Licenciatura em Engenharia Informática

Relatório do Trabalho Prático

Programação

Dorin Bosîi

2019112586

Índice

1.Introdução 3
2.Estrutura de Dados 4
3.Implementação 5
4.Manual de Utilização 7
5.Conclusão 9

1. Introdução

O presente trabalho foi realizado no âmbito da disciplina de Programação, com o objetivo de implementar um programa capaz de simular a propagação de um vírus entre pessoas em locais interligados.

Este relatório tem como principal objetivo descrever as várias partes e aspetos mais importantes do trabalho, como por exemplo a estrutura de dados e o funcionamento da mesma.

2. Estrutura de Dados

De forma a deixar o código mais claro este mesmo foi dividido por 3 ficheiros principais que contêm todas as funções necessárias para o bom funcionamento do programa.

Como o enunciado propõem foi necessário implementar as duas principais estruturas que armazenassem os dados a serem tratados e estes são: Local e Pessoa.

- Local: Esta struct tem como objetivo armazenar os dados lidos a partir de um ficheiro binário para uma lista de locais apresentada na última linha de código a seguir.

```
typedef struct local Local;
struct local{
    int id;
    int capacidade;
    int conections[MAX_CONECTIONS];
};
```

- Pessoa: Esta struct contem a informação de cada pessoa lida de um ficheiro texto e é utilizada para construir uma lista ligada. Esta struct contem um elemento a mais do que os que são descritos no enunciado do trabalho que é um ponteiro para um Local.

```
typedef struct pessoa  Pessoa;
struct pessoa {
    char nome[NAME_LENGHT];
    int idade;
    char estado;
    int dias;
    Local *local;
};
```

- PessoaList: Struct que serve para armazenar todas as pessoas lidas do ficheiro.

```
typedef struct pessoaList PessoaList;
struct pessoaList{
    Pessoa pessoa;
    struct pessoaList *next;
};
```

- SimaltionIter: Struct que tem como objetivo armazenar uma copia da lista de pessoas e locais de cada iteração realizada na simulação.

```
typedef struct simulationIter{
    PessoaList * pessoas;
    Local * locais;
    int iter;
}SimulationIter;
```

- BackUp: É uma struct que representa uma lista ligada de SimulationIter e que vai conter no máximo 3 nós na lista.

```
typedef struct backUp{
    //SimulationIter * backupIter[MAX_BACKUP];
    SimulationIter * iter;
    struct backUp * next;
}BackUp;
```

3. Implementação

No ficheiro Local.c está implementado uma função que lê os locais a partir de um ficheiro binário e faz automaticamente as verificações necessárias. Os locais são preenchidos para um array de Local que inicialmente esta a NULL.

No ficheiro Pessoa.c estão implantadas algumas das funções necessárias para o funcionamento necessário do programa como a transferência de pessoas aleatórias e adicionar um doente novo à simulação. Para a inicialização da lista ligada com as pessoas lidas do ficheiro txt foi criada uma lista ligada temporária em que lhe é adicionado as pessoas que tenham passado pelas verificações feitas, cada pessoa é adicionada ao ultimo nó da lista e no fim caso não exista nenhum erro é retornado o ponteiro do inicio da lista que vai ser utilizado no resto do programa.

No meu programa preferi que cada pessoa fosse inserida numa sala invés das salas conterem várias pessoas. Na transferência de pessoas elas apenas são transferidas de forma random caso o utilizador indique que o número de pessoas a ser transferida é menor que as pessoas existentes na sala.

Nos ficheiros main e simulação é onde se encontra a grande parte da funcionalidade do programa. Cada simulação representa um dia e no programa so é permitido avançar um dia de cada vez e não é considerado se o estado do doente tenha mudado na iteração e desta forma um doente pode ser infetado e curado na mesma interação.

Antes de se avançar 1 dia na simulação os dados das pessoas e dos locais são duplicados para um SimulationIter e posteriormente adicionados a uma lista ligada de buckups. A SimulationIter é adicionada ao primeiro nó da lista de buckups e sempre que a lista ultrapassar o máximo de iterações a serem guardadas (neste caso 3) o último nó da lista é removido da memoria. Estas operações são feitas em primeiro lugar quando o utilizador escolhe avançar iteração e de seguida é feita a nova iteração.

Quando o utilizador escolhe avançar iteração, depois das iterações anteriores serem guardadas, é executada a função de taxa de disseminação que tem como objetivo percorrer as pessoas todas num local e caso exista uma pessoa infetada nesse local, dependo do numero de pessoas na sala e do valor da taxa definida no código, tenta infetar um certo numero de pessoas aleatórias na sala.

No fim da execução do programa é guardado um relatório igual ao obtido nas estatísticas da simulação sendo que no final de o relatório ter sido criado, toda a memoria utilizada pelo programa é libertada.

4. Manual de Utilização

Quando o programa é executado é necessário escrever o nome dos ficheiros para a leitura dos dados.

```
Nome do ficheiro para os Locais:
- E1.bin
Nome do ficheiro para os Pessoas:
- pessoasA.txt
```

De seguida aparece um menu com várias opções a realizar.

```
------ MENU PRINCIPAL ------
1 -Avancar 1 iteracao
2 -Apresentar Estatistica
3 -Adicionar Doente
4 -Transferir Pessoas
5 -Retornar Iteracoes
6 -Terminar Simulacao
- Opcao:
```

Opção 1: a iteração avança de forma automática e o utilizador só precisa de escolher a opção.

```
------ MENU PRINCIPAL ------
1 -Avancar 1 iteracao
2 -Apresentar Estatistica
3 -Adicionar Doente
4 -Transferir Pessoas
5 -Retornar Iteracoes
6 -Terminar Simulacao
- Opcao: 1
Iteracao avancada
```

Opção 2: quando é escolhida é apresentado um relatório de fácil compreensão com os dados processados pelo programa.

```
- Opcao: 2
********** Distribuicao **********
Sala 1 (capacidade -> 48):
Nome: Tomas111 || Idade: 12 || Estado: S;
Nome: Zulmira2A || Idade: 17 || Estado: S;

Sala 2 (capacidade -> 47):
Nome: AnaLebre34A || Idade: 55 || Estado: I;
Nome: PauloPires2 || Idade: 67 || Estado: D || Dias infetado: 11;
Nome: LuisaSantos || Idade: 40 || Estado: D || Dias infetado: 4;

Sala 3 (capacidade -> 49):
Nome: auloPires1 || Idade: 23 || Estado: S;

Sala 4 (capacidade -> 50):


Numero de Infetados: 2 (33.333333 %)
Numero de Saudaveis: 3 (50.000000 %)
Numero de Imunes: 1 (16.666667 %)
Numero de Itereacoes: 1
```

Opção 3: Para transferir pessoas são pedidos vários dados para o utilizador introduzir sendo apresentadas as escolhas que este pode tomar.

```
- Opcao: 3
Local 1 (capacidade -> 50)
Local 2 (capacidade -> 46)
Local 3 (capacidade -> 49)
Local 4 (capacidade -> 49)
Intruduza o id da sala em que pretende adicionar o doente:
- 1
Intruduza o nome do doente:
- dorin
Intruduza a idade do doente:
- 20
Intruduza o numero de dias que o doente esta infetado:
- 4
Doente adicionado com sucesso
```

Opção 4: Para adicionar um novo doente é necessário introduzir os dados como são pedidos e entre cada dado a introduzir é mostrada uma ajuda para facilitar a escolha do utilizador.

```
- Opcao: 4
Local 1 (capacidade -> 50)
Local 2 (capacidade -> 46)
Local 3 (capacidade -> 49)
Local 4 (capacidade -> 49)
ID da sala que pretende transferir:
- 1
Conecoes da sala:3 ;
ID da sala que pretende para qual deseja transferir:
- 3
Capacidade do destino 3: 49
Pessoas no local 1: 1
Quantas pessoas deseja transferir de sala?
- 1
Transferencia efetuada com sucesso
```

Opção 5: Para retornar iterações o utilizador pode escolher no máximo 3 caso exista esse número de iterações a retornar, mas é sempre apresentado o número possível a retornar.

```
- Opcao: 5
Tem 3 iteracoes guardadas
Quantas iteracoes deseja voltar a traz? (MAX: 3)
- 3
Voltou para a iteracao : 2
```

Opção 6: Para terminar a Simulação é necessário indicar o nome do ficheiro para qual os dados vão ser exportados, por exemplo “Relatorio.txt”.

```
- Opcao: 6
Nome do ficheiro para guardar o Relatorio final da simulacao:
-Relatorio.txt
memoria libertada
Ficherio completo
Press [Enter] to close the terminal ...
```

5. Conclusão

Com a realização deste trabalhei fiquei a perceber melhor como armazenar e manipular dados de forma dinâmica de forma a partir cheiros com vários formatos. Este trabalho também foi essencial para perceber melhor o funcionamento de ponteiros na linguagem C.