

## Trabalho Prático

### Disciplina Programação Orientada a Objetos

Autores:

Daniel Duarte Dias Ferreira Albino – 2020134077

Nuno Alexandre Domingues – 2020109910

Engenharia Informática

Coimbra, janeiro de 2022

## Índice

1. Introdução .....	3
2. Classes .....	4
2.1 Zonas:.....	4
2.1.1 Deserto: .....	4
2.1.2 Pastagem .....	5
2.1.3 Floresta .....	6
2.1.4 Montanha .....	7
2.1.5 Pantano: .....	8
2.1.6 ZonaX .....	8
2.2 Edifícios .....	9
2.2.1 Mina de Ferro .....	10
2.2.2 Mina de Carvão .....	10
2.2.3 Central Elétrica .....	11
2.2.4 Bateria .....	12
2.2.5 Fundação .....	13
2.2.6 EdifícioX .....	13
2.3 Trabalhadores .....	14
2.3.1 Operário .....	15
2.3.2 Mineiro .....	15
2.3.3 Lenhador .....	16
2.4 Recursos .....	17
2.4.1 Dinheiro.....	17
2.4.2 Ferro.....	18
2.4.3 Barras de Aço .....	18
2.4.4 Carvão .....	19
2.4.5 Madeira .....	19
2.4.6 Vigas de Madeira.....	20
2.4.7 Eletricidade .....	20
2.5 Mapa .....	21
2.6. Dia .....	23
2.7 Infolha .....	23
3. Jogo.....	24
3.1 Aspeto do Jogo .....	24
4. Conclusões Finais .....	27

## Índice de Figuras

Figura 1 - Classe Zona.....	4
Figura 2 - Classe Deserto .....	5
Figura 3 - Classe de Pastagem .....	5
Figura 4 - Função que gera aleatoriamente o número de árvores no início do jogo.....	6
Figura 5 - Classe Floresta.....	7
Figura 6 - Classe Montanha.....	7
Figura 7 - Classe Pantano .....	8
Figura 8 - Classe Zona X.....	9
Figura 9 - Classe Edifícios .....	9
Figura 10 - Classe Mina de Ferro.....	10
Figura 11 - Classe Mina de Carvão .....	11
Figura 12 - Classe Central Elétrica .....	12
Figura 13 - Classe Bateria .....	12
Figura 14 - Classe Fundação .....	13
Figura 15 - Classe EdifícioX.....	14
Figura 16 - Classe Trabalhadores .....	14
Figura 17 - Classe Operário .....	15
Figura 18 - Classe Mineiro .....	16
Figura 19 - Classe Lenhador .....	16
Figura 20 - Classe Recursos .....	17
Figura 21 - Classe Dinheiro.....	17
Figura 22 - Classe Ferro .....	18
Figura 23 - Classe Barras de Aço .....	18
Figura 24 - Classe Carvão.....	19
Figura 25 - Classe Madeira .....	19
Figura 26 - Classe Vigas de Madeira.....	20
Figura 27 - Classe Eletricidade.....	20
Figura 28 - Classe Mapa .....	21
Figura 29 - Classe Dia .....	23
Figura 30 - Classe Infolha .....	23
Figura 31 - Menu Inicial do Jogo .....	24
Figura 32 - Início do Jogo.....	24
Figura 33 - Construção de um edifício .....	25
Figura 34 - Outros Comandos.....	25
Figura 35 - Comando List.....	26
Figura 36 - Exemplo de como obter recursos .....	26

## 1. Introdução

Este trabalho foi realizado no âmbito da unidade curricular de Programação Orientada a Objetos, do curso de Licenciatura Informática, do Instituto Superior de Engenharia de Coimbra.

O mesmo tem como objetivo aplicar conhecimentos sobre a linguagem C++, no âmbito de um trabalho prático da unidade curricular acima referida.

O trabalho em questão destina-se a desenvolver um simulador/jogo (single-player) de construção e desenvolvimento, onde será atribuído ao jogador a concessão de uma ilha e este deve desenvolver essa ilha, industrializando-a e construindo todo um complexo fabril.

Este relatório divide-se em três partes fundamentais: a primeira parte que corresponde á descrição das classes usadas; a segunda parte que consiste na apresentação da interface do jogo e da execução de alguns comandos e a terceira e última parte em que abordamos umas conclusões finais.

## 2. Classes

### 2.1 Zonas:

A classe Zona é a classe de onde derivam as restantes zonas como: o Deserto, a Floresta, o Pantano, a Pastagem, a Montanha e a ZonaX.

```
class Zona {  
protected:  
    std::string tipo;  
public:  
    Zona(std::string t) : tipo(t){};  
    const std::string &getZona() const {return tipo;}  
    virtual int getArvores();  
    virtual void removeArvores(int q){ removeArvores(q);}  
    virtual void adicionaArvores(int q){ adicionaArvores(q);}  
    virtual int getQuantidade(){return getQuantidade();}  
    virtual void adicionaQuantidade(int q) { adicionaQuantidade(q);}  
    virtual void removeQuantidade(int q){removeQuantidade(q);}  
    virtual float getProducao() const {return getProducao();}  
    virtual bool isDemissao() const {return isDemissao();}  
    virtual int getLimiteArvores() const {return getLimiteArvores();}  
    virtual bool getConstrucao() const {return getConstrucao();}  
    virtual void addConstrucao() {addConstrucao();}  
    virtual void removeConstrucao() {removeConstrucao();}  
  
    virtual ~Zona() = default;  
};
```

Figura 1 - Classe Zona

#### 2.1.1 Deserto:

Na figura 2, podemos observar a classe Deserto que representa a zona Deserto da ilha. Esta apresenta várias características como:

- A produção das minas cai para 50%;
- Permite a construção de qualquer edifício (a variável construção define se existe uma construção na zona e nada tem haver com a permissão de construir edifícios);
- Pode existir a demissão dos trabalhadores.

```
class Deserto : public Zona{
    bool demissao;
    float producao;
    bool construcao = false;
public:
    Deserto(bool d = true, float p = 0.5) : Zona(t: "dsr"), demissao(d), producao(p){}
    bool isDemissao() const override{return demissao;}
    float getProducao() const override{return producao;}
    bool getConstrucao() const override{return construcao;}
    void addConstrucao() override;
    void removeConstrucao() override;

    ~Deserto() = default;
};
```

Função que retorna se nesta zona existe demissão dos trabalhadores.

Função que retorna a produção nesta zona.

Função que retorna se existe um edifício construído nesta zona.

Função que adiciona uma construção á zona.

Função que remove uma construção á zona.

Figura 2 - Classe Deserto

## 2.1.2 Pastagem

Na seguinte imagem, está representada a classe Pastagem que representa respetivamente a zona Pastagem da ilha. Esta também tem as suas próprias características como:

- Pode existir a demissão dos trabalhadores;
- Permite a construção de qualquer edifício (a variável construção define se existe uma construção na zona e nada tem haver com a permissão de construir edifícios);
- A produção de cada edifício é o seu normal.

```
class Pastagem : public Zona{
    bool demissao;
    float producao;
    bool construcao = false;
public:
    Pastagem(bool d = false, float p = 1) : Zona(t: "pas"), demissao(d), producao(p){}
    bool isDemissao() const override{return demissao;}
    float getProducao() const override{return producao;}
    bool getConstrucao() const override{return construcao;}
    void addConstrucao() override;
    void removeConstrucao() override;

    ~Pastagem() = default;
};
```

Figura 3 - Classe de Pastagem

### 2.1.3 Floresta

Na figura 5, podemos ver a classe Floresta que representa a zona Floresta da ilha. Esta tem diversas características como:

- Nesta zona crescem árvores (duas a cada 2 dias caso não exista nenhum edifício construído, ou seja, caso a variável construção esteja a *false*) até um certo limite, neste caso, esse limite é 100 árvores;
- O número de árvores que existem no início do jogo é gerado aleatoriamente pela seguinte função:

```
int getNArvores(){  
    static default_random_engine e( s: time( timer: 0 ));  
    static uniform_int_distribution<int> d( a: 20, b: 40 );  
    return d( &: e );  
}
```

Figura 4 - Função que gera aleatoriamente o número de árvores no início do jogo

- Permite a construção de qualquer edifício. Contudo e como foi dito anteriormente não crescem mais árvores e morre uma a cada dia;
- Os trabalhadores não pedem a demissão nesta zona;
- A produção de cada edifício é o seu normal.

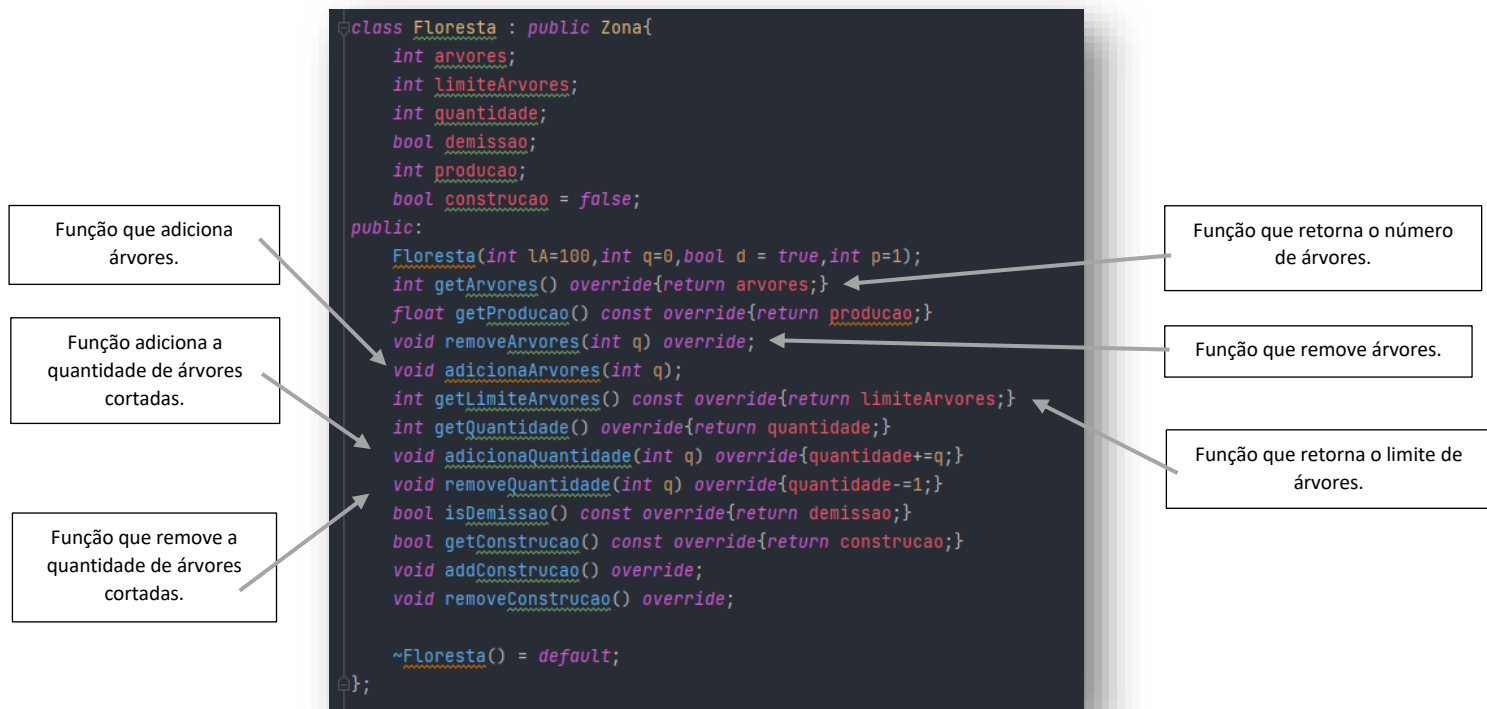


Figura 5 - Classe Floresta

## 2.1.4 Montanha

Na figura 6, está representada a classe Montanha que representa respetivamente a zona Montanha da ilha. Esta apresenta inúmeras características como:

- Pode existir a demissão dos trabalhadores;
- Permite a construção de qualquer edifício (a variável construção define se existe uma construção na zona e nada tem haver com a permissão de construir edifícios);
- Esta zona tem a particularidade de a cada dia os trabalhadores apanham 0.1 kg de ferro;
- A produção de cada edifício é o dobro.

```

class Montanha : public Zona{
    bool demissao;
    float producao;
    bool construcao = false;
public:
    Montanha(float p = 2, bool d = true) : Zona("mnt", producao(p), demissao(d){}
    float getProducao() const {return producao;}
    bool isDemissao() const override{return demissao;}
    bool getConstrucao() const override{return construcao;}
    void addConstrucao() override;
    void removeConstrucao() override;

    ~Montanha() = default;
};
    
```

Figura 6 - Classe Montanha



### 2.1.5 Pantano:

Na figura 7, podemos observar a classe Pantano que representa a zona Floresta da ilha. Esta tem diversas características como:

- Pode existir a demissão dos trabalhadores;
- Permite a construção de qualquer edifício (a variável construção define se existe uma construção na zona e nada tem haver com a permissão de construir edifícios);
- Nesta zona os edifícios desabam depois do 10 dia construídos;
- A produção de cada edifício é o seu normal.

```
class Pantano : public Zona{
    float producao;
    bool demissao;
    bool construcacao = false;
public:
    Pantano(float p = 1, float d = true) : Zona("pnt", producao(p), demissao(d)){}
    float getProducao() const {return producao;}
    bool isDemissao() const override {return demissao;}
    bool getConstrucao() const override {return construcacao;}
    void addConstrucao() override;
    void removeConstrucao() override;

    ~Pantano() = default;
};
```

Figura 7 - Classe Pantano

### 2.1.6 ZonaX

Na figura 8, está representada a classe ZonaX que representa respetivamente a zona ZonaX da ilha. Esta também tem as suas próprias características como:

- Pode existir a demissão dos trabalhadores;
- Permite a construção de qualquer edifício (a variável construção define se existe uma construção na zona e nada tem haver com a permissão de construir edifícios);
- Esta zona tem a particularidade de a cada dia os trabalhadores apanham 0.2 kg de vigas de madeira;
- A produção de cada edifício é o dobro.

```
class ZonaX : public Zona{
    float producao;
    bool demissao;
    bool construcao = false;
public:
    ZonaX(float p = 1, float d = true) : Zona("znz"), producao(p), demissao(d) {};
    float getProducao() const {return producao;}
    bool isDemissao() const override {return demissao;}
    bool getConstrucao() const override {return construcao;}
    void addConstrucao() override;
    void removeConstrucao() override;

    ~ZonaX() = default;
};
```

Figura 8 - Classe Zona X

**Nota:** Nesta parte do trabalho, foi necessário utilizar o polimorfismo, visto que cada zona tem a sua particularidade e comportamentos diferentes.

## 2.2 Edifícios

A classe Edifícios é a classe de onde derivam os restantes edifícios como: o Mina de Ferro, a Mina de Carvão, a Central Elétrica, a Bateria, a Fundação e o EdifícioX.

```
class Edificios {
    std::string tipo;
    bool ligado = false;
    std::string produz;
    int diasConstruido;
public:
    Edificios(std::string tipo, std::string p, int d=0);

    const std::string &getEdificio() const;

    void DesEdificio();
    void LigaEdificio();
    const std::string &getProdut() const {return produz;}

    virtual int getPreco() const {return getPreco();}
    virtual int getPrecoVigas() const {return getPrecoVigas();}
    virtual int getProducao() const {return getProducao();}
    virtual bool adicionaArmazem(int x) {return adicionaArmazem(x);}
    virtual void removeArmazem(int x);
    virtual int getArmazem() const {return getArmazem();}
    virtual float getDesabar() const {return getDesabar();}
    virtual bool addNivel() {return addNivel();}
    virtual int getNivel() const {return getNivel();}
    virtual int getPrecoNivel() const {return getPrecoNivel();}
    virtual void addArmazenamento() {addArmazenamento();}

    int getDiasConstruido() const {return diasConstruido;}
    void addDiasConstruido() {diasConstruido+=1;}
    bool isLigado();

    virtual ~Edificios() = default;
};
```

Figura 9 - Classe Edifícios

## 2.2.1 Mina de Ferro

Na figura 10, podemos ver a Mina de Ferro que representa a zona Mina de Ferro da ilha. Esta apresenta diversas características como:

- Preço em dinheiro, neste caso a cada viga de madeira custa 10€, ou seja, 100€ no total;
- Preço em vigas de madeira (10 vigas de madeira);
  - Produz 2 kg de ferro por dia caso tenha um mineiro na zona (exceto no deserto que passa para metade);
- Tem 15% de desabar;
- Tem a capacidade de armazenar 100 kg de ferro no nível 1 (vai até ao nível 5 e a cada nível é incrementado 10kg á sua capacidade máxima);
- Preço de cada nível em dinheiro é de 15€ e mais 1 viga de madeira.

```
class MinaFerro : public Edificios{
    int preco;
    int precoVigas;
    int producao;
    float desabar;
    int armazena;
    int armazem;
    int nivel;
    int precoLevelUp; //+1 viga
public:
    MinaFerro(int pre = 100,int preV = 10,int pro = 2,float d= 0.15,int a = 100,int n=1,int arm = 0,int pl = 15)
        : Edificios( tipo: "mnF", pl: "ferro"),preco(pre),precoVigas(preV),producao(pro),desabar(d),armazena(a),nivel(n), armazem(arm),precoLevelUp(pl){}

    int getPreco() const override{return preco;}
    int getPrecoVigas() const override {return precoVigas;}
    int getProducao() const override;
    bool adicionaArmazem(int x) override;
    void removeArmazem(int x) override {armazem-=x;}
    float getDesabar() const override{return desabar;}
    int getArmazem() const override{return armazem;}
    int getNivel() const override{return nivel;}
    int getPrecoNivel() const override {return precoLevelUp;}
    bool addNivel() override;
    void addArmazenamento() override {armazena+=10;}
    ~MinaFerro() = default;
};
```

Função que  
retorna o preço  
em €.

Função que  
retorna a  
produção.

Função que  
adiciona ferro ao  
armazém.

Função que retorna  
o preço em vigas de  
madeira.

Função que remove  
ferro ao armazém.

Figura 10 - Classe Mina de Ferro

## 2.2.2 Mina de Carvão

Na figura 11, está representada a classe Mina de Carvão que representa respetivamente a zona Floresta. Esta apresenta inúmeras características como:

- Preço em dinheiro, neste caso a cada viga de madeira custa 10€, ou seja, 100€ no total;
- Preço em vigas de madeira (10 vigas de madeira);

- Produz 2 kg de carvão por dia caso tenho um mineiro na zona (exceto no deserto que passa para metade);
- Tem 10% de desabar;
- Tem a capacidade de armazenar 100 kg de carvão no nível 1 (vai até ao nível 5 e a cada nível é incrementado 10kg á sua capacidade máxima);
- Preço de cada nível em dinheiro é de 15€ e mais 1 viga de madeira.

```
class MinaCarvao : public Edificios{
    int preco; //preço em dinheiro
    int precoVigas;
    int producao;
    float desabar;
    int armazena;
    int armazen;
    int nivel;
    int precoLevelUp; //+1 viga
public:
    MinaCarvao(int pre = 100,int preV = 10,int pro = 2,float d= 0.10,int a = 100,int n=1,int arm = 0,int pl = 10)
        : Edificios( tipo: "mnc", p: "carvao"),preco(pre),precoVigas(preV),producao(pro),desabar(d),armazena(a),nivel(n),armazen(arm),precoLevelUp(pl){}
    int getPreco() const override{return preco;}
    int getPrecoVigas() const override {return precoVigas;}
    int getProducao() const override;
    bool adicionaArmazen(int x) override;
    void removeArmazen(int x) override {armazen-=x;}
    float getDesabar() const override{return desabar;}
    int getArmazen() const override{return armazen;}
    int getNivel() const override{return nivel;}
    int getPrecoNivel() const override {return precoLevelUp;}
    bool addNivel() override;
    void addArmazenamento() override {armazena+=10;}
    ~MinaCarvao() = default;
};
```

Função que  
retorna o nível.

Função que  
retorna o  
preço por  
nível.

Função que retorna  
a percentagem de  
desabar.

Função que  
retorna  
quantidade no  
armazém.

Figura 11 - Classe Mina de Carvão

### 2.2.3 Central Elétrica

Na figura 12, podemos ver a classe Central Elétrica que representa a zona Central Elétrica da ilha. Esta tem diversas características como:

- Preço em dinheiro é 15€;
- Produz 1 kg de carvão e 1 KWh de eletricidade por dia caso tenha um operário na zona e tenha uma zona do tipo floresta e um edifício do tipo bateria adjacente;
- Tem 0% de desabar;
- Tem a capacidade de armazenar 100 kg de carvão no nível 1 (vai até ao nível 5 e a cada nível é incrementado 10kg á sua capacidade máxima);
- Preço de cada nível em dinheiro é de 20€.

```

class CentralEletrica : public Edificios{
    int preco;
    int producao;
    int armazenagem;
    int nivel;
    int precoLevelUp;
public:
    CentralEletrica(int pre = 15, int pro = 2, int a = 100, int n=1, int arm = 0, int pl = 20)
        : Edificios( tipo: "elec", p: "eletricidade carvao"), preco(pre), producao(pro), armazenagem(a), nivel(n), armazenagem(arm), precoLevelUp(pl){}
    int getPreco() const override {return preco;}
    int getProducao() const override {return producao;}
    bool adicionaArmazem(int x) override;
    void removeArmazem(int x) override {armazem-=x;}
    int getArmazem() const override {return armazenagem;}
    int getNivel() const override {return nivel;}
    int getPrecoNivel() const override {return precoLevelUp;}
    bool addNivel() override;
    void addArmazenamento() override {armazem+=10;}
    ~CentralEletrica() = default;
};

```

Função que adiciona 1 nível ao edifício.

Função que adiciona 10 de capacidade ao armazém.

Figura 12 - Classe Central Elétrica

## 2.2.4 Bateria

Na figura 13, podemos observar a classe Bateria que representa a zona Bateria da ilha. A mesma tem várias características como:

- Preço em dinheiro é 15€ + 10 vigas de madeira;
- Armazena 1 KWh de eletricidade por dia caso tenha uma central elétrica adjacente;
- Tem 0% de desabar;
- Tem a capacidade de armazenar 100 KWh de eletricidade no nível 1 (vai até ao nível 5 e a cada nível é incrementado 10kg á sua capacidade máxima);
- Preço de cada nível em dinheiro é de 15€.

```

class Bateria : public Edificios{
    int preco; //+10 vigas
    int armazenagem;
    int nivel;
    int precoLevelUp;
public:
    Bateria(int pre = 10, int a = 100, int n=1, int arm = 0, int pl = 15)
        : Edificios( tipo: "bat", p: "NA"), preco(pre), armazenagem(a), nivel(n), armazenagem(arm), precoLevelUp(pl){}
    int getPreco() const override {return preco;}
    bool adicionaArmazem(int x) override;
    void removeArmazem(int x) override {armazem-=x;}
    int getArmazem() const override {return armazenagem;}
    int getNivel() const override {return nivel;}
    int getPrecoNivel() const override {return precoLevelUp;}
    bool addNivel() override;
    void addArmazenamento() override {armazem+=10;}
    ~Bateria() = default;
};

```

Figura 13 - Classe Bateria

## 2.2.5 Fundição

Na imagem seguinte, está representada a classe Fundição que representa a zona Fundição da ilha. A mesma apresenta imensas características como:

- Preço em dinheiro é 10€;
- Produz 1 barra de ferro por dia caso tenha um operário na zona e tenha um edifício adjacente do tipo mina de ferro e mina de carvão/central elétrica;
- Tem 10% de desabar;
- Tem a capacidade de armazenar 100 kg de barras de ferro no nível 1 (vai até ao nível 5 e a cada nível é incrementado 10kg á sua capacidade máxima);
- Preço de cada nível em dinheiro é de 15€.

```
class Fundicao : public Edificios{
    int preco;
    int producao;
    int armazena;
    int armazem;
    int nivel;
    int precoLevelUp;
public:
    Fundicao(int pre = 10,int pro = 1,float d= 0.10,int a = 100,int n=1,int arm = 0,int pl = 25)
        : Edificios( tipos: "fun", p: "barras de aço"),preco(pre),producao(pro),armazena(a),nivel(n),armazem(arm),precoLevelUp(pl){}
    int getPreco() const override{return preco;}
    int getProducao() const override {return producao;}
    bool adicionaArmazem(int x) override;
    void removeArmazem(int x) override {armazem-=x;}
    int getArmazem() const override{return armazem;}
    int getNivel() const override{return nivel;}
    int getPrecoNivel() const override {return precoLevelUp;}
    bool addNivel() override;
    void addArmazenamento() override {armazena+=10;}

    ~Fundicao() = default;
};
```

Figura 14 - Classe Fundição

## 2.2.6 EdifícioX

Na figura 15, podemos observar a classe EdifícioX que representa a zona EdifícioX. Esta tem inúmeras características como:

- Preço em dinheiro é 20€;
- Produz 1 vigas de madeira por dia caso tenha um operário na zona e tenha uma zona adjacente do tipo floresta;
- Tem 0% de desabar;

- Tem a capacidade de armazenar 100 kg de vigas de madeira no nível 1 (vai até ao nível 5 e a cada nível é incrementado 10kg á sua capacidade máxima);
- Preço de cada nível em dinheiro é de 10€;

```
class EdificioX : public Edificios{
    int preco;
    int producao;
    int armazen;
    int armazena;
    int nivel;
    int precoLevelUp;
public:
    EdificioX(int pre = 20, int pro = 1, int a = 50, int n=1, int arm = 0, int pl = 10)
        : Edificios( tipo: "EdX", pr: "vigas de madeira"), preco(pre), producao(pro), armazena(a), nivel(n), armazen(arm), precoLevelUp(pl){}
    int getPreco() const override{return preco;}
    int getProducao() const override{return producao;}
    bool adicionaArmazen(int x) override;
    void removeArmazen(int x) override {armazen-=x;}
    int getArmazen() const override{return armazen;}
    int getNivel() const override{return nivel;}
    int getPrecoNivel() const override{return precoLevelUp;}
    bool addNivel() override;
    void addArmazenamento() override {armazena+=10;}

    ~EdificioX() = default;
};
```

Figura 15 - Classe EdificioX

**Nota:** Nesta parte do trabalho, foi necessário utilizar o polimorfismo, visto que cada edifício tem a sua particularidade e comportamentos diferentes.

## 2.3 Trabalhadores

A classe Trabalhadores é a classe de onde derivam os restantes trabalhadores como o Operário, Mineiro e Lenhador.

```
class Trabalhadores {
    std::string tipo;
    static unsigned int n;
    unsigned int numero;
    std::string id;
    int DiasTrabalhar;
public:
    Trabalhadores(std::string t, Dia d);
    const std::string &getTipo() const;
    const std::string &getId() const;
    virtual int getPreco() const;
    virtual float getCansar() const{return getCansar();}
    virtual int diaDesp() const{return diaDesp();}
    bool depedimento(int d);
    int diasTrabalhar() const{return DiasTrabalhar;}
    void adicionaDiasTrabalho();
    virtual void DiasTrabalhoLen() {DiasTrabalhoLen();}
    virtual void setCansar(float x) {setCansar(x);}
    virtual bool diaDescanso() {return diaDescanso();}
    Trabalhadores operator++();

    virtual ~Trabalhadores() = default;
};
```

Função que retorna o id do trabalhador.

O id é formado pelo dia e pelo número de trabalhadores contratados (ou seja, no dia 1 e o 1º trabalhador a ser contratado fica com o id de 1.1).

Função que retorna o tipo do trabalhador.

Figura 16 - Classe Trabalhadores

### 2.3.1 Operário

Na seguinte figura, podemos ver a classe Operário que representa o trabalhador operário. Este tem algumas características como:

- Preço de contratação de 15€;
- A probabilidade de se demitir que é de 5%;
- Pode-se despedir a partir do 10º dia.

```
class Operario : public Trabalhadores{
    int preco;
    float cansar;
    int diaDespedimento;
public:
    Operario(Dia d, int p = 15, float c = 0.05, int diaD = 10) : Trabalhadores("oper", d, preco(p), cansar(c), diaDespedimento(diaD) {});
    void setCansar(float c) override {Operario::cansar += c;}
    int getPreco() const override {return preco;}
    float getCansar() const override {return cansar;}
    int diaDesp() const override {return diaDespedimento;}

    ~Operario() = default;
};
```

Figura 17 - Classe Operário

Função que adiciona mais probabilidade de o trabalhador se despedir, por exemplo na zona montanha o trabalhador tem mais 5% de probabilidade de se despedir.

Função que retorna os dias de despedimento.

Função que retorna a probabilidade de o trabalhador se despedir.

Função que retorna o preço do trabalhador.

### 2.3.2 Mineiro

Na figura 18, está representada a classe Mineiro que representa o trabalhador mineiro. As suas características prendem-se com:

- Preço de contratação é de 15€;
- Tem 10% de probabilidade de se despedir;
- Ao fim de 2 dias pode pedir a demissão.



```
class Mineiro : public Trabalhadores{
    int preco;
    float cansar;
    int diaDespedimento;
public:
    Mineiro(Dia d,int p = 15,float c = 0.10,int diaD = 2) : Trabalhadores(t "miner",d), preco(p), cansar(c), diaDespedimento(diaD) {};
    void setCansar(float c) override {Mineiro::cansar += c;}
    int getPreco() const override{return preco;}
    float getCansar() const override {return cansar;}
    int diaDesp() const override {return diaDespedimento;}

    ~Mineiro() = default;
};
```

Figura 18 - Classe Mineiro

### 2.3.3 Lenhador

Na figura 19, podemos observar a classe Lenhador que representa o trabalhador lenhador. Este apresenta as seguintes características:

- Preço de contratação é de 15€;
- Trabalha 4 dias e descansa 1 e nunca se despede, exceto se estiver num pântano;

```
class Lenhador : public Trabalhadores{
    int preco;
    int diasTrabalho;
    float cansar;
    bool Descanso;
    int diaDespedimento;
public:
    Lenhador(Dia d,int p = 15,int dT = 4,int c=0,int diaD = 0) : Trabalhadores(t "Len",d), preco(p), diasTrabalho(dT),cansar(c),diaDespedimento(diaD),Descanso(false) {};
    void setCansar(float c) override {Lenhador::cansar += c;}
    int getPreco() const override{return preco;}
    float getCansar() const override {return cansar;}
    bool diaDescanso() override;
    void DiasTrabalhoLen() override;
    int diaDesp() const override {return diaDespedimento;}

    ~Lenhador() = default;
};
```

Figura 19 - Classe Lenhador

Função que retorna verdadeiro caso o lenhador esteja a descansar e falso caso contrário.

- Corta 1 árvore por dia.

**Nota:** Nesta parte do trabalho, foi necessário utilizar o polimorfismo, visto que cada trabalhador tem a sua particularidade e comportamentos diferentes.

## 2.4 Recursos

A classe Recursos é a classe de onde derivam os restantes recursos como: o Dinheiro, o Ferro, as Barras de Aço, o Carvão, a Madeira, as Vigas de Madeira e a Eletricidade.

```
class Recursos {
    std::string tipo;
public:
    Recursos(std::string t):tipo(t){}
    virtual std::string getAsString() const;
    const std::string &getTipo() const {return tipo;}
    virtual float getQuantidade() const;
    virtual void removeQuantidade(float q);
    virtual float getPreco() const;
    virtual void adicionaQuantidade(float d) {adicionaQuantidade(d);}

    virtual ~Recursos() = default;
};
```

Função que retorna uma string com as informações dos recursos.

Função que retorna a quantidade de um determinado recurso.

Função que retorna o preço de cada recurso.

Função que retorna o tipo de recurso.

Função que remove a quantidade de um determinado recurso.

Função que adiciona a quantidade de um determinado recurso.

Figura 20 - Classe Recursos

### 2.4.1 Dinheiro

Na imagem seguinte, podemos ver a classe Dinheiro que representa o recurso dinheiro. É uma classe mais simples onde a única coisa que é representada é a quantidade de dinheiro que o jogador tem disponível. Não sendo possível vender.

```
class Dinheiro : public Recursos{
    float quantidade;
public:
    Dinheiro(int q = 2000): Recursos(t "dinheiro", quantidade(q)){
        std::string getAsString() const override;
        float getQuantidade() const override {return quantidade;}
        void removeQuantidade(float q) override;
        void adicionaQuantidade(float d) override {quantidade += d;}

        ~Dinheiro() = default;
};
```

Figura 21 - Classe Dinheiro

## 2.4.2 Ferro

Na figura 22, está representada a classe Ferro que representa o recurso ferro e que tem as seguintes características:

- A quantidade geral da ilha, ou seja, a soma de todo o ferro gerado pelas minas de ferro;
- O seu preço que é de 1€ por cada Ferro.

```
class Ferro : public Recursos{
    float preco;
    float quantidade;
public:
    Ferro(float p = 1,int q=0) : Recursos(t: "ferro"), preco(p), quantidade(q){}
    std::string getAsString() const override;
    float getPreco() const override{return preco;}
    float getQuantidade() const override{return quantidade;}
    void adicionaQuantidade(float d) override {quantidade += d;}
    void removeQuantidade(float q) override;
    Ferro &operator+=(int q);

    ~Ferro() = default;
};
```

Figura 22 - Classe Ferro

## 2.4.3 Barras de Aço

Na seguinte figura, podemos observar a classe Barras que representa o recurso barras de aço. A mesma apresenta inúmeras características como:

- A quantidade geral da ilha, ou seja, a soma de todas as barras de ferro geradas pelas fundições;
- O seu preço que é de 2€ por cada Barra.

```
class Barras : public Recursos{
    float preco;
    float quantidade;
public:
    Barras(float p = 2,int q=2) : Recursos(t: "barras de aço"),preco(p), quantidade(q){}
    std::string getAsString() const override;
    float getPreco() const override{return preco;};
    float getQuantidade() const override{return quantidade;}
    void adicionaQuantidade(float d) override {quantidade += d;}
    void removeQuantidade(float q) override;
    Barras &operator+=(int q);

    ~Barras() = default;
};
```

Figura 23 - Classe Barras de Aço

## 2.4.4 Carvão

Na imagem 24, podemos ver a classe Carvão que representa o recurso carvão. Esta tem as características a seguir referidas:

- A quantidade geral da ilha, ou seja, a soma de todo o carvão gerado pelas minas de carvão;
- O seu preço que é de 1€ por cada Carvão.

```
class Carvao : public Recursos{
    float preco;
    float quantidade;
public:
    Carvao(float p = 1,int q=0) : Recursos(t "carvao"),preco(p), quantidade(q){}
    std::string getAsString() const override;
    float getPreco() const override{return preco;};
    float getQuantidade() const override{return quantidade;};
    void adicionaQuantidade(float d) override {quantidade += d;};
    void removeQuantidade(float q) override;
    Carvao &operator+=(int q);

    ~Carvao() = default;
};
```

Figura 24 - Classe Carvão

## 2.4.5 Madeira

Na figura 25, está representada a classe Madeira que representa o recurso madeira e que tem as seguintes características:

- A quantidade geral da ilha, ou seja, a soma de toda a madeira cortada pelos lenhadores;
- O seu preço que é de 1€ por cada Madeira.

```
class Madeira : public Recursos{
    float preco;
    float quantidade;
public:
    Madeira(float p = 1,int q=0) : Recursos(t "madeira"),preco(p), quantidade(q){}
    std::string getAsString() const override;
    float getPreco() const override{return preco;};
    float getQuantidade() const override{return quantidade;};
    void adicionaQuantidade(float d) override {quantidade += d;};
    void removeQuantidade(float q) override;
    Madeira &operator+=(int q);

    ~Madeira()= default;
};
```

Figura 25 - Classe Madeira

## 2.4.6 Vigas de Madeira

Na figura que se segue, podemos observar a classe Vigas que representa o recurso vigas de madeira. A mesma é composta pelas seguintes características:

- A quantidade geral da ilha, ou seja, a soma de todas as vigas de madeira gerada pelo edifício EdifícioX;
- O seu preço que é de 2€ por cada Viga de Madeira.

```
class Vigas : public Recursos{
    float preco;
    float quantidade;
public:
    Vigas(float p = 2,int q=20) : Recursos("vigas de madeira"),preco(p), quantidade(q){}
    std::string getAsString() const override;
    float getPreco() const override{return preco;};
    float getQuantidade() const override{return quantidade;};
    void adicionaQuantidade(float d) override {quantidade += d;};
    void removeQuantidade(float q) override;
    Vigas &operator+=(int q);

    ~Vigas() = default;
};
```

Figura 26 - Classe Vigas de Madeira

## 2.4.7 Eletricidade

Na imagem 27, podemos ver a classe Eletricidade que representa o recurso eletricidade. Esta tem várias características como:

- A quantidade geral da ilha, ou seja, a soma de toda a eletricidade gerada pelo edifício Central elétrica e armazenada na bateria;
- O seu preço que é de 1.5€ por cada KWh de eletricidade.

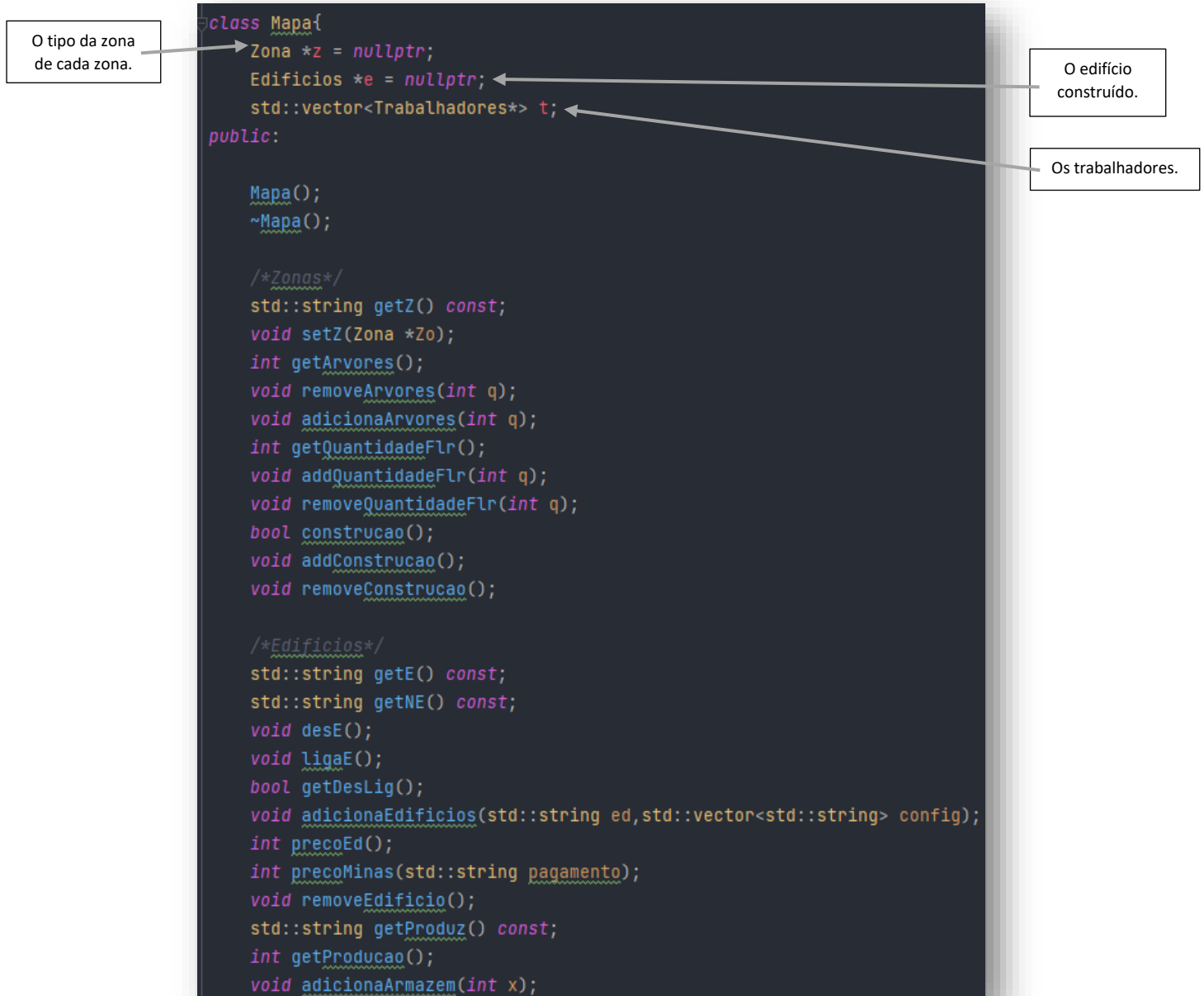
```
class Eletricidade : public Recursos{
    float preco;
    float quantidade;
public:
    Eletricidade(float p = 1.5,int q=0) : Recursos("eletricidade"),preco(p), quantidade(q){}
    std::string getAsString() const override;
    float getPreco() const override{return preco;};
    float getQuantidade() const override{return quantidade;};
    void adicionaQuantidade(float d) override {quantidade += d;};
    void removeQuantidade(float q) override;
    Eletricidade &operator+=(int q);

    ~Eletricidade() = default;
};
```

Figura 27 - Classe Eletricidade

## 2.5 Mapa

Esta classe representa a ilha, ou seja, é nesta classe onde são definidas o tipo de cada zona, o edifício que está construído em cada zona e os trabalhadores de cada zona.



```
class Mapa{
    Zona *z = nullptr;
    Edificios *e = nullptr;
    std::vector<Trabalhadores*> t;
public:

    Mapa();
    ~Mapa();

    /*Zonas*/
    std::string getZ() const;
    void setZ(Zona *Zo);
    int getArvores();
    void removeArvores(int q);
    void adicionaArvores(int q);
    int getQuantidadeFlr();
    void addQuantidadeFlr(int q);
    void removeQuantidadeFlr(int q);
    bool construcao();
    void addConstrucao();
    void removeConstrucao();

    /*Edificios*/
    std::string getE() const;
    std::string getNE() const;
    void desE();
    void ligaE();
    bool getDesLig();
    void adicionaEdificios(std::string ed, std::vector<std::string> config);
    int precoEd();
    int precoMinas(std::string pagamento);
    void removeEdificio();
    std::string getProduz() const;
    int getProducao();
    void adicionaArmazem(int x);
}
```

Figura 28 - Classe Mapa

```

void removeArmazem(int x);
int getDiasConstruido();
void addDiasEdificioConstruido();
void zonaPnt(std::vector<Recursos*> recursos);
int getNivel();
int getPrecoLevelUp();
bool LevelUp(std::vector<Recursos*>recursos, InfoIlha II, Mapa **i);
float getDesabar();
void Desabar(std::vector<Recursos*> recursos);

/*Trabalhadores*/
void adicionaTrabalhador(std::string trabalhador, Dia d, std::vector<std::string> config);
std::string getId();
std::string getT(int x) const;
int getNT();
bool verificaID(std::string id);
Trabalhadores* removeTrabalhador(std::string id);
void moveTrabalhador(Trabalhadores *trab);
int precoTrab(std::string id);
void despedimento(int d);
void adicionaNDT();
void setCansar(float x, std::string id);
int getDiasTrabalhar(int x);
void removeT(std::string id);
bool procuraT(std::string trab, int *q);

/*Verificações*/
bool verificaTrabalhadorNaZona(std::string tipoProducao);
bool verificaEspaco(float *x);
bool procuraTrabalhador(std::string trab, float *q);
int quantidadeRecursosZona();
bool quantidadeDisponivel();
};

```

Função que verifica se existem trabalhadores na zona.

Função procura trabalhadores, neste caso, lenhadores.

Função que verifica se existe espaço no armazém de um edifício.

Função que retorna a quantidade de recursos numa determinada zona.

Função que retorna se existe quantidade de recursos numa determinada zona.

**Nota:** Todas as funções não identificadas nesta classe dependem de funções já explicadas nas classes anteriores.

## 2.6. Dia

Esta classe representa o dia. É uma classe simples onde a sua única função é de informar o Dia ao resto do programa.

```
class Dia {  
    int dia;  
public:  
    Dia(int dia=1) : dia(dia) {}  
    int getDia(){return dia;}  
    Dia operator++();  
    ~Dia() = default;  
};
```

Função que retorna que incrementa o dia.

Função que retorna o dia atual.

Figura 29 - Classe Dia

## 2.7 Infolha

Esta classe representa as dimensões da ilha e a sua única função é de informar as dimensões da ilha ao resto do programa.

```
class InfoIlha{  
    int linha;  
    int coluna;  
public:  
    InfoIlha(int l, int c);  
    int getLinha() const;  
    int getColuna() const;  
    ~InfoIlha() = default;  
};
```

Funções que retornam as dimensões da ilha.

Figura 30 - Classe Infolha



### 3. Jogo

No início do jogo o jogador para jogar pode usar o comando 1 ou “Jogar” e assim inicia o jogo. Caso queira ver algumas informações sobre o jogo pode usar o comando 2 ou “Informações” e para sair pode usar o comando 3 ou “Sair”.

```

=====Simulador de construção e desenvolvimento=====
Indique o que pretende fazer:
  1. Jogar.
  2. Informação.
  3. Sair.
  Opção: |

```

Figura 31 - Menu Inicial do Jogo

#### 3.1 Aspeto do Jogo

Depois de iniciado o jogo é pedido para indicar o número de linhas e de colunas que deseja que a ilha apresente e de seguida é apresentado o Dia e os recursos disponíveis, começando o jogo com 2000€, 2 barras de aço e 20 vigas de madeira.

```

Introduza as dimensões para a ilha:
Linhas: 4
Colunas: 5
Dia: 1
dinheiro = 2000 ferro = 0 carvão = 0 barras de aço = 2 vigas de madeira = 20 madeira = 0 eletricidade = 0
*****
*znz *mnt *pas *znz *pnt *
* * * * *
* * * * *
* 0 * 0 * 0 * 0 * 0 *
*****
*znz *flr *pas *mnt *pnt *
* * * * *
* * * * *
* 0 * 0 * 0 * 0 * 0 *
*****
*dsr *flr *dsr *pnt *mnt *
* * * * *
* * * * *
* 0 * 0 * 0 * 0 * 0 *
*****
*mnt *dsr *znz *pas *flr *
* * * * *
* * * * *
* 0 * 0 * 0 * 0 * 0 *
*****

Introduza um comando: |

```

Figura 32 - Início do Jogo

Caso o jogador decida construir um edifício, por exemplo uma mina de ferro terá que usar o comando para construir um edifício e será decrementado caso possível aos recursos.

```

Introduza um comando: cons mnF 1 1

Pretende pagar o edifício em dinheiro ou em vigas de madeira?
R: dinheiro

Aviso: O edifício mnF criado com sucesso!

Introduza um comando: next
Dia: 2
dinheiro = 1900 ferro = 0 carvão = 0 barras de aço = 2 vigas de madeira = 20 madeira = 0 eletricidade = 0
*****
*znz *mnt *pas *znz *pnt *
*mnF * * * * *
* * * * *
* 0 * 0 * 0 * 0 * 0 *
*****
*znz *flr *pas *mnt *pnt *
* * * * *
* * * * *
* 0 * 0 * 0 * 0 * 0 *
*****
*dsr *flr *dsr *pnt *mnt *
* * * * *
* * * * *
* 0 * 0 * 0 * 0 * 0 *
*****
*mnt *dsr *znz *pas *flr *
* * * * *
* * * * *
* 0 * 0 * 0 * 0 * 0 *
*****

```

Figura 33 - Construção de um edifício

Quando o jogador contrata um trabalhador, este vai para a zona Pastagem e de seguida pode ser movido para qualquer zona do mapa. O jogador pode ainda ligar ou desligar o edifício caso queira que este produza ou não. (Todos os edifícios quando são construídos estão desligados).

```

Introduza um comando: cont miner
Trabalhador miner foi contratado com sucesso e o seu id é 2.1!

Introduza um comando: liga 1 1

Aviso: O edifício ligado com sucesso!

Introduza um comando: move 2.1 1 1
Trabalhador movido com sucesso!

```

Figura 34 - Outros Comandos

Caso o jogador pretenda ver as informações de uma determinada zona pode usar o comando list. Ao executar este comando é apresentado um quadro com o nome da zona, o nome do edifício e a que nível está e o número de trabalhadores de cada tipo.

```

Introduza um comando: list 1 1

Informação da zona:
*****
*Zona -> Zona-X                                     *
*Edifício -> Mina de ferro (ligado)                   *
*  Nível -> 1                                         *
*Trabalhadores:                                       *
*  Mineiros -> 1                                     *
*  Operarios -> 0                                    *
*  Lenhadores -> 0                                    *
*****

```

Figura 35 - Comando List

Como podemos ver na zona selecionada é apresentado o edifício (Mina de Ferro – mnF) e o trabalhador (Mineiro – M) e a quantidade de trabalhadores na zona (1).

Com o edificio ligado podemos verificar que a quantidade de ferro na ilha aumentou.

```

Dia: 3
dinheiro = 1885  ferro = 2  carvao = 0  barras de aço = 2  vigas de madeira = 20.2  madeira = 0  eletricidade = 0
*****
*znz *mnt *pas *znz *pnt *
*mnF *  *  *  *  *
*M  *  *  *  *  *
* 1 * 0 * 0 * 0 * 0 *
*****
*znz *flr *pas *mnt *pnt *
*  *  *  *  *  *
*  *  *  *  *  *
* 0 * 0 * 0 * 0 * 0 *
*****
*dsr *flr *dsr *pnt *mnt *
*  *  *  *  *  *
*  *  *  *  *  *
* 0 * 0 * 0 * 0 * 0 *
*****
*mnt *dsr *znz *pas *flr *
*  *  *  *  *  *
*  *  *  *  *  *
* 0 * 0 * 0 * 0 * 0 *
*****

```

Figura 36 - Exemplo de como obter recursos

## 4. Conclusões Finais

Em suma, este trabalho permitiu aprender e aplicar os conhecimentos da linguagem C++ dados em contexto de sala de aula.

Assim, podemos dizer que este trabalho nos permitiu explorar a programação orientada a objetos e as suas vantagens em relação a outras linguagens e a outros estilos de programação.