Relatório Trabalho Prático – “Trabalho Jardim”

*Table of Contents*

***Tabela de Figuras***

* Resumo

Este trabalho consiste no desenvolvimento de um simulador de jardim em C++ utilizando programação orientada a objetos. O simulador modela um ecossistema virtual onde diferentes espécies de plantas competem por diferentes recursos com água e nutrientes, tudo isto num terreno retangular, enquanto um jardineiro, controlado pelo utilizador pode intervir através de várias ações e ferramentas.

Nesta fase foi criada a estrutura base do simulador de um jardim, com ênfase na leitura e validação de comandos, planeamento das classes e organização do projeto em ficheiros .h e .cpp.

* Equipa de trabalho

Rodrigo Beja da Costa

João Carlos Lopes Marques

* Desenvolvimento

**Objetivos da meta 1:**

* Implementar a leitura e validação sintática dos comandos introduzidos pelo utilizador.
* Definir a estrutura de classes do simulador e as relações entre elas.
* Organizar o código-fonte em ficheiros separados por classe.
* Garantir que o projeto compila corretamente no CLion e segue os princípios da disciplina de POO.

**Estrutura de classes**

O projeto encontra-se dividido por componentes lógicos, respeitando herança, encapsulamento e separação de responsabilidades.

|  |  |
| --- | --- |
| **Jardim** | Representa a grelha do terreno e contém uma matriz de objetos “Posicao”. Gere a impressão e evolução temporal do simulador. |
| **Posicao** | Representa uma célula do solo, com valores de água, nutrientes e ponteiros para uma “Planta” e “Ferramenta”. |
| **Planta (abstrata)** | Define atributos comuns (água, nutrientes, beleza) e métodos genéricos como “atualiza()” ou “verificaMorte()”. |
| **Cato, Erva, Roseira,cantora** | Subclasses concretas de “Planta” que implementam comportamentos específicos definidos no enunciado. |
| **Ferramenta (abstrata)** | Classe-base das ferramentas do jardineiro. Contém identificador único e método virtual usar(). |
| **Adubo, Tesoura,regador,pulverizador** | Subclasses que implementam a ação concreta de cada ferramenta. |
| **Jardineiro** | Controla a posição do jogador, ferramentas transportadas e ações permitidas por turno |
| **Interface** | Responsável por ler e interpretar os comandos, validando a sintaxe e encaminhando para as ações correspondentes. |
| **Settings** | Fornecida no enunciado, contém constantes de configuração (valores de nutrientes, etc..) |

Dentro de todas estas classes, temos várias relações, passando então a descrever:

* “Jardim” contém várias “Posicao”;
* Cada “Posicao” pode conter uma “Planta” e/ou uma “Ferramenta”;
* “Planta” e “Ferramenta” são hierarquia de herança;
* “Jardineiro” atua sobre o “Jardim” e interage com as “Ferramentas”;
* “Interface” coordena toda a comunicação entre utilizador e simulador.

**Estruturas de dados**

O **Jardim** é representado por uma matriz dinâmica de objetos “Posicao”, implementada sem recorrer a coleções standard (std::vector), conforme exigido no enunciado.

A matriz é criada com o número de linhas e colunas definido pelo comando “jardim <n> <m>”, e percorre-se linha a linha para a impressão na consola.

Cada “Posicao” guarda:

* Int agua; - valor de água no solo
* Int nutrientes; - valor de nutrientes
* Planta\* planta; - ponteiro para a planta presente
* Ferramenta\* ferramenta; - ponteiro para a ferramenta presente

O inventário do jardineiro usa uma estrutura dinâmica simples e a ferramenta ativa é guardada num ponteiro separada.

**Leitura e Validação de comandos**

A leitura dos comandos é efetuada pela classe “Interface”, através da função que processa cada linha introduzida.

O texto do comando é dividido em tokens, sendo o primeiro interpretado como o nome do comando. São verificadas:

* Quantidade e tipo de parametros;
* Formato das posições (<linha><coluna> -> letras minusculas entre ‘a’ e ‘z’);
* Tipos de ferramentas válidos (‘g’, ‘a’, ‘t’, ‘z’);
* Valores numéricos positivos (“avanca n,” etc..).

Mesmo que alguns ainda não tenham uma ação concreta, estes são os comandos já reconhecidos:

jardim <n> <m>;

avanca [n];

planta <l><c> <tipo>;

colhe <l><c>;

entra <l><c>;

sai;

pega <n>;

larga;

compra <c>;

fim;

Erros de sintaxe devolvem mensagens descritivas na consola.

**Organização do projeto**

Estrutura atual dos ficheiros principais:

Jardim.h / Jardim.cpp;

Posicao.h / Posicao.cpp;

Planta.h / Planta.cpp;

Cato.h / Cato.cpp;

Erva.h / Erva.cpp;

Jardineiro.h / Jardineiro.cpp;

Ferramenta.h / Ferramenta.cpp;

Adubo.cpp;

Tesoura.h / Tesoura.cpp;

Interface.h / Interface.cpp;

Regador.h/Regador.cpp

Pulverizador.h/Pulverizador.cpp

Cantora.cpp/Cantora.h

Settings.h;

main.cpp;

comandos\_jardim.txt;

Cada classee possui o seu ficheiro-cabeçalho e implementação.

O main.cpp inicia a interface e processa o ciclo principal de leitura de comandos.

* Conclusão

A meta 1 cumpre os requisitos definidos:

* O parser e a validação de comandos estão implementados;
* As classes principais foram criadas e estruturadas de acordo com a arquitetura planeada;
* O código encontra-se devidamente separado em ‘.h’ e ‘.cpp’.

Na próxima fase (Meta 2) serão implementadas as lógicas completas das plantas, o funcionamento detalhado das ferramentas, persistência de dados ETC .