capacidade de se mover, de ser alimentado, de interagir e reagir às propriedades do que o rodeia até uma certa distância.
    - Quanto à alimentação: o animal vai manter internamente um histórico do que comeu. Cada entrada no histórico tem: nome do que comeu, pontos de valor nutritivo, pontos de toxicidade (ver mais adiante o que são estas coisas). Não existe limite para o número de coisas comidas por um animal.
- Só existem animais dentro da reserva (o espaço exterior é para uso exclusivo de humanos).

Espécies de animais

Tal como na natureza, existem neste simulador diversas espécies de animais que partilham características genéricas mas podem também ter comportamentos específicos e variados. O simulador irá suportar as espécies de animais descritas abaixo.

Nota: pretende-se que alguns aspectos dos animais possam ser configurados a partir de valores lido automaticamente (no inicio da execução) do ficheiro **constantes.txt**. Esses aspetos são quantidades e têm nome para melhor identificação (exemplo: saúde inicial de um coelho: "SCoelho"). O ficheiro tem um valor destes por linha no formato *nome espaço valor* (exemplo "**SCoelho 20**").

Coelho

- Representado visualmente por um "C".
- Saúde inicial = "SCoelho" = 20 pontos.
- Consegue aperceber-se do que o rodeia a 4 posições de distância à sua volta na reserva.
- Desloca-se uma ou duas posições em cada instante, aleatoriamente (distância e direção).
- Tem um peso entre 1 e 4 kg, sendo o próprio coelho que decide quanto pesa quando nasce.

- Morre espontaneamente “VCoelho” = 30 instantes após o seu nascimento. Morre se a sua saúde chegar a 0 (o que acontecer primeiro).
- A cada instante aumenta a sua fome em 1. Se tiver mais do que 10 de fome passa a perder 1 ponto de saúde a cada instante e passa a deslocar-se de 1 a 3 posições de cada vez (a fome dá asas). Se tiver mais do que 20 de fome passa a mover-se entre 1 e 4 posições de cada vez e perde 2 pontos de saúde a cada instante.
- Se se aperceber de um alimento que cheire a “verdura” nas suas redondezas, dirige-se na sua direção (neste caso o movimento deixa de ser aleatório).
- Se se encontrar na mesma posição que um alimento que cheire a “verdura”, consome-o.
- Caso se aperceba de um animal que aparenta ter mais do que 10 kg de massa, desloca-se na direção oposta (neste caso o movimento deixa de ser aleatório).
- Faz nascer outro coelho a cada 8 instantes com probabilidade de o coelho realmente nascer de 50%. Se o novo coelho nascer, ficará numa posição que não esteja a mais de 10 posições de distância (linha e coluna).

Ovelha

- Representado visualmente por um “O”.
- Saúde inicial = “SOvelha” = 30 pontos.
- Consegue aperceber-se do que o rodeia a 3 posições de distância à sua volta na reserva.
- Desloca-se uma posição em cada instante, aleatoriamente (direção).
- Tem um peso entre 4 e 8 kg, sendo a própria ovelha que decide quanto pesa quando nasce.
- A cada instante aumenta a sua fome em 1. Se tiver mais do que 15 de fome passa a perder 1 ponto de saúde a cada instante e passa a deslocar-se de 1 a 2 (aleatoriamente) posições de cada vez (faz lembrar o coelho). Se tiver mais do que 20 de fome perde 2 pontos de saúde a cada instante.
- Morre espontaneamente “VOvelha” = 35 instantes após o seu nascimento. Morre se a sua saúde chegar a 0.
- Se se aperceber de um alimento que cheire a “erva” nas suas redondezas, dirige-se na sua direção (neste caso o movimento não é aleatório).
- Se se encontrar na mesma posição que um alimento que cheire a “erva”, consome esse alimento.
- Caso se aperceba de um animal que aparenta ter mais do que 15 kg de massa, desloca-se na direção oposta (neste caso o movimento deixa de ser aleatório).
- Faz nascer outra ovelha a cada 15 instantes numa posição que não esteja a mais de 12 posições de distância (linha e coluna). A saúde da nova ovelha é igual à da ovelha original nesse instante.
- Quando morre faz aparecer numa posição ao seu lado um Alimento do tipo Corpo, cujo valor nutricional é igual ao peso da ovelha e o nível de toxicidade é 0.

Lobo

- Representado visualmente por um “L”.
- Saúde inicial = “SLobo” = 25 pontos.
- Consegue aperceber-se do que o rodeia a 5 posições de distância à sua volta na reserva.
- Desloca-se uma posição em cada instante, aleatoriamente (direção), a não ser que veja algo interessante.
- Tem um peso inicial de 15 Kg quando nasce.
- A cada instante aumenta a sua fome em 2. Se tiver mais do que 15 de fome passa a perder 1 ponto de saúde a cada instante e passa a deslocar-se a 2 posições de cada vez. Se tiver mais do que 25 de fome perde 2 pontos de saúde a cada instante.
- Não morre espontaneamente. Morre se a sua saúde chegar a 0.
- Se se aperceber de um alimento que cheire a “carne” nas suas redondezas, dirige-se na sua direção (neste caso o movimento não é aleatório).
- Se se aperceber de um animal nas suas redondezas dirige-se na sua direção com velocidade 2 posições (3 se tiver mais do que 15 de fome). Em caso de haver vários, persegue o que for mais pesado.
- Se se encontrar na mesma posição, ou numa posição adjacente a um animal que pese menos que ele, mata-o.
- Se se encontrar na mesma posição que um alimento que cheire a “carne”, consome esse alimento.
- Se se encontrar com outro animal que pese mais do que ele irá lutar com ele, morrendo um deles (aleatoriamente).
- Faz nascer outro Lobo uma única vez na sua vida. O instante após o seu nascimento em que isso acontece é aleatório, definido logo quando nasce (sendo o valor entre 5 e “VLobo” = 40 instantes). Se o lobo morrer antes de fazer aparecer outro, azar! Como consequência a população de lobos diminuirá. O novo lobo aparece numa posição que não esteja a mais de 15 posições de distância (linha e coluna).
- Quando morre faz aparecer numa posição ao seu lado um Alimento do tipo Corpo, cujo valor nutricional é 10 e o nível de toxicidade é 0.

Canguru

- Representado visualmente por um “G”.
- Saúde inicial = “SCanguru” = 20 pontos.
- Duração de vida “VCanguru” = 70.
- Consegue aperceber-se do que o rodeia a 7 posições de distância à sua volta na reserva.
- Desloca-se uma posição em cada instante, aleatoriamente (direção), excepto num caso que é descrito já a seguir.

- Se tiver menos que 10 instantes de idade desloca-se sempre de forma a ficar perto do canguru que o originou (a 4 ou menos posições de distância, linha e coluna).
- Se tiver menos que 10 instantes de vida e vir um animal na sua redondeza para além do canguru que o originou, fica com medo e desloca-se para a posição do canguru progenitor a uma velocidade de 2 posições por instante; quando atinge a posição do seu progenitor entra na sua bolsa marsupial ficando lá durante 5 instantes. Enquanto esse tempo decorre deixa de estar visível na reserva, mas o que acontecer ao seu progenitor também lhe acontece a ele.
- Tem um peso inicial de 10 Kg quando nasce. Após 20 instantes é considerado adulto e passa a pesar 20 Kg.
- Este bicho é algo estranho. Parece que não come nem passa fome. Tem uma saúde de ferro que não se altera excepto quando morre.
- Para além da duração máxima da sua vida, morre se for atacado por outros bichos. Mas o ataque nunca é da iniciativa do canguru, que é pacífico.
- Faz nascer outro Canguru a cada 30 instantes. O novo Canguru aparece a uma distância igual ou inferior a 3 (linha e coluna).
- Quando morre faz aparecer numa posição ao seu lado um Alimento do tipo Corpo, cujo valor nutricional é 15 o nível de toxicidade é 5.

Animal-mistério

- Representado por um “M”.
- Este animal faz o que os alunos bem entenderem.
- Será muito estranho se dois grupos diferentes tiverem dois animais-mistérios mesmo muito parecidos.
- Situações estranhas nos trabalhos práticos são sempre vistas com mais “atenção”.

Alimentos

Podem existir alimentos espalhados na reserva. Os alimentos têm todos as seguintes características:

- Podem ser consumidos.
- Variam algumas das suas características com o passar do tempo.
- Exibem um valor nutritivo que se traduz em pontos de saúde adicionados a quem os consome.
- Exibem uma toxicidade que se traduz em pontos de saúde que são removidos a quem os consome.
- Se forem consumidos ou simplesmente esgotarem a sua duração desaparecem da reserva.
- Exibem um ou mais cheiros, o que é útil para os animais perceberem se querem ou não consumir o alimento.

Existem os seguintes alimentos (sem exclusão de outros que os alunos queiram adicionar):

Relva

- Representado por um “r”.
- Tem uma duração de “VRelva” = 20 instantes.
- O valor nutritivo é 3.
- A toxicidade é sempre 0.
- Tem cheiro a “erva” e a “verdura”.
- Fazem aparecer outra relva passados 75% da sua duração de vida numa posição próxima, aleatória, entre 4 e 8 posições (linha e coluna) de distância. Se essa posição já tiver um alimento, não acontece nada e “tenta” novamente no instante seguinte.

Cenoura

- Representado por um “t”.
- Duração infinita (até ser consumida).
- Valor nutritivo 4.
- A toxicidade aumenta 1 ponto a cada 10 instantes até ao máximo de 3.
- Tem cheiro de "verdura".

Corpo

- Representado por um “p”.
- Duração de vida: infinita.
- A cada instante diminui o seu valor nutritivo em 1 unidade e aumenta o seu valor de toxicidade em 1. O valor de toxicidade pára de aumentar quando passarem tantos instantes como duas vezes o valor nutritivo inicial.
- Cheira sempre a “carne”, por mais tóxico que esteja (Lobo: cuidado com o que comes).

Bife

- Representado por um “b”
- Duração: 30. Passados estes instantes desaparece.
- Valor nutritivo inicial: 10.
- Toxicidade: 2, sempre.
- A cada instante diminui o seu valor nutritivo em 1 unidade até chegar a zero.
- Cheira sempre a “carne” e a “ketchup”, mesmo que já não tenha valor nutritivo.

Alimento-mistério

- Representado pela letra “a”.
- Faz o que os alunos quiserem.
- Grupos diferentes com alimentos misteriosos misteriosamente parecidos: ver comentário no animal-mistério.

Funcionalidades e comandos do utilizador

Forma de interação

Em cada novo instante o simulador disponibiliza a informação relativa à generalidade do que se passa: o número do instante em que se vai, o número total de animais, de alimentos, as coordenadas dos extremos da área da reserva visível, etc. Apresenta também a área da reserva visível. A simulação fica pausada enquanto o utilizador manda executar ações, expressas em comandos, e que são logo executadas. Os seus efeitos ficam imediatamente visíveis. A simulação apenas passa para o instante seguinte por ordem do utilizador. Passar para o instante seguinte significa fazer as ações correspondentes à passagem de um instante.

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. A linha de texto correspondente à ordem é escrita na totalidade, só então sendo interpretada e executada pelo simulador. O programa deve validar a sintaxe e coerência do comando (toda a informação/parâmetros necessários foram escritos? Os valores que era suposto serem inteiros são mesmo inteiros? Estão pela ordem certa?). Pode ser assumido que será sempre tudo escrito em minúsculas.

Comandos

Nota: na lista abaixo, os caracteres < e > não fazem parte do comando e indicam apenas o significado daquilo que deve ser escrito nesse lugar (exemplo: <*especie*> deve aparecer no comando como ***l*** (se for lobo) ou ***c*** se for coelho).

- **criar animal**
  animal <especie: c / o / l / g / m> <linha> <coluna> (coelho/ovelha/lobo/cang./mist.)
  animal <especie: c / o / l / g / m> (fica numa posição aleatória)

- **matar animal**
  kill <linha> <coluna> (identifica por posição)
  killid <id> (identifica por ID)

- **colocar alimento**
  food <tipo: r / t / b / a> <linha> <coluna> (relva / cenoura / bife /mistério)
  food <tipo: r / t / b / a> (fica numa posição aleatória)
    - "corpos" só podem ser criados colocando primeiro um animal e depois matando-o

- **alimentar diretamente animais** (no histórico de alimentação o nome do alimento é “user”
  feed <linha> <coluna> <pontos nutritivos> <pontos de toxicidade>
    - Alimenta os animais que estiverem nessa posição
  feedid <ID> <pontos nutritivos> <pontos de toxicidade>
    - Alimenta o animal com esse ID

- **remover alimento**
  nofood <linha> <coluna>
  nofood <ID> (só parametro - assume que é o ID)

- **eliminar o que quer que esteja numa posição**
  empty <linha> <coluna>
    - o que quer que esteja na posição é removido do simulador sem mais cerimónias (por exemplo, se forem animais, nem chegam a passar pelo processo de morrer nem produzir corpos).

- **ver o que se encontra numa posição**, indicando toda a informação da posição e das coisas que lá estão
  see <linha> <coluna>

- **ver informação acerca de uma elemento do simulador (animal ou alimento)**
  info <ID>

- **passar para o instante seguinte da simulação**
  n (“n” de “next” - mais rápido de escrever)
    - faz avançar 1 instante
  n <N>
    - avança N instantes, mas 1 de cada vez, executando as ações de cada um dos instantes. Após cada um é, como habitualmente, apresentada a informação relevante (sumário do que se passa no simulador e área visível da reserva).
- n <N> <P>
    - Mesmo que o anterior mas faz uma pausa de P segundos entre cada instante para dar tempo para ver.

- **listar ID dos animais na reserva**
  anim
    - Apresenta a lista de animais: ID, espécie, saúde

- **listar ID dos animais na área visível da reserva**
  visanim
    - Apresenta a lista de animais: ID, espécie, saúde

- **Armazenar o estado da reserva em memória**
  store <nome>
    - Grava o estado do da reserva em memória, associando-lhe um nome. Esta ação consiste em fazer uma espécie de savegame para memória, possibilitando ao utilizador manter em memória diversas (várias) reservas, correspondentes a diversos momentos, permitindo-lhe a qualquer momento recuperar um desses momentos. A reserva gravada continua ativa, mas a cópia feita para memória já não será afetada pelos comandos escritos a partir deste momento.

- **Reativar um estado da reserva previamente armazenado em memória**
  restore <nome>
    - Recupera um dado estado da reserva em memória, identificado ao nome indicado, e carrega-o. A reserva recuperada passa a ser o que está em efeito: os comandos passam a agir sobre esta. A reserva que estava ativa antes é descartada.

- **Carregar e executar comandos a partir de um ficheiro de texto**
  load <nome-do-ficheiro>
    - os comandos são lidos, validados e executados da forma habitual, mas em vez de virem da *stream* associada ao teclado, vêm da *stream* associada ao ficheiro
    - Não é admissível assumir um nome fixo para o ficheiro
    - Este comando pode ser executado em qualquer altura

- **Deslocar a área de visualização** (analogia: mover uma "janela deslizante")
  slide <direcao: up/down/right/left> <linhas/colunas>
    - **É aceite** uma alternativa (documentada) que seja mais fácil de escrever pelo utilizador

- **Encerrar o simulador**
  exit

Restrições quanto à implementação

- Poderá ou não ser necessário armazenar o espaço da reserva em memória (NL x NC posições): existem diversas implementações possíveis. No caso de se optar por armazenar o espaço da reserva em memória, não é razoável nem aceite ocupar espaço para a situação de maior reserva possível (500x500) quando a reserva seja menor que isso.
- Nas estruturas ou coleções de dados que tenham a ver com o assunto, não é admissível armazenar alimentos e animais na mesma coleção como se fossem o mesmo tipo de coisa.
- A reserva deve permitir mais do que um animal nas mesmas coordenadas da reserva. Quanto aos alimentos, é apenas um.
- O histórico de alimentos não pode usar coleções da STL (Biblioteca standard C++). Tudo o resto do histórico pode usar as classes STL que quiser mas o conjunto das entradas no histórico é implementado pelo aluno sem o auxílio de coleções.
- O programa deve compilar sem nenhum *warning* ou erro. O simulador 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.

Há mais do que uma estratégia e implementação possíveis.

Acerca do enunciado

- É inevitavelmente longo dado que a) é necessário descrever características dos elementos do simulador, b) tenta responder de antemão a questões que normalmente surgem. Extensão de texto não se traduz necessariamente em complexidade ou trabalho adicional.
- Deduzem-se do texto algumas classes necessárias para as entidades mais salientes, mas poderão ser necessárias outras classes.
- São omitidos intencionalmente alguns assuntos. O enunciado 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, 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. Estes aspetos em aberto incluem:
    - Funcionalidades cuja necessidade se torna óbvia por outras que são mencionadas;
    - Aspectos de implementação que envolvam o uso correto dos mecanismos e semântica de C++;
    - Aspetos de organização segundo as boas práticas de programação e do paradigma de orientação a objetos.

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 3 não são aceites. Grupos de 1 devem ser previamente justificados junto do professor.
- Em ambas as metas entregar: projeto CLion e relatório num aquivo zip com o nome indicado mais adiante.
    - Até à data de entrega podem ser pedidos mais informações e conteúdos (por exemplo, a listagem organizada num único ficheiro pdf)
- Defesa obrigatória em ambas as metas. Ambos os alunos devem comparecer ao mesmo tempo e quem faltar fica sem nota.
- A defesa afecta bastante a nota, tem carácter individual e os alunos do mesmo grupo podem ter notas diferentes.

Metas e entregas

Meta 1 – 27 de novembro

Requisitos - funcionalidades para a meta 1:

- Leitura do ficheiro de comandos e também dos valores do ficheiro constantes.txt.
- Construção da reserva. A representação da reserva irá ser melhorada com matéria dada posteriormente e agora só se pretende algo que possa ser representado no ecrã.
- Definição do conceito de Animal. Não é preciso considerar as variações inerentes às espécies e deve mesmo limitar-se aquilo que é genérico e comum a todos.
- Definição do conceito de Alimento. Idem focar apenas o que é comum a todos os alimentos.
- Representação visual da reserva e conteúdo incluído nesta meta. Inclui-se aqui a questão de ver apenas a área visível da reserva.
- 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, incluindo a sintaxe de todos os parâmetros que tenham.
- Implementar os comandos para: ver animais e alimentos, deslizar a área visível para o lado/cima/abaixo, executar comandos em ficheiro (que são também validados), e terminar.
- O projeto já deverá estar devidamente organizado em .h e .cpp.

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).

-> **Arquivo** a entregar: arquivo **zip** (não é rar nem arj - **é zip**) com o **nome obrigatório** seguinte:

poo_2223_**m1**_*nome1_numero1_nome2_numero2*.**zip**

Meta 2 – 8 de janeiro

Requisitos: programa completo, com relatório detalhado e completo. No relatório deve incluir uma lista com os requisitos implementados, parcialmente implementados e não implementados (com indicação da razão nos não implementados)
-> **Arquivo** zip com o nome: poo_2223_**m2**_*nome1_numero1_nome2_numero2*.**zip**

Em ambas as entregas:

- **Apenas um dos alunos** do grupo faz a submissão e **associa obrigatoriamente a submissão ao colega de grupo**