Programação Orientada a Objetos

Trabalho Prático - Relatório

Simulador Jardim

2025/2026

Diogo Coutinho Amor Antunes - 2018016615
Joao Pedro Bento Oliveira - 2022127823

Coimbra, 31 de Dezembro de 2025

Conteúdo

1 - Introdução

O trabalho prático da disciplina de Programação Orientada a Objetos tem como objetivo a construção de um simulador de um jardim, desenvolvido em C++. Ao longo do projeto foram implementadas as funcionalidades previstas no enunciado, incluindo a modelação do jardim, a interação com o utilizador através de comandos, a evolução temporal da simulação e a gestão de diferentes entidades como plantas, ferramentas e o jardineiro.

Neste relatório é apresentada a descrição das principais classes do sistema, bem como as relações estabelecidas entre elas, justificando as decisões de organização e de design adotadas. São ainda abordadas as funcionalidades implementadas e o funcionamento geral do simulador, evidenciando a aplicação dos conceitos fundamentais da programação orientada a objetos.

2 - Descrição das Classes

2.1 Simulador

A classe Simulador funciona como o controlador principal da aplicação, é a partir dela que existe a interação com o utilizador, interpreta os comandos e executa os mesmos, invocando as operações sobre o jardim e dando o feedback das operações ao utilizador. Gere o ciclo de execução do programa, a criação e inicialização do jardim, controla os instantes da simulação, suporta a execução de comandos a partir de ficheiros texto e via consola e suporta a gravação e reposição de estados do jardim através dos comandos guardar e recuperar.

```
struct Snapshot {
    Jardim jardim;
    int instante;
    bool jardimInicializado;

    Snapshot() : instante(0), jardimInicializado(false) {}
    Snapshot(const Jardim& j, int inst, bool ini) : jardim(j), instante(inst), jardimInicializado(ini) {}
};

std::map<std::string, Snapshot> copias;
```

Para a gestão das cópias do jardim foi criada a estrutura snapshot, para facilitar o armazenamento das cópias. Isto permite-nos guardar o jardim e os seus elementos, os instantes e a flag do jardim inicializado. O conjunto de snapshots é guardado numa estrutura **std::map <string, Snapshot> copias,** ou seja uma coleção de estados do

programa guardados. Foi utilizado o map, para associar cada estado do programa guardado a um nome escolhido pelo utilizador, como pede no enunciado.

2.2 Comando Parser e Comando

A classe ComandoParser trata da interpretação e validação dos comandos escritos pelo utilizador. Para auxiliar na validação criamos a classe comando que é uma estrutura que armazena o tipo, um array de strings que divide o comando em parâmetros, o número de parâmetros e uma mensagem de erro.

```
struct Comando {
    int tipo;
    std::string tokens[8];
    int numTokens;
    std::string erro;

    Comando() : tipo(ComandoTipo::CMD_INVALID), numTokens(0), erro( s: ""){}
};
```

Com esta estrutura ficou muito mais fácil validar a sintaxe e os parâmetros dos comandos escritos pelo utilizador.

2.3 Jardim

A classe jardim é o ambiente principal da simulação, tendo como propriedades:

- Linhas e Colunas - Dimensões do jardim, definindo os limites do mundo e utilizadas para validação de posição.
- Células - Array dinâmico para todas as posições do solo.
- Jardineiro - O Jardim fica como dono do Jardineiro para centralizar o ponto de acesso a todas as ações como plantar, colher, pois é o único objeto que pode validar as coordenadas e possui o array de células e é o único que sabe qual célula corresponde no mapa.

```
const int MAX_LINS = 26;
const int MAX_COLS = 26;

class Jardim {

    int lins;
    int cols;
    Celula* celulas;
    Jardineiro jardineiro;
```

Atua entre o mundo exterior (simulador) e o armazenamento de dados internos (célula), como o solo e existirem as plantas e as ferramentas.

O jardim tem como principais responsabilidades:

- Gestão do mapa do jardim, criação e inicialização do mesmo, aceder a cada uma das células por coordenadas e validar limites;
- Gestão de memória, é aqui que é feita a alocação de memória para as células e guardar e libertar as plantas e as ferramentas, centralizando assim esta responsabilidade;
- Operações sobre o estado do jardim. É este o responsável por colocar e remover uma planta numa célula, colocar e remover uma ferramenta numa célula, consultar o conteúdo de uma célula e o estado do solo, assim como a representação visual do jardim;
- O jardim é também responsável pela evolução temporal dos elementos do jardim. Processar o comportamento de cada planta em cada instante (absorção e perda de recursos), remove plantas que morrem, tratar da multiplicação e novas plantações;
- O jardim é também responsável por atualizar os estados por turno, reset dos contadores de uso de ferramentas e limites por turno, assim como pela movimentação do jardineiro pelo jardim;
- Por fim, a classe Jardim é responsável por permitir a cópia integral e consistente do estado do sistema, o que é realizado através da implementação de um construtor por cópia e de um operador de atribuição. Sempre que um objeto jardim é copiado, o construtor por cópia é invocado, sendo utilizado para recriar todo o estado interno do jardim. Este processo envolve a cópia das dimensões e a alocação de uma nova estrutura de células. Para cada célula são copiados os dados internos do solo, as plantas existentes, as ferramentas e todos os seus recursos.

2.4 Solo

Solo representa as características físicas do solo numa determinada célula do Jardim. Armazena os valores de água e nutrientes e disponibiliza métodos para a sua modificação (adição, remoção e consulta), apoiando de forma transparente o funcionamento das plantas e ferramentas durante a simulação. Esse modelo torna cada célula única e própria para diferentes estratégias de crescimento de plantas

2.5 Célula

Célula é a unidade base do terreno do Jardim, agregando o objeto Solo e referências opcionais para uma planta e, ou uma ferramenta. A célula serve como um ponto de controlo para todas as interações. Qualquer ação que o Jardineiro ou por incremento dos instantes que existam numa coordenada específica deve, primeiramente, interagir com a célula correspondente. Esta classe permite aceder e modificar o solo, e depois plantar ou colher as plantas, assim como a adição e remoção de ferramentas no jardim, modificando os seus ponteiros. A célula também é a responsável por determinar o caráter de exibição em cada coordenada do jardim.

2.6 Planta

Planta constitui uma classe abstrata, base para qualquer tipo de planta no simulador. Define as propriedades e métodos comuns, como os valores de água, nutrientes, estado de vida, indicador se vai multiplicar e processamento do ciclo de instantes. As classes derivadas Cacto, Roseira, PlantaExotica e ErvaDaninha implementam comportamentos distintos relacionados com absorção de recursos, multiplicação, morte e interação com o solo, seguindo os requisitos definidos para cada espécie.

As principais responsabilidades da classe base são:

- A definição de funções virtuais, sendo as mais importantes:
    - **virtual void ProcessaInstante(Celula&) -** Este método é responsável por atualizar o estado da planta e do solo em cada instante da simulação. É chamado pelo jardim sempre que o tempo avança, sendo o principal ponto onde se implementam as regras de crescimento, sobrevivência e morte. Sendo esta uma função virtual, cada planta define o seu próprio comportamento.
    - **virtual Planta* multiplicar()** - Este método define como cada uma das plantas se multiplicam, criando uma nova planta quando se reúnem as condições. É chamada pelo jardim, quando a flag **prontaParaMultiplicar** é ativada na função anterior.
    - **virtual Planta* clone() -** O método clone permite criar uma cópia exata de uma planta, mantendo a sua espécie, atributos internos. É essencial para a cópia do jardim.

2.6.1 - Planta Exótica

Para a planta Exótica, decidimos que é uma planta de beleza neutra com as seguintes características:

- Inicialmente tem 40 unidades de água e nutrientes;
- A cada instante perde 3 unidades de àgua;
- Absorve até 5 unidades de àgua e até 5 unidades de nutrientes do solo;
- Morre se a água acumulada for inferior a 10 unidades durante 5 instantes consecutivos;
- Ao morrer deixa todos os nutrientes no solo, não deixando àgua nenhuma;
- Multiplica se se tiver mais de 120 de altura (água + nutrientes), quando se multiplica, perde 40% dos seus nutrientes e água.
- A nova planta nasce com metade do valor original inicial da planta exótica, ou seja 20 unidades de água e nutrientes.

2.7 Ferramenta

A classe Ferramenta representa todos os objetos que podem ser manuseados pelo jardineiro e aplicados a uma célula do jardim para produzir um efeito (alterar solo ou planta, remover, adicionar recursos). Tal como acontece com Planta, esta classe existe para permitir que diferentes ferramentas/suplementos sejam tratados de forma uniforme através de polimorfismo.
Neste programa as ferramentas podem estar numa célula, no inventário do jardineiro ou na mão do mesmo.
A classe Ferramenta tem como atributos, o nrSerie (identificador único da ferramenta), recursos (capacidade de recursos restantes, ou usos), e uma flag usadaNesteInstante, para sinalizar se aquela ferramenta já foi utilizada naquele instante.

As principais responsabilidades da classe base são a definição dos métodos virtuais, sendo os mais importantes:

- **virtual bool usa(Celula&) -** Este é o método principal da ferramenta, estando aqui implementado o efeito da ferramenta sobre a célula concreta. Cada ferramenta implementa individualmente esta função, tendo em conta o seu comportamento. Esta função é utilizada pelo jardineiro para poder utilizar a ferramenta que tem na mão automaticamente.

- **virtual Ferramenta* clone() -** Permite fazer cópias corretas dos objetos polimórficos da classe ferramenta. Este método é essencial para a implementação das cópias do programa, no que toca às ferramentas que estão no jardim, como no inventário do jardineiro.

- Também é aqui que se faz o controlo de uso das ferramentas em cada instante.

2.7.1 Suplemento Natural

Tem capacidade de 75 nutrientes e afeta apenas as plantas bonitas ou neutras. Quando o jardineiro com este em posse, estiver numa posição com uma destas plantas, a planta tem um incremento de 15 unidades de nutrientes.

2.8 Inventário

A classe Inventário é responsável por gerir todas as ferramentas disponíveis para o jardineiro, controlando as que estão armazenadas e aquela que está na sua mão para a puder utilizar. Serve como elemento de ligação entre as ferramentas e o jardineiro.

```
class Inventario {
private:
    std::vector<Ferramenta *> ferramentas;
    Ferramenta *naMao;
```

Foi utilizado um vetor dinâmico de ferramentas, guardando um ponteiro da ferramenta. Foi utilizada esta estrutura de dados para poder guardar um número variável de ferramentas, facilitar a inserção e remoção das mesmas.

Para além do vetor, o inventário mantém um ponteiro para a ferramenta atualmente selecionada (naMao), permitindo distinguir claramente entre a ferramenta ativa e as restantes ferramentas armazenadas. Esta separação simplifica a utilização das ferramentas pelo jardineiro e a implementação de operações de troca.

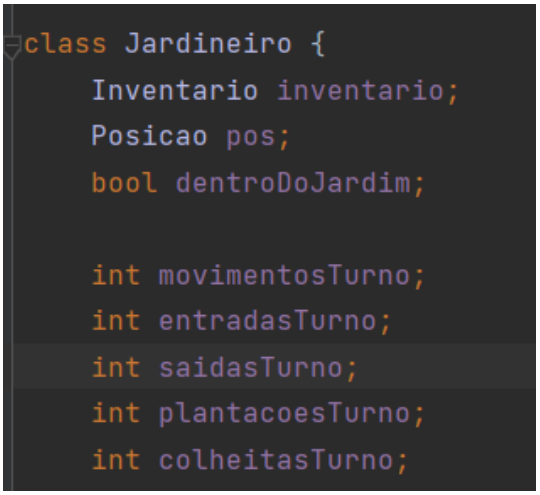
Responsabilidades principais do inventário:

- Adicionar ferramentas ao inventário, tanto vindas do jardim, quando o jardineiro passa em cima de uma célula que contém uma ferramenta, como quando o jardineiro compra uma;
- Mudar para a mão uma ferramenta que tem no seu inventário e depositar aquela que tinha selecionada;

- Devolver ao inventário a ferramenta que tem em posse;
- Listar ou consultar o inventário;
- Eliminar uma ferramenta quando não tem mais recursos;
- Criar as cópias das ferramentas que tem no seu inventário, através de um construtor por cópia utilizando os métodos clone() da ferramenta.

2.9 Jardineiro

A classe Jardineiro representa o agente que interage com o jardim. É responsável por executar as ações pedidas pelo utilizador (através de comandos), respeitando as regras do sistema (limites por instante) e utilizando as ferramentas disponíveis no inventário. Em termos de arquitetura, o Jardineiro funciona como uma camada de coordenação de ações, enquanto o Jardim continua a ser o dono do estado global



O jardineiro guarda a sua posição no jardim, para saber com que célula vai interagir, assim como validar movimentos e ações.
Possui também um objeto inventário, para poder utilizar e guardar as ferramentas.
O jardineiro é também aquele que controla os limites de movimentos, plantações, colheitas, entradas e saídas por turno.

Responsabilidades principais do jardineiro:

- Ações de movimento, entrar e sair do jardim, validação de limites do jardim e validação de movimentos por turno. Estes movimentos desencadeiam ações automáticas associadas à célula onde pisa, como recolher ferramentas e utilizar as mesmas;
- Interação com as células,, remover plantas (no caso da tesoura), apanhar ferramentas, utilizar as ferramentas que tem na mão;
- Gerir o inventário, adicionando ferramentas, colocar uma ferramenta na mão e devolvê-la ao inventário e eliminar a ferramenta quando esta fica sem recursos;
- É responsável por controlar os limites de ações por turno, como movimentos, colheitas e plantações.

2.10 Posição

Posição atua como uma classe de utilidade para registar as coordenadas (linha e coluna) de qualquer elemento no Jardim. Disponibiliza métodos de validação de limites e conversão entre diferentes formatos de representação das coordenadas, contribuindo para a robustez e previsibilidade das operações sobre o terreno.

3 - Relações entre classes

As relações entre classes neste simulador foram concebidas de forma a ilustrar os principais tipos de associação na Programação Orientada a Objetos, promovendo clareza estrutural, robustez e facilidade de manutenção do código. Nesta seção descrevem-se as relações existentes entre as principais classes do sistema.

O Simulador é o responsável por coordenar a execução do programa. É o simulador que cria e mantém o Jardim, controla o avanço do tempo e gere a gravação e recuperação de estados da simulação. Neste sentido, o simulador pode ser visto como o "dono do mundo", uma vez que controla o ciclo de vida do jardim e o estado global do sistema.

O Jardim representa o modelo central da simulação. Esta classe é responsável por gerir a estrutura do sistema, alocando dinamicamente o array de Celula que constitui a grelha do jardim. O Jardim é também responsável pela libertação dessa memória no seu destrutor, sendo assim o dono das células. Para além disso, o Jardim possui um objeto do tipo Jardineiro, refletindo o facto de o jardineiro fazer parte integrante do estado do mundo simulado.

A Celula representa a unidade básica do jardim e encapsula o estado local de cada posição da grelha. Cada célula é dona do seu Solo, uma vez que este é armazenado por composição; quando a célula é destruída, o solo é automaticamente destruído. Por outro lado, a célula mantém apenas uma relação de agregação com as classes Planta e Ferramenta, armazenando ponteiros para estas. Assim, uma célula pode conter uma planta e/ou uma ferramenta, mas não é responsável pela sua criação nem destruição, delegando essa responsabilidade ao Jardim.

No caso do Jardineiro, esta classe representa o agente que interage com o jardim. O jardineiro é dono do seu Inventario, uma vez que este é criado e destruído juntamente com o jardineiro. O inventário, por sua vez, mantém uma relação de agregação com as ferramentas, armazenando referências (Ferramenta*) para estas. As ferramentas não são exclusivamente pertencentes ao inventário, podendo existir quer no inventário

quer colocadas em células do jardim, o que justifica a escolha desta relação de agregação em vez de composição.

As classes Planta e Ferramenta são classes abstratas que definem interfaces comuns para os respectivos tipos de entidades. O sistema trabalha com estas entidades através de ponteiros para as classes base, permitindo polimorfismo. As classes derivadas implementam comportamentos específicos através de métodos virtuais, como processaInstante, usa e clone.

Em conjunto, estas relações permitem uma separação clara de responsabilidades: o simulador coordena a execução, o jardim gere o estado global e a evolução temporal, as células encapsulam o estado local, o jardineiro executa ações e o inventário gere ferramentas. Esta organização contribui para um sistema modular, coerente e facilmente extensível.

4 - Funcionalidades Implementadas

Funcionalidade	**Implementada**
Criação do Jardim	Sim
Representação do Jardim	Sim
Plantação de plantas	Sim
Tipos de Plantas	Sim
Evolução por Instantes	Sim
Morte de Plantas	Sim
Multiplicação de Plantas	Sim
Solo com recursos	Sim
Movimentação do jardineiro	Sim
Limites por Turno	Sim
Tipos de Ferramentas	Sim
Visualização dos dados do Jardim	Sim
Uso de ferramentas	Sim

Execução de comandos e feedback	Sim
Execução de comandos por ficheiro	Sim
Gravar, recuperar e eliminar cópias	Sim
Fim da simulação	Sim