

# INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

INSTITUTO POLITÉCNICO DE COIMBRA

Licenciatura em Engenharia Informática 2º Ano – 1º  
Semestre 2021/2022

## Natural Reserve Simulator

Rafael Couto      Nº 2019142454

Rafaela Carvalho Nº 2019127935

COIMBRA

22 de novembro de 2021

## Índice

Introdução .....	3
Critérios de funcionamento.....	4
Classes.....	4
Interface .....	7
Decisões tomadas.....	8
Anexos.....	9

## Introdução

A elaboração deste trabalho prático visa consolidar conhecimentos em linguagem C++, explorados nas aulas teóricas e práticas, criando capacidade de desenvolvimento de aplicações nesta linguagem de programação.

Pretende-se criar um simulador, denominado *Natural Reserve Simulator*, destinado a gestão e desenvolvimento de uma reserva natural. Ao jogador será atribuída uma reserva que o mesmo deve gerir, provando-a e alimentando todos os seres vivos.

Deste modo, será implementada uma classe geral *Reserva* responsável por suportar todo o tipo de dados relativos a cada *célula* e à sua pormenorização.

Ora, cada *Reserva* tem *nLinhas* por *nColunas* e cada unidade será uma *cell* que é descrita por *animais* e *alimento*.

A interação com o jogo processa-se através de comando e ação por parte do jogador.

## Critérios de funcionamento

O jogo *Natural Reserve Simulator* foi inicialmente desenvolvido na perspetiva de poupança de recursos e simplicidade de código, para tal, foi reduzido aos ficheiros e classes estritamente necessárias.

## Classes

Foi criada uma classe *Cell* que representa cada unidade da *Reserva*. É uma classe descritora dos pormenores existentes em cada célula da reserva.

```
#ifndef TPP002223_CELL_H
#define TPP002223_CELL_H

#include "Utils.h"
#include "Animais.h"
#include "Alimentos.h"

class Cell {
private:
    int Lin, Col;
    vector<Animais> animais;
    Alimentos* food;
public:
    Cell(int Lin, int Col);
    static string getInfo(Cell *c);
    void getAnimalSpecie();
    void setAnimal(string specie);
    string getFood(Cell *c);
    void setFood(Alimentos);
    int test = 3;
    ~Cell();
    string getAnimais(Cell *c);
};

#endif //TPP002223_CELL_H
```

Foi criada uma classe *Alimento* que representa cada tipo de alimento que se encontra na *Reserva*. É uma classe que contém as características dos alimentos.

```
#ifndef TPP002223_ALIMENTOS_H
#define TPP002223_ALIMENTOS_H

#include "Utils.h"

class Alimentos {
private:
    string const nome;
    int duracao;
    int valorNutritivo;
    int toxicidade;
    string const cheiro;
    vector<string> AnimaisMortes;
public:
    Alimentos(string n, int d, int vn, int t, string c): nome(n), duracao(d), valorNutritivo(vn), toxicidade(t), cheiro(c){}
    string getNome() const{return nome;}
    string getCheiro() const{return cheiro;}
    int getDuracao() const{return duracao;}
    int getValorNutritivo() const{return valorNutritivo;}
    int getToxicidade() const{return toxicidade;}
    void aumentaToxicidade();
    void diminuiValorNutritivo();
};
```

Foi criada uma classe *Animal* que representa cada tipo de animal que se encontra na *Reserva*. É uma classe que contém as características dos animais.

```
class Animais {
private:
    string const nome;
    string const especie;
    int vida;
    int saude;
    int fome;
    float peso;
public:
    Animais(string e,int v,float p,int s,int f=1):especie(e),vida(v),saude(s),fome(f),peso(p){}
    int getVida() const {return vida;}
    int getSaude() const {return saude;}
    int getFome() const {return fome;}
    float getPeso() const {return peso;}
    string getNome() const{return nome;}
    string getEspecie() const{return especie;}
    void setVida(int vida);
    void setSaude(int saude);
    void setFome(int fome);
    void setPeso(float peso);
    bool vivo();
    void aumentaFome();
    void diminuiFome();
    void aumentaSaude();
    void diminuiSaude();
};
```

Foi criada uma classe *Reserva* que é onde serão guardados os dados do jogo e é usado como intermediário no acesso à informação das *Cell* por parte da *Interface*.

```
class Reserva {
private:
    int NL,NC;
    int viewWindow = 5;
    int topLeftCornerX = 0, topLeftCornerY = 0;
    int id = 1;
    int simulatedTime();
    vector<Reserva*> reservas; // Vector to store game states
    vector< vector<Cell*>> area; // Vector to store the cells
public:
    Reserva()=default; // Default constructor, applies predefined size
    Reserva(int Id=0):id(Id){}
    Reserva(Reserva *Reserva); // Copy constructor
    Reserva(int NL, int NC, string filename = ""); // Constructor with size and filename
    int getNL() const;
    int getNC() const;
    int getViewWindow() const;
    int getTopLeftCornerX() const;
    void setTopLeftCornerX(int topLeftCornerX);
    int getTopLeftCornerY() const;
    void setTopLeftCornerY(int topLeftCornerY);
    int getId() const {return id;};
    int getSimulatedTime() const;
    string getCellInfo(int lin, int col);
    const vector<vector<Cell*>> &getArea() const;
    ~Reserva(); // Destructor
};
```

Foi criada um ficheiro header *utils* que contém todos os *includes* necessários para a execução do simulador.

```
#include <string>
#include <fstream>
#include <random>
#include <iostream>
#include <vector>
#include <sstream>
#include <ctime>
#include <cstdlib>
#include <cstdio>
#include <iomanip>
#include <unistd.h>

using namespace std;
```

## Interface

Criou-se uma classe *Interface*, onde é criado e tratado o ambiente do jogo e onde é tratado também da validação dos comandos que vão ser utilizados ao longo do simulador.

[illegible]

## Decisões tomadas

Em relação à parte que visível no ecrã tomamos a decisão de mostrar um tamanho fixo, definido na classe *Reserva* com o nome *viewWindow*.

De modo que esta *viewWindow* seja independente de toda a visualização do tabuleiro de jogo foi criado uma variável *topLeftCorner* que permitirá fazer o cálculo de apresentação das células no intervalo desejado.

Tal como descrito no enunciado o número de linhas e colunas do tabuleiro de jogo é pedido ao utilizador em *runtime* no início da execução do programa.

Foi decidido que o jogo iniciaria sempre a sua área visual no canto superior esquerdo (0,0) e daí em diante é controlado através do utilizador.

Tomamos por iniciativa mais uma vez criar um comando clear, comando que irá ser destinado para limpar tudo o que já foi escrito no terminal para que se possa continuar a usar o simulador sem falta de informação por falta de espaço no terminal.



## Anexos

- Reserva.h
- Reserva.cpp
- Cell.h
- Cell.cpp
- Animais.h
- Animais.cpp
- Alimentos.h
- Alimentos.cpp
- Interface.h
- Interface.cpp
- Utils.h
- main.cpp
- constantes.txt
- commands.txt
- POO – 2223 - Relatório.pdf