Programação Orientada a Objetos - 2025/2026

Trabalho Prático

O trabalho prático de POO consiste na elaboração do programa aqui descrito. O programa deve:

- Ser feito na linguagem estudada nas aulas, usando corretamente a sua semântica e os princípios de orientação a objetos;
- Usar as classes/bibliotecas standard da linguagem abordadas nas aulas, onde apropriado e permitido;
- Não devem ser usadas outras bibliotecas sem o consentimento prévio dos docentes;
- Deve concretizar as funcionalidades referidas no enunciado;
- Não é permitida uma abordagem baseada na mera colagem de excertos de outros programas, de exemplos ou outras fontes. Todo o código apresentado terá que ser demonstradamente entendido por quem o apresenta e explicado em defesa, caso contrário o trabalho não será contabilizado.

Visão geral

Pretende-se um simulador de um Jardim. O Jardim é constituído por uma área retangular, que pode ser vista como uma grelha, e que em cada posição pode não haver nada (apenas solo vazio), pode existir uma planta e pode também haver uma ferramenta.

O utilizador do simulador é o jardineiro, o qual poderá também manifestar-se dentro da área do jardim, numa determinada posição, sobre a qual poderá atuar, e podendo deslocar-se. O jardineiro pode usar ferramentas, de entre um conjunto que tem ao seu dispôr (exemplos: regador, saco de adubo), e cada ferramenta desempenha uma ação específica.

O simulador incluirá a simulação da passagem do tempo. Isto é importante para o crescimento e multiplicação ou morte das plantas. A passagem do tempo é totalmente independente do tempo real medido pelo computador, sendo descrita em unidades abstratas de “instantes” e avança-se para o instante seguinte por ordem do utilizador. A cada instante que passa as plantas exibem um comportamento que pode depender da espécie de planta, das características da zona do jardim onde se encontram, ou outros fatores.

A simulação termina quando o utilizador entender: não é um jogo com objetivos para ganhar ou perder.

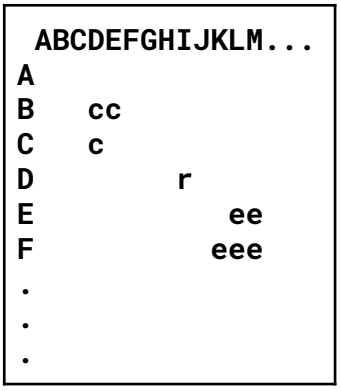
Interação com o utilizador

A representação visual do jardim consiste em apresentar toda a sua área na consola, um caracter por cada posição. Se numa determinada posição não existir nada, a sua representação é um espaço vazio. Na interação com o utilizador, a identificação da posição é feita usadando letras para a referir a linha e a coluna. Por exemplo, a posição na linha 3 e coluna 5 será a posição **ce**; a posição na linha 11 e coluna 3 será a posição **kc**. O canto superior esquerdo corresponde à posição **aa**. Para ajudar o utilizador a identificar as posições, a representação da área do jardim deve incluir uma "barra" ("régua") a toda a volta constituída pelas letras que identificam as linhas e colunas. Para não se confundir a régua com o conteúdo do jardim, a régua é impressa em maiúsculas.

O tamanho da área é decidido pelo utilizador no início da simulação, nunca mudando até ao fim desta. O tamanho máximo é 26x26, podendo ser menor. Na implementação da área do jardim:

- **Não pode recorrer a nenhuma coleção de biblioteca standard**. Esta restrição é apenas para o armazenamento das posições de solo do jardim.
- **Não deve ser usada mais memória do que realmente é necessária**.

A figura abaixo apresenta um exemplo da representação visual do jardim:



É importante considerar que este programa se destina a ambiente consola, onde apenas se consegue imprimir "para a frente e para baixo". Esta realidade tem consequências que devem ser consideradas já na fase inicial de planeamento e uso das estruturas de dados.

- Se se tentar imprimir o conteúdo do jardim percorrendo os seus elementos, um por um, esbarra-se na inexistência de funções que permitam colocar o cursor na posição linha, coluna *do ecrã* que corresponde a cada dos elementos. Mas não é necessário percorrer o conteúdo dessa forma:
- Dada a restrição de não usar coleções na implementação do armazenamento das posições de solo do jardim, a implementação mais natural é uma estrutura de dados de natureza "retangular". O jardim pode então ser representado na consola percorrendo essa estrutura linha a linha, de cima para baixo, imprimindo o conteúdo linha a linha na consola.

O utilizador irá interagir com o simulador através de comandos escritos. Esses comandos, descritos mais adiante, permitem especificar a utilização de ferramentas, listar informação, passar para o instante seguinte, e outras operações de controle.

Atenção: Ao longo do texto vão sendo indicados valores numéricos para vários parâmetros. Os valores apresentados são apenas exemplos. Os valores encontram-se especificados numa classe Settings (fornecida) que deve ser usada.

Conceitos principais envolvidos e detalhes

Jardim: O jardim consiste num conjunto de posições ("bocados de solo") com uma forma retangular. O terreno do jardim não é todo igual e cada posição pode ter características diferentes. Nomeadamente, cada posição do jardim tem os seus próprios valores de água e nutrientes, descritos por unidades sem nenhum significado especial (exemplo: "50 unidades de água"). Não há limites específicos para nenhuma das características do solo (nutrientes = 1 bilião? ok). Inicialmente, cada posição do jardim tem:

- água: valor inicial aleatório entre 80 e 100
- nutrientes: inicial aleatório entre 40 e 50

O valor inicial é aleatório, sendo eventualmente diferente para posições diferentes no jardim.

Na mesma posição pode haver: 0 ou 1 planta, independentemente do tipo, e 0 ou 1 ferramenta, (independentemente do seu tipo). O jardineiro pode ou não estar presente no jardim. Se estiver, estará numa determinada posição, podendo coexistir com o que lá se encontrar.

Se existirem várias coisas na mesma posição, apenas uma será visível. A ordem de prioridade para visualização é (do mais prioritário para o menos): jardineiro - planta - ferramenta. Para cada coisa existe um caracter que o representa no ecrã. Essa informação é dada mais adiante em cada um dos elementos.

Jardineiro. Personagem controlada pelo utilizador. Tanto pode estar na área do jardim, para efetuar ações específicas, como estar fora dele. Se se encontrar dentro da área do jardim, ocupará uma única posição, e a sua representação visual sobrepõe-se a tudo o resto, aparecendo um "*****" nessa posição. O jardineiro pode deslocar-se para posições vizinhas, sair do jardim, e voltar a entrar.

O jardineiro pode transportar um conjunto de ferramentas, mas apenas uma delas estará na sua mão, sendo essa a que está ativa e que poderá fazer algo. A ação da ferramenta é exercida sobre a posição onde se encontra o jardineiro, e essa ação é automática, atuando uma vez por instante. Não há forma de parar o seu efeito a não ser que a ferramenta deixe de estar na mão (mas continua na posse do jardineiro). O jardineiro pode mudar a ferramenta que tem na mão a qualquer momento e não é obrigado a ter ferramenta nenhuma na mão (pode estar só "a ver"). As ferramentas são muito leves e o jardineiro pode levar consigo tantas quantas quiser, sem limite à partida.

As ferramentas são obtidas de duas formas: encontrando ferramentas que se encontram na área do jardim, ou comprando-as online. A compra de ferramentas é feita por comando do utilizador. Quando o jardim é criado, são colocadas 3 ferramentas aleatórias em posições aleatórias. As ferramentas que se encontram na área do jardim são automaticamente apanhadas assim que o jardineiro entra na posição onde elas se

encontram, não sendo importante se a ferramenta está ou não visível. Sempre que uma ferramenta é apanhada, aparece “por magia” outra, aleatória, numa posição aleatória.

A ferramenta tem um número de série, que é um número crescente e único a cada uma, seja ela de que tipo for, automaticamente atribuído de fábrica sem intervenção do utilizador. A primeira tem o número 1.

Existem as seguintes ferramentas:

Regador: Ao ser aplicado, aumenta a quantidade de água em 10 unidades na posição em que o jardineiro se encontra, que são subtraídas ao regador. O regador tem capacidade de 200 unidades de água. Ao acabarem o regador torna-se inútil e o jardineiro, frustrado, atira-o para tão longe que até sai do simulador.

Pacote de adubo: Ao ser aplicado numa célula de solo, aumenta a quantidade de nutrientes em 10 unidades. A quantidade de adubo é inicialmente de 100, e quando se gastar, gastou. Os pacotes de adubo vazios são muito leves, sendo imediatamente arrastados pelo vento para o quintal do vizinho, deixando de ser um assunto para o jardineiro.

Tesoura de poda: muito boas para cortar cachos de uvas de videiras, mas, quando na mão do jardineiro vai ter o efeito de eliminar todas as plantas consideradas feias na posição em que se encontra o jardineiro. A tesoura é feita de aço cromado e nunca se desgasta.

FerramentaZ: definida por cada grupo, sendo esquisito se aparecerem dois grupos com ferramentas iguais.

A representação visual das ferramentas é a seguinte: Regador -> **g**, Pacote de adubo -> **a**, Tesoura de poda -> **t**, FerramentaZ -> **z**

Plantas

As plantas pertencem a várias espécies. Todas elas partilham algumas características comuns, no entanto, exibem características e comportamentos que variam de espécie para espécie. Todas as plantas têm uma reserva interna de nutrientes e água, podem obter (retirar) água e nutrientes do local onde se encontram, e podem até colocar nutrientes (e água) no terreno (tal como na vida real). As plantas também tem um atributo de beleza, sendo algumas plantas “feias”, outras “bonitas” e outras “neutras”, não havendo outra classificação possível. O comportamento das plantas varia consoante a espécie à qual pertencem. Neste trabalho existem: Catos, Roseiras, Ervas-daninhas, e Exóticas.

Cacto: Trata-se de um tipo de planta de beleza neutra, com as seguintes características:

- A cada instante:
    - absorve 25% das unidades de água existentes no solo na posição onde se encontra.
    - absorve até 5 unidades de nutrientes do solo
- Morre:
    - se a quantidade de água no solo for maior do que 100 durante 3 instantes seguidos, ou
    - se a quantidade de nutrientes no solo for 0 durante mais do que 3 instantes seguidos.

- Ao morrer deixa no solo todos os nutrientes que absorveu durante a sua vida (a água não).
- Multiplica-se para uma posição vizinha vazia (se existir) se a quantidade acumulada de nutrientes no cato for maior do que 100 e a água acumulada maior que 50. A água e os nutrientes presentes no cacto são divididos em iguais partes pelo cacto inicial e pelo novo.
- Nos comandos e no jardim é representado por um **c**.

Roseira: Trata-se de um tipo de planta que todos consideram bonita, que tem as seguintes características:

- Inicialmente tem 25 nutrientes acumulados dentro de si e 25 de água.
- A cada instante:
    - perde 4 unidades de água e 4 de nutrientes.
    - absorve 5 unidades de água do solo (se existir)
    - absorve 8 unidades de nutrientes do solo (se existir)
- Morre:
    - se a quantidade de água acumulada chegar a 0, ou
    - se a quantidade de nutrientes acumulada chegar a 0, ou
    - se a quantidade de nutrientes acumulada atingir 200, ou
    - se todas as posições imediatamente vizinhas tiverem uma planta.
- Ao morrer deixa no solo metade dos nutrientes que absorveu durante a sua vida e metade da água
- Multiplica-se para uma posição vizinha vazia se:
    - a quantidade acumulada de nutrientes for maior do que 100 e se existir essa posição vazia. A nova planta começa com 25 nutrientes e metade da água que a planta original tem. A roseira original fica com 100 unidades de nutrientes e com metade da água que tinha.
- Nos comandos e no jardim é representada por um **r**.

Erva daninha: Trata-se de um tipo de planta marcadamente feia, com as seguintes características:

- Inicialmente tem 5 nutrientes acumulados dentro de si e 5 de água.
- A cada instante:
    - absorve 1 unidade de água do solo, se existir.
    - absorve 1 unidade de nutrientes do solo, se existir.
- Morre sozinha passados 60 instantes.
- Multiplica-se para uma posição vizinha, matando a planta que lá estiver (se lá houver alguma), se:
    - a quantidade acumulada de nutrientes for maior do que 30 e tiverem passados 5 instantes desde a última vez que fez isso. A nova planta fica com 5 de água, e 5 de nutrientes, a inicial não perde nenhum destes.
- Nos comandos e no jardim é representada por um **e**.

Planta Exótica: Planta exótica importada pelos alunos de um país distante. Cada grupo saberá o que esta planta faz (nota: grupos diferentes importam plantas exóticas diferentes de países diferentes). Independentemente do que for, é representada visualmente por um **x**.

Funcionamento e uso do simulador

A simulação decorre num tempo simulado em “instantes” numa lógica semelhante aos jogos por turnos. O tempo avança por ordem do utilizador. Em cada novo instante as plantas atuam, as ferramentas agem, e o simulador pede mais comandos ao utilizador.

O utilizador é livre de mandar executar as ações que entender antes de avançar para o instante seguinte. As consequências visíveis dos seus comandos devem ser imediatamente representadas.

Em cada novo instante (“turno”) o simulador mostra a informação relativa à generalidade do que se passa: que plantas morreram, que plantas apareceram, e apresenta a nova representação do jardim. O utilizador pode solicitar informações adicionais sobre determinados aspetos, existindo comandos para o efeito.

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. 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 do utilizador

Na descrição dos comandos:

- <n> refere-se a um valor (número ou texto), e os caracteres **<** e **>** não fazem propriamente parte do que deve ser escrito.
- < x | y | z> indica que deve ser indicado apenas um dos x, y, z (novamente, sem os **<>**, nem ***|***).
- [v] indica que o parâmetro v é opcional, e os caracteres ***[*** e ***]*** também não devem ser escritos.

Comandos para o tempo simulado

avanca [n] (exemplo: `avanca 10`)

Avança **n** instantes (valor inteiro positivo; assume o valor 1 se não for passado o **n**), um de cada vez, em cada instante é despoletado o efeito das plantas, que reagem à passagem do tempo, assim como o deteriorar das ferramentas.

Comandos para listar informação

lplantas

Lista as plantas existentes no jardim, indicando o tipo de planta, a posição em que está (l, c) e o valor das suas propriedades internas. Mostra também as propriedades do solo onde estão.

lplanta <l><c> (exemplo: `lplanta ej`)

Lista no ecrã as propriedades da planta que se encontrar na posição l,c.

larea

Lista no ecrã as propriedades e conteúdo de cada uma das posições de solo do jardim que não esteja totalmente vazia, juntamente com a indicação da posição (l,c).

lsolo <l><c> [n] (exemplo: `lsolo df 2`)

Lista no ecrã as propriedades da posição indicada por l,c e respetiva lista de coisas que nela se encontra (e respectivas propriedades). Se tiver sido indicado o valor n, mostra a informação relativa a todas as posições dentro do quadrado de raio n centrado em l, c.

lferr

Lista no ecrã, uma por linha, as ferramentas que o jardineiro transporta, mais aquela que estiver, eventualmente, na mão deste. Para cada uma deve ser dada informação sobre que ferramenta é e os seus detalhes, incluindo a sua identificação.

Comandos para ações

NOTA:

-> Todos os comandos que alteram o jardim devem causar nova impressão deste na consola.

-> Em todas as ações que não sejam concretizadas por não estarem reunidas as condições (por exemplo, tentar colher mais plantas do que pode), o utilizador deve ser informado adequadamente.

colhe <l><c> (exemplo: `colhe fb`)

Colhe uma planta, removendo-a do jardim.

O jardineiro só pode colher 5 plantas entre dois instantes (ou seja “por turno”). Se já tiver esgotado este número, terá que avançar para o instante seguinte para poder colher mais plantas. A planta pode ser colhida mesmo que o jardineiro não esteja perto dela, ou sequer no jardim.

planta <l><c> <tipo> (exemplo: `planta fb c`)

Coloca uma nova planta do tipo indicado (sendo tipo um caracter C, R, E, X, que representa o tipo de planta), na posição do jardim l, c. Se a posição já estiver ocupada por uma planta, o comando falha.

Entre cada dois instantes (“em cada turno”), o jardineiro só pode plantar 2 plantas.

larga

Se tiver alguma ferramenta na mão, larga-a, juntando-as às outras que transporta.

pega <n> (exemplo: `pega 7`)

Coloca na mão a ferramenta como número de série, assumindo que nesse momento transporta realmente uma ferramenta com esse número.

compra <**c**> (exemplo: `compra a`)

Compra uma ferramenta do tipo indicado em c: **g**, **a**, **t**, **z**.

Comandos para o movimento do jardineiro

e -> Desloca o jardineiro uma posição para a esquerda

d -> Idem, direita

c -> Idem, cima

b -> Idem, baixo

O jardineiro pode mover-se 10 vezes entre dois instantes. Esgotado esse número, tem de avançar para o instante seguinte antes de se mover novamente.

entra <l><c> (exemplo: `entra fg`)

Faz o jardineiro entrar no jardim na posição l,c. Se o jardineiro já estiver no jardim, o efeito é o de se teletransportar para essa nova posição.

sai

O jardineiro sai do jardim.

O jardineiro só pode sair uma vez, e entrar uma vez "por turno" (entre dois instantes seguidos).

Comandos adicionais de carácter geral

jardim <n> <n> (exemplo: `jardim 10 12`)

Comando que obrigatoriamente será o primeiro a ser executado (exceto o **executar** - ver mais à frente), e depois de executado com sucesso, não será aceite novamente.

Este comando cria o Jardim com as dimensões *número de linhas* (o primeiro número indicado) e *número de colunas* (o segundo número indicado).

Por razões óbvias, nenhum dos outros comandos pode ser executado (excepto o **executar**) sem que este tenha sido executado com sucesso.

grava <*nome*> (exemplo: `grava teste`)

Salvaguarda **em memória** uma cópia do jardim e o que lá se encontrar. A cópia fica associada ao nome indicado, podendo ser recuperada mais tarde. Após a cópia feita, o jardim continua a agir e ser usado normalmente.

recupera <*nome*> (exemplo: `recupera teste`)

Recupera a cópia do jardim anteriormente feita. O jardim passa a ser como estava no momento em que a cópia agora recuperada foi feita. A cópia é eliminada.

apaga <*nome*>

Apaga a cópia do jardim com o nome indicado. Isto não afeta o jardim "atual".

executa <*nome-do-ficheiro*> (exemplo: `executa cmdteste.txt`)

Lê comandos a executar do ficheiro indicado. Existe um comando por linha, exatamente com o mesmo formato para os comandos usados pelo utilizador, aliás, o processamento dos comandos

deve ser o mesmo independentemente da origem dos mesmos (introdução pelo teclado ou leitura do ficheiro - vai dar ao mesmo e só muda a origem do texto).

fim
Encerra o programa. Isto implica uma saída “limpa”, ou seja, libertar todos os recursos alocados.

Classe de configuração

Todos os valores referidos no enunciado (exemplos: nutrientes iniciais do solo, duração de vida da planta y, etc.) devem ser obtidos pelas constantes que se encontram já definidas na classe Settings. Os nomes das constantes são auto-explicativos. Esta classe é fornecida, estando o seu código no ficheiro que acompanha este enunciado. O uso desta classe é obrigatório. Os valores das constantes podem ser alterados para efeitos de testes, mas recomenda-se que sejam repostos na versão entregue. Durante a defesa, poderá ser pedida uma alteração a estes valores e o programa deverá assumir os novos valores.

Restrições quanto à implementação

- O programa deve compilar sem nenhum *warning*. Deve executar sem nenhum erro ou exceção. O código deve ser robusto e completo.
- É esperado um programa verdadeiramete orientado a objetos. Não o sendo, terá uma cotação de 0 ou muito próximo disso.
- **A implementação da área do jardim (o conjunto de posições de solo) não pode recorrer a nenhuma coleção de biblioteca standard** (não pode usar, por exemplo, vectores para o armazenamento do solo, e muito menos vectores de vectores).
- Todos os valores indicados para plantas (quantidades de nutrientes, água, etc.) podem ser modificados pelos alunos se isso ajudar a testar o programa. Para tal devem modificar as constantes na classe Settings (fornecida), mas na entrega devem repor os valores iniciais.

Há mais do que uma estratégia de implementação possível.

Importante: O enunciado é longo para tentar responder a perguntas que ocorrem frequentemente. Quantidade de texto não significa quantidade de trabalho. Pode haver omissões que podem ser resolvidas com recurso à matéria das aulas e ao bom senso. Na resolução não podem ser tomados atalhos que removam matéria ou contrariem o enunciado. Em caso de dúvida, devem abordar os docentes nas aulas.

Regras gerais do trabalho

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

- Grupos de dois alunos no máximo, sem exceção.
- Em ambas as metas entregar: projeto CLion e relatório num arquivo zip com o nome indicado mais adiante.

- Apresentação/defesa obrigatória em ambas as metas. Ambos os alunos devem comparecer ao mesmo tempo e quem faltar fica sem nota.
- A defesa tem carácter individual, afecta bastante a nota, e os alunos do mesmo grupo podem ter notas diferentes.

Metas e entregas

Meta 1 – 2 de novembro - 12h00

Requisitos da meta 1

- Leitura dos comandos e respetiva validação (mesmo que não façam ainda nada, devem ser validados quanto à sintaxe e parâmetros).
- Planeamento das classes: que classes existem, qual o relacinamento entre elas, qual o seu objetivo e quais as funções diretamente relacionadas com o objetivo da classe.
- O projeto já deverá estar devidamente organizado em ficheiros .h e .cpp.
- **Relatório** – Nesta meta o relatório é simplificado, mas deve incluir a descrição das opções tomadas e a descrição das estruturas de dados 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). Máximo: 5 páginas (a capa e o índice não contam).

Entrega -> Arquivo **zip** (não é rar nem arj - **é zip**), contendo o projeto CLion e o relatório em pdf. O arquivo tem obrigatoriamente o **nome** seguinte: poo_2526_**m1**_*nome1_numero1_nome2_numero2*.**zip**

Meta 2 – 20 de dezembro

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 dos não implementados).

Entrega -> arquivo **zip** (não é rar nem arj), com o projeto CLion e o relatório em pdf. O arquivo tem obrigatoriamente o **nome** seguinte: poo_2526_**m2**_*nome1_numero1_nome2_numero2*.**zip**

Em cada uma das entregas:

- **Apenas um dos alunos** do grupo faz a submissão e **associa obrigatoriamente a submissão ao colega de grupo.**
- **Na escolha de um horário de defesa, o procedimento é o mesmo: apenas um aluno do grupo marca o horário, mas tem de confirmar primeiro a disponibilidade do colega.**
- **Os alunos têm de estar obrigatoriamente inscritos numa turma (podem ser turmas diferentes no mesmo grupo).**