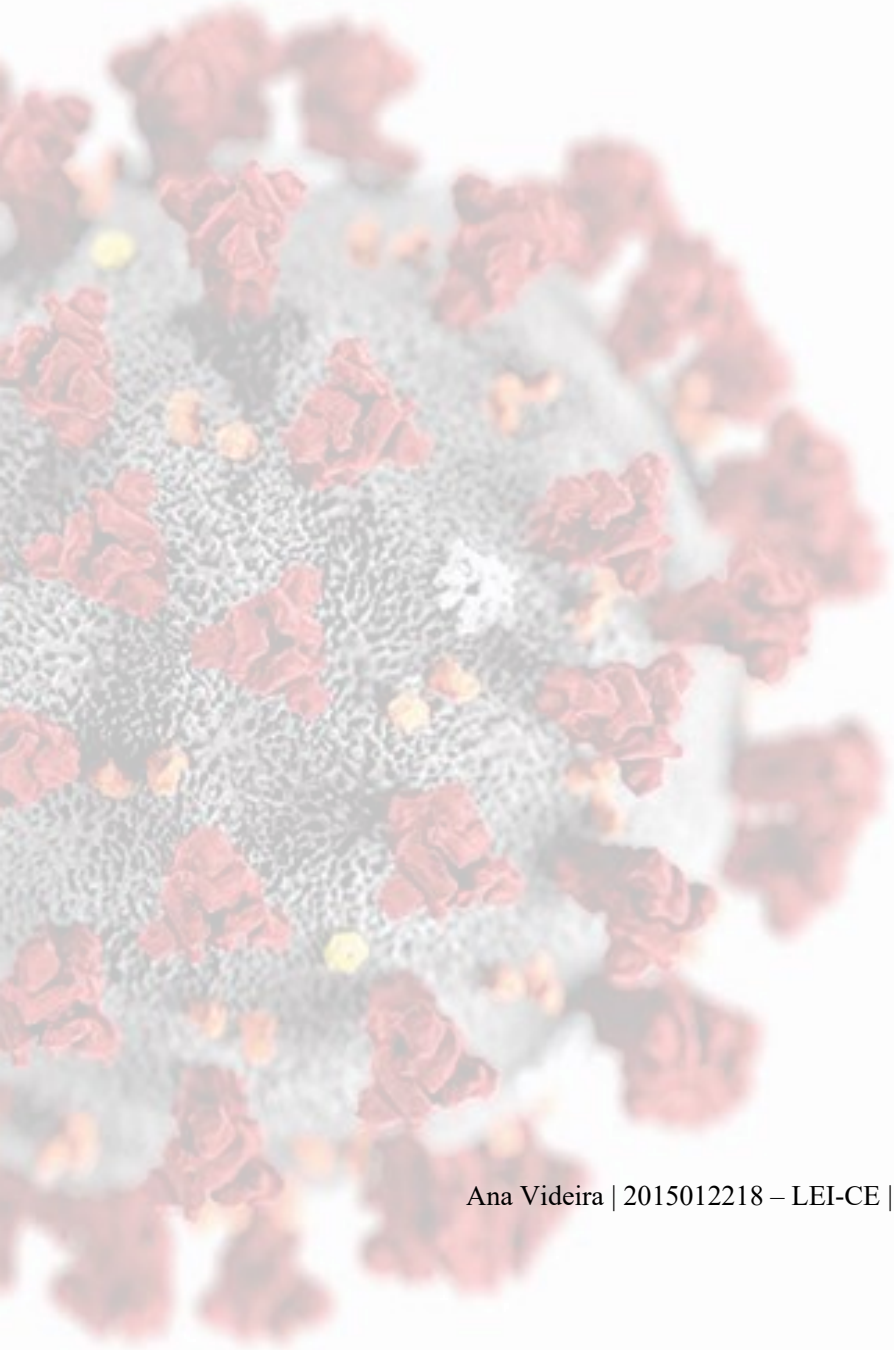
Simulação de Propagação de Vírus

[Programação 2019-2020]



Ana Videira | 2015012218 – LEI-CE | a21250074@isec.pt

Índice

Introdução

A implementação deste programa provém de uma proposta de trabalho prático da unidade curricular de Programação. Desenvolvido com recurso ao Netbeans e em ambiente Windows, este é um projeto que tem como objetivo desenvolver competências em linguagem de programação C.

Constituído por diversos tipo de ficheiros, o programa permite a recolha de informação inscrita nos ficheiros binário e texto fornecidos para sua memoria dinâmica. Assim, através de vários dos vários mecanismos da linguagem C é possíveis manipular e apresentar essa informação de acordo com o que requerido.

Estruturas de dados

```
struct sala{
    int id;
    int capaciadade;
    int liga[3];
};
```

Sala – Estrutura incluída no enunciado que contem as informações sobre uma sala

```
struct Pessoa{
    char identificador[MAX_NAME];
    int idade;
    char estado;
    int dias;
    int local;
    ppessoa prox;
};
```

Pessoa – Estrutura construída par armazenar as informações de cada pessoa

```
struct Simulacao{
    local l;
    psimulacao prox;
    ppessoa listaP;
    int nPessoas;
};
```

Simulacao – Estrutura criada para armazenar as informações das salas e pessoas durante o tempo de simulação

```
typedef struct sala local, *plocal;
typedef struct Pessoa pessoa, *ppessoa;
typedef struct Simulacao simulacao, *psimulacao;
```

Estruturas Dinâmicas

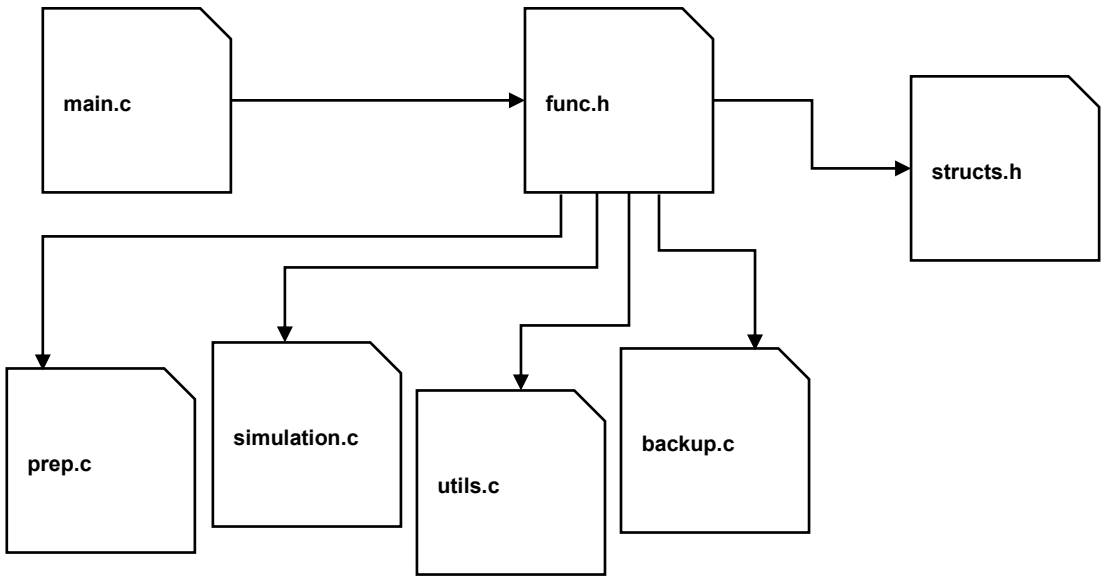
locais – Vetor dinâmico de estruturas locais; As informações de cada sala presentes no ficheiro binário são carregadas para este vetor dinâmico.

pessoas – Lista ligada de estruturas Pessoa. Lista carregada pelas informações do ficheiro texto fornecido.

s – Lista ligada de estruturas Simulacao . É carregada com informação pela função **loadList(pessoas);**

back – Lista ligada de estruturas Simulacao. Esta é uma cópia da lista ligada s para ser possível retroceder 1 a 3 iterações durante a simulação.

Implementação



ESTRUTURA GERAL DA IMPLEMENTAÇÃO DO PROGRAMA

No ficheiro **main.c** encontram-se as declarações das várias variáveis e ponteiros necessários ao funcionamento do programa e também o menu e submenu que possibilita a escolha das ações a executar.

Em **utils.c / utils.h** foram implementadas apenas funções de suporte a outras, como, por exemplo: a função de apresentação no ecrã das opções dos menus presentes no **main.c** , as funções fornecidas para calculo de probabilidade, entre outras .

backup.c contém as funções responsáveis por retroceder até 3 iterações e desfazer a alterações efetuadas.

Espaços e Pessoas

Inicialmente no ficheiro **prep.c / prep.h** , e segundo as indicações do enunciado, as informações sobre as salas e as pessoas presentes no ficheiros fornecidos são carregadas para o vetor dinâmico locais e lista ligada pessoas, respetivamente. Também aqui é configurada / carregada a lista ligada para a simulação.

local * loadVetor(int *n); -> carrega o vetor dinâmico local

ppessoa loadList(ppessoa p); -> carrega a lista ligada pessoas

psimulacao simulatioSettings(psimulacao s,ppessoa p, plocal l ,int *tam); -> carrega a lista ligada s

Está aqui também implementada a função **addPessoa** que sendo que inicialmente foi feita como suporte á função **loadList**, também é reutilizada ao longo da aplicação por outras funções .

Configuração e Simulação do Modelo de Propagação do vírus

No ficheiro **simulation.c / simulation.h** encontram-se as funções responsáveis por toda a simulação , ou seja, pela manipulação da lista ligada **s** e **back** . A função seguinte é a principal responsável por todo o processo: **psimulacao simulationGO(psimulacao s, psimulacao back);**

A função "avança " uma iteração / um dia de cada vez que é executada. A cada iteração atualiza a lista ligada **s** , responsável por armazenar dinamicamente as informações sobre as salas e pessoas durante a simulação. A cada iteração são também guardadas as informações da lista **s** em **back** de modo a futuramente existir a hipótese de ter acesso a essas informações e reverter algumas alterações.

Esta começa por incrementar a variável global Iteração e segue passando os "nós" da lista ligada de sala em sala, pessoa a pessoa, aplicando as taxas configuradas. Faz isto selecionando sala e dividindo as pessoas seguindo o seu estado:

- Se a pessoa em questão é doente, aplica a função da taxa de disseminação, o tempo de infeção e de recuperação;
- De acordo com o tempo de infeção a função **infectionTime** decide de a pessoa muda o seu estado para Morto, Recuperado ou apenas adiciona mais um dia aos dias de infeção.
- A recuperação é dada pela função **recovery** que aplica a probabilidade de 20%, como descrito no enunciado de uma pessoa recuperar totalmente e mudar o seu estado de doente para recuperado.

Quando conclui a passagem por todas as pessoas da sala, aplica-lhe a função:

ppessoa infectPeople(ppessoa p, int n);

Esta passa novamente por todas as pessoas da sala em questão e infecta as pessoas saudáveis ou recuperadas, de acordo com o número de pessoas a infetar devolvido pela taxa de disseminação encontrado anteriormente. Repete o processo por cada uma das salas presentes na lista ligada s e retoma ao menu da simulação.

Funcionalidades

Após a configuração da simulação através do submenu estão disponíveis as seguintes opções:

Avançar 1 iteração na simulação

Pela função **psimulacao simulationGO(psimulacao s, psimulacao back);** como falado anteriormente.

Voltar atrás 1 a 3 iterações

psimulacao goBack(psimulacao s, psimulacao b); - Não funcional – Esta permitiria o retroceder , até 3 iterações nas alterações efetuadas á lista ligada da simulação onde se encontram todas as informações sobre locais/salas e pessoas.

Apresentar estatística

void showStatistics(psimulacao s); - Permite calcular e mostrar várias estatísticas: taxa de mortalidade, de recuperação de infetado e tempo medio de dias doente. Foi necessário utilizar o método de casting de forma a fazer a divisão de dois inteiros com resultado float.

Adicionar doente

void addInfected(psimulacao s); - Permite inserir uma nova pessoa doente numa das salas disponíveis. Utiliza a função **addPessoa** como recurso.

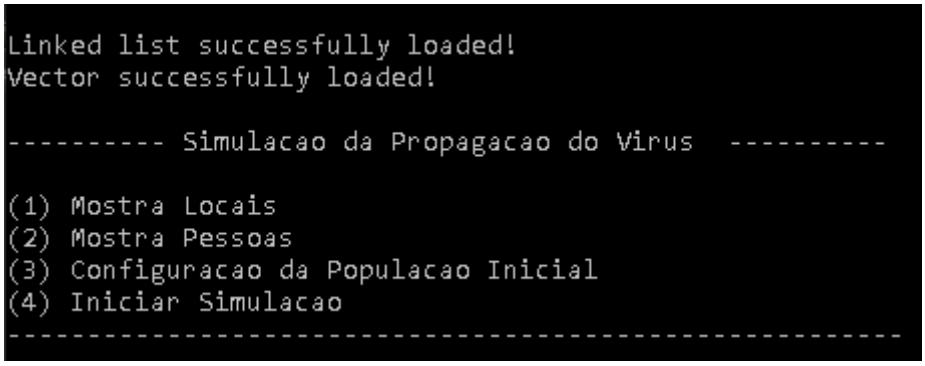
Transferir pessoas

pela função **void transferPerson(psimulacao s);** - Permite selecionar uma pessoa especifica e movê-la de sala. Isso acontece adicionando os dados de uma pessoa na nova sala pretendida e eliminando-os na sala onde se encontrava antes.

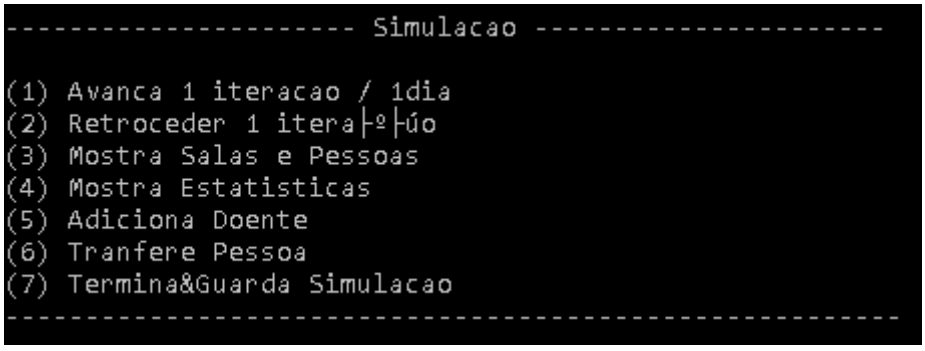
Terminar simulação

void simulationEXIT(psimulacao i); - Função que mostra todas as informações presentes na simulação e grava -as num ficheiro texto , com recurso á função **saveData(psimulacao s);** Por fim liberta a memória utilizada pela lista ligada , com recurso à função **freeList(psimulacao s);** .

Manual de Utilização



MENU PRINCIPAL



MENU SECUNDÁRIO DE SIMULAÇÃO

Apenas disponível depois da seleção da opção 3 no menu principal

Conclusão

Virus , programa implementado em linguagem C, permite a gestão de informações relativas a simulações da propagação de qualquer vírus. De interface e funcionamento simples , este programa , embora dependa do carregamento de informações de ficheiros externos, utiliza uma organização de memória dinâmica que permite uma maior agilidade na manipulação de informação.

O grau de dificuldade de implementação deste pequeno projeto não se considera muito alto pois vai de encontro com os assuntos tratados na unidade curricular de programação. Apesar deste facto, podemos também considerar a implementação e manipulação de memoria dinâmica um desafio devido á necessidade e conhecer com algum grau de profundidade diversos conceitos complexos.

Assim, retira-se uma nota positiva desde desafio. Ainda que não concluído na sua totalidade, funções consideradas principais estão operacionais e em correta execução.