Programação Orientada a Objetos

Relatório da Meta 2
2021/2022

Trabalho realizado por:

Beatriz Maia 2020128841

João Santos 2020136093

Introdução

Com este trabalho pretende-se construir em C++ um jogo de construção e desenvolvimento. Será atribuída ao jogador a concessão de uma ilha e o jogador deve desenvolver essa ilha, industrializando-a e construindo todo um complexo fabril

A geografia da ilha consiste num conjunto de zonas adjacentes umas às outras, havendo vários tipos de zona possíveis. O jogador pode edificar edifícios, que ficarão colocados nas zonas que o jogador escolher (no máximo um edifício por zona). O jogador irá gerir uma equipa de trabalhadores que incluirá diversos tipos de trabalhador. O jogador pode movimentar os trabalhadores na ilha para que estes possam desempenhar as suas tarefas. Ao logo do jogo irão ser usados recursos que são produzidos na ilha. A ação do jogo é controlada por ações que o jogador especifica escrevendo ordens (“comandos”).

O jogo termina quanto o jogador quiser, ou quando fica impossibilitado de continuar por já ter perdido todos os seus recursos/trabalhadores. A pontuação do jogo é a riqueza acumulada pelo jogador no momento em que o jogo termina.

Funcionamento do código:

A classe Interface é responsável pela leitura e processamento de todos os comandos digitados.

```
class Interface {

public:
    Interface();
    ~Interface();

// Menu Inicial (Preparacao)
    int comeca(); //função que inicia o programa
    void static menucomandosInicio(); //lista de comandos inicial
    void instrucoes();

// Menu durante o jogo (Jogar)
    void static menucomandosJogo(); //lista de comandos durante o jogo
    int jogada(int linhas, int colunas);
};
```

A classe ilha é responsável pela criação da ilha através de um ponteiro de zonas, pela visualização desta, armazenamento de dados de recursos e trabalhadores e por todas as funções(que chamam uma outra função na classe Zona) que envolvam estes dois e edifícios de cada zona (recolher recursos, alterar o estado dos edifícios, contratar trabalhadores, etc).

```
class ilha{
protected:
    int nlinhas;
    int ncolunas;
    int nTrabalhadores=0;

public:
    ilha(){};
    ilha(int linhas, int colunas) : nlinhas(linhas), ncolunas(colunas){}

    void configIlha();
    void mostrarIlha();

    void rZona(int a);
    void rEdificio(int a);
    void rTrabalhadores(int a);

    vector <Zona> pointerZonas;

    int sucessoEdificio=0;

    // Recursos e informação
    float dinheiro = 250;
    int dia = 1;
    int vigasMadeira = 99;
    int madeira = 10;
    float ferro = 10.0f;          // f no final para o compilador nao tornar o valor em double por default
    int barrasAco = 10;
```

```
    // Trabalhadores
    void contratarTrabalhador(string tipo);
    int getNumeroTrabalhadores(int a);
    void moverTrabalhador(string id, int lin, int col);
    void incrementarDiasTrabalhador();
    void despedirDiasContrato();
    int contadorMineiros = 0;
    int contadorOperarios = 0;
    int contadorLenhadores = 0;
    void debkillTrabalhador(string id);

    bool verificaFimJogo();
};
```

```
    float carvao = 10.0f;
    int eletricidade = 10;
    // ====================

    // Recolher recursos
    void producaoMinaF();
    void producaoMinaC();
    void producaoCentralE();
    void armazentamentoBateria();
    void producaoFundicao();
    void producaoEdX();
    void producaoMontanha();
    void producaoFloresta();
    void zonaZNZ();

    void addFerro(float quantidade);
    void addCarvao(float quantidade);
    void addEletricidade(int quantidade);
    void addAco(int quantidade);
    void addVigas(int quantidade);
    // ================================================

    // Edificios
    int configEdificio(string ed, int linha, int coluna);
    void ligarEdificio(int lin, int col);
    void desligarEdificio(int lin, int col);
    void upgradeEdificio(int lin, int col);
    void incrementarDiasEdificios();
    void vendeEdificio(int linha, int coluna);
    void desabarEdificio();
```

A classe Zona tal como foi dito antes contém funções que alteram maioritariamente o estado dos edifícios e trabalhadores (construção de edifícios, contratação de trabalhadores, etc).

```
class Zona {

protected:
    int linha;
    int coluna;
    string tipo;

    Edificio edificio;

public:
    Zona(){};
    Zona(string tipo, int lin, int col) : tipo(tipo), linha(lin), coluna(col) {}

    vector <Trabalhador> pointerTrabalhador;

    string getTipo();
    void zZNZ();

    // Edificios
    void construirEdificio(string tipo);
    void setTipoEdificio(string tipo);
    string getEdificio();
    void aumentarDiasEdificios();
    void desabamentoEdificio(int flag_pantano);
    int getArmazenamentoEdificio();
    void addArmazenamentoEdificio(int valor);
    int getPower();
    void setNivelEdificio();
    int getNivelEdificio();

    // Recursos (extra)
    int produzMontanha();
    int produzFloresta();

    // Trabalhadores
    void configTrabalhador(string tipo, Trabalhador trab);
    string getTrabalhadorTipo();
    string getTrabalhadorID(int nPosicao);
    int getNumTrabalhadores();
    void getTrabalhador();
    void incrementarDiasContrato();
    void despedirTrabalhadorDia(string zona);
    void removerTrabalhador(int posicao);
    Trabalhador copia(int posicao);

    bool trabalhadorMineiro();
    bool trabalhadorOperario();
    bool trabalhadorLenhador();
```

A classe Edifício é responsável pela criação de vários tipos de edifícios (através da herança) e que contém toda a informação sobre estes (tipo, armazenamento, contagem a partir do dia em que foi construído,…).

```
class Edificio {

public:
    string tipo;
    int nivel;
    int ligar;
    int armazenamento;
    int diasConstrucao;

    Edificio(){}
    Edificio(string type, int level, int power, int storage, int consDay) : tipo(type) , nivel(level), ligar(power), armazenamento(storage), diasC

    string getTipo();
    void setTipo(string nome);
    int getArmazenamento();
    void addArmazenamento(int a);
    void aumentarDiasConstrucao();
    int getDiasConstrucao();
```

```
public:
    MinaFerro(string t = "mnF", int level=1, int power=0, int storage=0, int consDay=1) : Edificio( type: t,level,power,storage,consDay) {};
};

class MinaCarvao : public Edificio{

public:
    MinaCarvao(string t = "mnC", int level=1, int power=0, int storage=0, int consDay=1) : Edificio( type: t,level,power,storage,consDay) {};
};

class CentralEletrica : public Edificio{

public:
    CentralEletrica(string t = "elec", int level=1, int power=0, int storage=0, int consDay=1) : Edificio( type: t,level,power,storage,consDay) {};
};

class Bateria : public Edificio{

public:
    Bateria(string t = "bat", int level=1, int power=0, int storage=0, int consDay=1) : Edificio( type: t,level,power,storage,consDay) {};
};
```

```
class Fundicao : public Edificio{

public:
    Fundicao(string t = "fun", int level=1, int power=0, int storage=0, int consDay=1) : Edificio( type: t,level,power,storage,consDay) {};
};

class EdificioX : public Edificio{

public:
    EdificioX(string t = "edX", int level=1, int power=0, int storage=0, int consDay=1) : Edificio( type: t,level,power,storage,consDay) {};
};
```

A classe Trabalhador é responsável pela criação de vários tipos de trabalhadores (através da herança) e que contém toda a informação sobre estes (tipo, id, dias de contrato,…).

```
class Trabalhador {


public:
    Trabalhador(){};
    Trabalhador(string id, float probability, int contDay) : id(id), probabilidade(probability), diasContrato(contDay) {};

    // Variáveis do trabalhador
    string id;
    string tipo;
    double probabilidade;
    int diasContrato;

    void setTipo(string nomeTipo);
    string getTipo();

    void setID(int a, int dia);
    string getID();
    void setIDcopia(string idCopia);

    void aumentarDiasContrato();
    int getDiasContrato();
```

```
class Mineiro : public Trabalhador{
public:
    Mineiro(string id, float probability, int contDay = 1) : Trabalhador( id: id,probability,contDay) {}
};

class Operario : public Trabalhador{
    public:
    Operario(string id, float probability, int contDay = 1) : Trabalhador( id: id,probability,contDay) {}
};

class Lenhador : public Trabalhador{
    public:
    Lenhador(string id, float probabilidade, int contDay = 1) : Trabalhador( id: id,probabilidade,contDay){}
};
```

A classe Engine é responsável pelo controle das fazes do dia e os seus efeitos(recolha de recursos ao anoitecer, aplicar as características e restrições de cada zona ao amanhecer e a permissão do uso de comandos que afetam a ilha a meio do dia) e a função que processa e executa comandos incluídos dentro de um ficheiro .txt.

```
class engine{

public:
    engine(){};
    ~engine(){};

    void amanhecer(ilha &i);
    void meioDia();
    void anoitecer(ilha &il);

    void processaFicheiro(string nome_ficheiro, ilha &i, int linhas, int colunas);
    int processaComandoFicheiro(istringstream &iss, ilha &i, int linhas, int colunas);
};
```

O utils.h contém funções do tipo random usadas em situações de probabilidades e geração de números aleatórios.

```
#include <cstdlib>
#include <ctime>

int random(int max);
int randomEntreDois(int min, int max);
bool probabilidades(float prob);
```

Edifícios:

Foram implementados os edifícios Mina de Ferro, Mina de Carvão, Central Elétrica, Bateria, Fundição e Edifício-X.

Para o Edifício-X optamos por utilizar um edifício equivalente a um edifício de Serração em que 2kg de madeira dão origem a 1 viga. Além disso este edifício apenas funciona se tiver um lenhador na zona e a cada nível aumenta a produção de vigas.

Zonas:

Foram implementadas as zonas de Deserto, Pastagem, Floresta, Montanha, Pantano e ZonaX. A ZonaX é uma praia. Os trabalhadores são atraídos para este local devido à sua beleza natural. Por ficarem tão felizes, a sua produtividade é maior e a produção de recursos aumenta o dobro, porém a probabilidade de acontecer um tsunami é de 65%. Quando isto acontece o edifício que esteja presente nesta zona é destruído.

Comandos Implementados:

Quase todos os comandos pedidos foram implementados, alguns deles com necessidade de algum melhoramento. Não foram implementados o comando save, load e apaga.
Além disso adicionámos ainda o comando Upgrade que permite a subida de nível dos edifícios.

Conclusão:

Com a realização deste trabalho conseguimos aprofundar a matéria da disciplina e apesar de ainda necessitarmos de melhorar nalguns aspetos foi bastante importante para melhorar as nossas capacidades em C++.