INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA
DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
E DE SISTEMAS ENGENHARIA INFORMÁTICA
PROGRAMAÇÃO 2018/2019

Programação

Simulação da Propagação de Virus

Francisco Amaral
2018019123

Índice

Estruturas de dados

- Locais - Os locais estão guardados na estrutura dada no enunciado.A estrutura foi definida como struct sala e no resto do trabalho referenciada por local (através da diretiva typedef). O ponteiro para esta está nomeado como plocal no resto do trabalho.

```
typedef struct sala local, *plocal;
struct sala{
        int id; // id numérico do local
        int capacidade; // capacidade máxima
        int liga[3]; // id das ligações (-1 nos casos não usados)
};
```

- Pessoas - Esta estrutura guarda a informação necessária para a manutenção das pessoas em lista ligada, conforme foi pedido. Definida struct persons e posteriormente utilizada sob o nome pessoa e pontpessoa.

```
struct persons{
   char id[200];
   int idade;
   char estado;
   int dias;
   pontpessoa prox;
   int localizacao;
   };
```

O campo **id** guarda o nome da pessoa, a **idade** a idade da pessoa, o **estado** identifica a pessoa como S(saudável), D(doente) ou I(imune). O campo **dias** diz-nos quanto tempo a pessoa está infectada no caso desta ser Doente, pois se for imune ou saudável este campo está a 0 e o ponteiro **pontpessoa prox** aponta para uma estrutura do mesmo tipo constituindo as pessoas em forma de lista ligada. O campo **localização** guarda a posição no array dos locais em que a pessoa vai ficar.

Estruturas dinâmicas

Na minha resolução para este problema, somente usei 2 estruturas dinâmicas que são essenciais para o seu desempenho. Seguindo as regras estabelecidas no enunciado, criei um vetor dinâmico de estruturas para os locais e também um vetor dinâmico de listas ligadas para tratar da manipulação das pessoas.

- ## Vetor dinâmico de locais

  Este vetor dinâmico adquire o seu tamanho na leitura do ficheiro binário de estruturas de locais,que seguindo a formatação deste, aloca e realoca memória enquanto for necessário até que todos os locais e as suas informações estejam guardadas. Com ele consigo carregar os locais para colocar as pessoas, verificar se os locais estão cheios e ligados entre si. Normalmente ao longo do código chamo este vetor por **loctab** ou simplesmente por **l,** do tipo **plocal**

- ## Lista ligada para pessoas

  Esta por regra geral tem o nome de **pontpessoa p.** É uma lista ligada dinâmica simples, que guarda a informação das pessoas e também guarda em si em que sala a pessoa está.Adquire o seu tamanho na leitura do ficheiro de texto fornecido, segue a formatação deste e aloca memória aquando preenche a lista ligada.

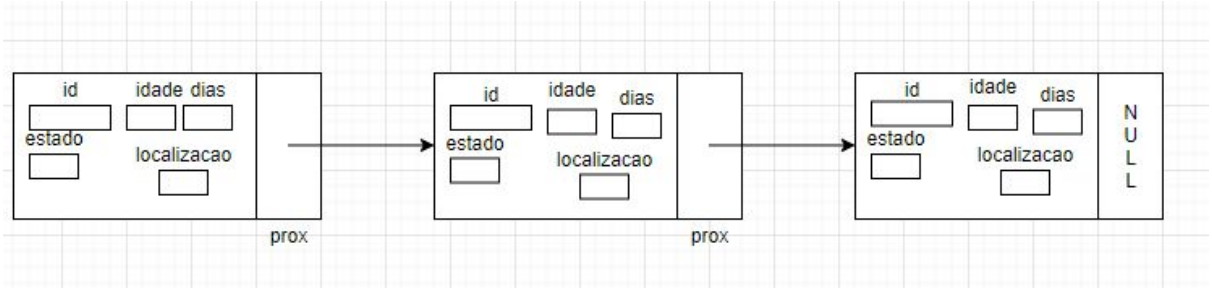


Fig 1. Representação da lista ligada implementada. Esta fica maior quanto mais pessoas houver.

Ficheiros utilizados

Neste projeto utilizei para além do main.c e dos ficheiros que foram fornecidos os ficheiros:

- **locais.c,** ficheiro com a função que faz a leitura do ficheiro binário de locais e preenche o vetor dinâmico de estruturas. Esta função devolve também, por referência, a variável **tam** que me dá o tamanho do array.
- **pessoas.c**, neste incluo todas as funções relativas às pessoas, desde iniciar as pessoas com a leitura do ficheiro de texto,criar a lista ligada e colocá-las nesta lista. Tem também as funcionalidades pedidas,desde adicionar pessoas, mover pessoas, avançar iterações, apresentar estatísticas e escrever num ficheiro de texto um report e a última iteração num ficheiro de texto com o nome escolhido pelo utilizador, exceto a funcionalidade adicional pedida.

- Cada um destes possui um **header file** destinado a si próprio, ou seja **locais.h e pessoas.h,** e as passagens de funções para o exterior do ficheiro original. Nestes também estão as estruturas declaradas.

Algoritmo de funcionamento e opções de implementação

1. O programa inicia com as fases de preparação, iniciando os locais através da leitura dos ficheiros binários e também as pessoas através da leitura dos ficheiros de texto e armazenamento de informação nas pessoas. Se houver algum erro, tal como o ficheiro não existir,e no caso das pessoas se a idade da pessoa for igual ou inferior a 0 ou os dias infectados forem negativos o programa **termina de imediato.** Isto é feito através das funções **iniciaLoc** e **iniciaPessoa**(que por sua vez vai chamar a função **criaLista** que vai criar a lista ligada e esta por sua vez vai chamar a função **preenche**, que vai preencher a lista ligada)

   a)Na função **preenche** é feita a atribuição de pessoa a local. Esta atribuição é feita de maneira bastante simples, dando uso às funções fornecidas pelos docentes, neste caso a função **intUniformRnd.** A variável que dei o nome de random fica igual a **intUniformRnd** no intervalo de 0 a tamanho do array de locais. Assim, é garantida a aleatorização da sala atribuída à pessoa. Este

número gerado fica no campo localização da estrutura **pessoas**. Tive também o cuidado de verificar se a capacidade da sala em qual a pessoa é colocada está cheia, e se estiver é feito outro random,random2, garantindo que nunca é igual ao random. Assim, é feita a atribuição aleatória de pessoas por sala.

Depois disto está concluída a fase de preparação e é apresentado um menu com as escolhas

2. Na configuração do modelo de propagação começa com a função **taxaDiss,** que com uma constante de 0,05, a qual multiplico pelo número de pessoas dá o número de pessoas que irão ficar infectados. É percorrida a lista ligada, e se a pessoa for saudável, e para garantir a aleatorização é dado uso à função **probEvento,** que com a divisão entre o número de pessoas que vão ser infetados e a população total, consigo garantir que são infetados de forma aleatória e não de forma seguida.
3. A probabilidade de recuperação é semelhante à taxa de disseminação, dou uso à função **probRecuperacao,** que com a variável **taxa** que é igualada a 1/idade da pessoa, com o apoio da função **probEvento**(se esta for == 1) que recebe esta variável como argumento, é feita a recuperação de forma aleatória. Nesta função é feita também a **imunidade**, outra vez com recurso à **probEvento**, mas neste caso com um argumento recebido de **0,2,** conforme dito no enunciado.
4. Em todas estas funções são usados ponteiros auxiliares para percorrer a lista, **nunca perdendo o ponteiro de lista.**
5. A implementação da nova funcionalidade, foi feita de forma com recurso a mais três listas ligadas do mesmo gênero, de forma a guardar até três iterações. Dadas o nome de **tab1,tab2, e tab3. Tab3** guarda a iteração mais recente,tab2 a penúltima e tab1 a mais recente. Quando se ultrapassam 3 iterações, tab1 toma o valor de tab2, tab2 o valor de tab3 e tab3 o valor atual antes de ser avançado uma iteração. Isto é feito através da função **avancalte, que permite copiar listas.**

Opções e Reflexão

Algumas das opções que tomei foram iguais às que foram lecionadas em aulas, no entanto, noutras, apesar de não ser a solução mais ótima e perfeita, funciona.

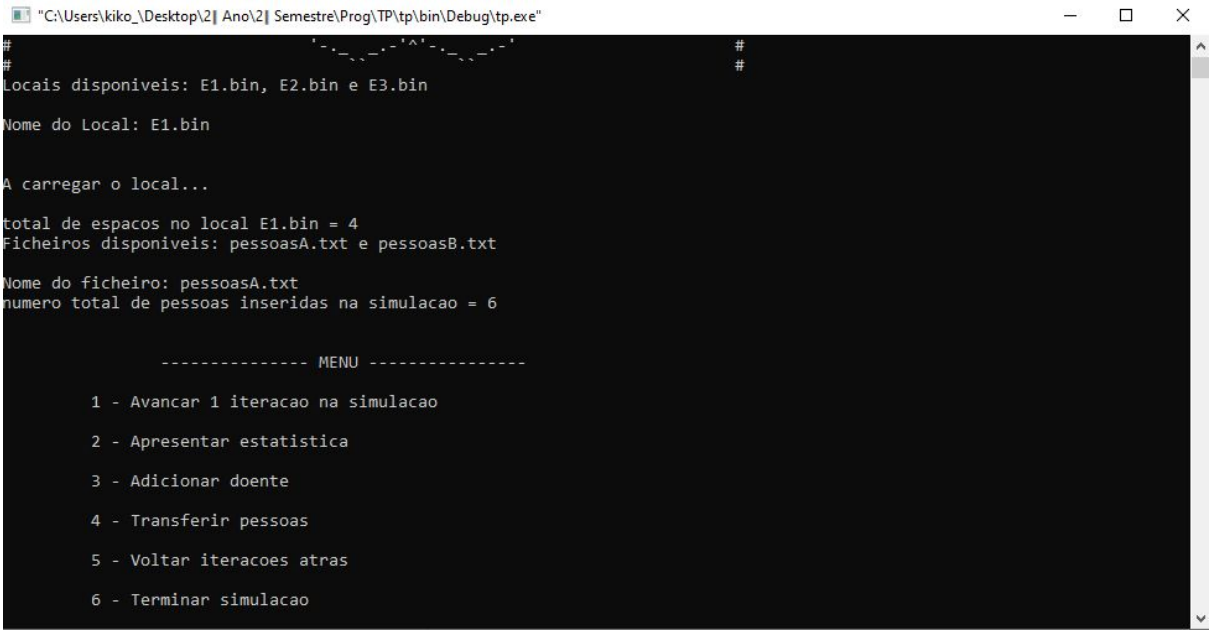
1. Para colocar as pessoas nas salas simplesmente usei um inteiro que me guarda a posição do array, ficando assim com uma sala associada. Este método de atribuição simplificou-me bastante o trabalho, pois de início tinha começado a atribuição de pessoa-> sala através de um ponteiro para os locais em cada pessoa.
2. As atribuições serem aleatórias foram todas com funções fornecidas pelos professores. Gastei bastante tempo a tentar perceber como usar a função **probEvento** no caso de mover pessoas, e a transmissão ser feita de forma aleatória.
3. A implementação da nova funcionalidade foi feita de uma forma simples e concisa, sem recurso a grandes funções. Estava a pensar implementar de forma recursiva, mas como não podia andar mais de 3 iterações para trás não o fiz

Manual de funcionamento

Quando se inicia o programa, é apresentado os locais disponíveis na simulação para o utilizador escolher. Se o utilizador escrever um local diferente dos que estão disponiveis o programa termina de imediato.

Logo após a introdução do local, é dada a informação de quantos espaços há no local escolhido, e depois são apresentados os ficheiros disponíveis para a introdução de pessoas

na simulação.Conforme nos locais, se puser um ficheiro inválido o programa termina de imediato.
Após a introdução do ficheiro das pessoas é apresentado um menu, onde tem as opções de avançar 1 iteração na simulação, apresentar estatística, adicionar doentes, transferir pessoas, voltar iteracoes atras ou terminar a simulação.



O utilizador pode andar quantas iterações quiser.

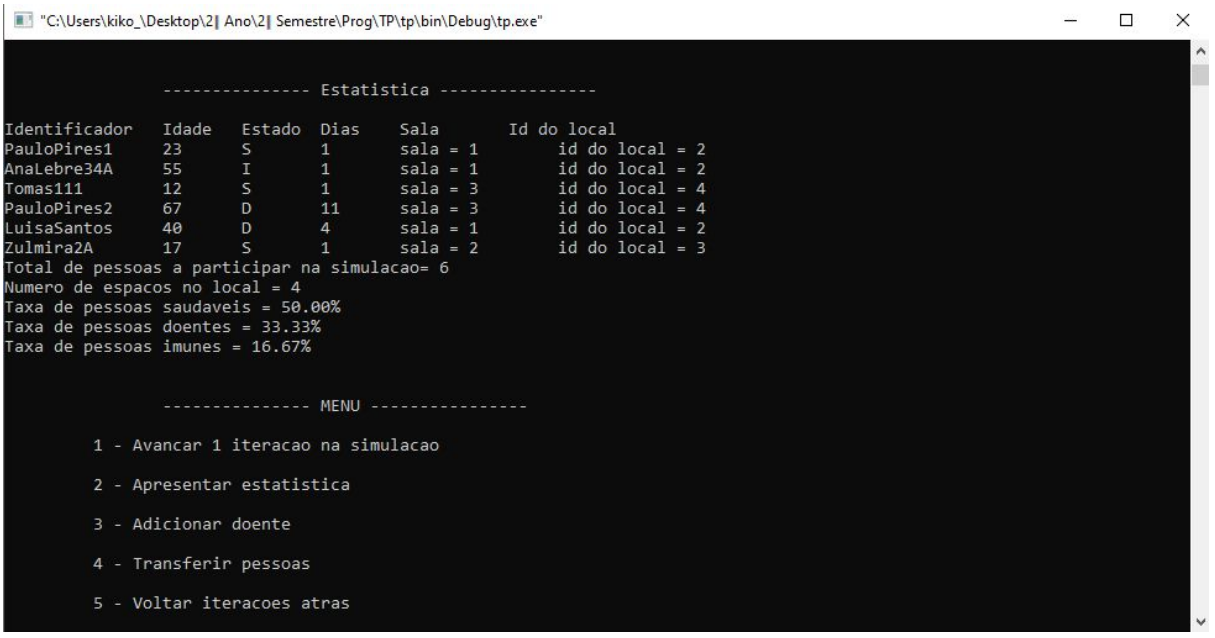


Fig.4 - Estatistica

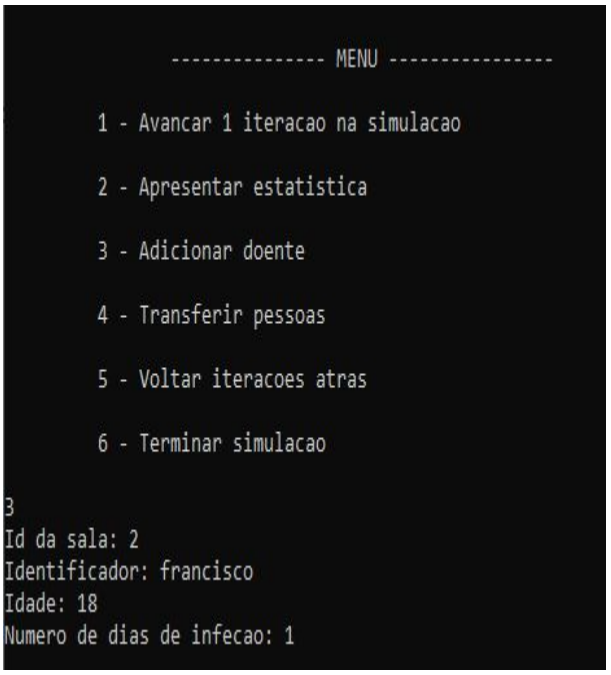


Fig.5 - Adicionar doente

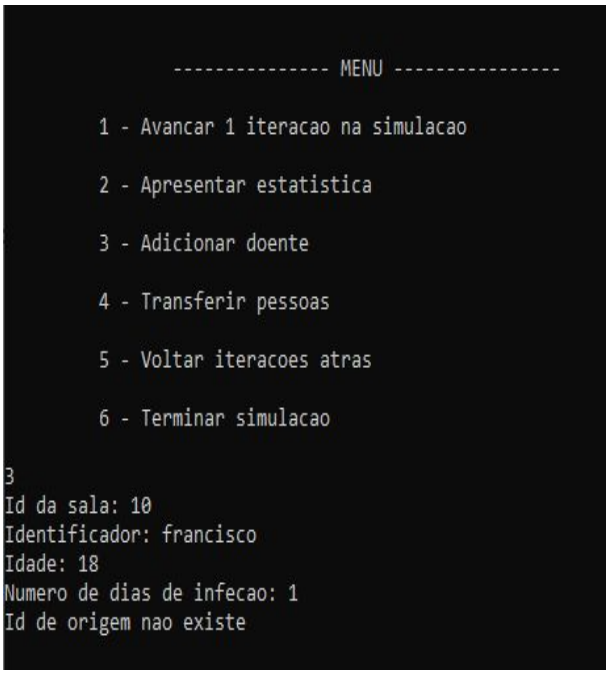


Fig.6 - Se tentar adicionar o doente numa sala não existente, ou com idade negativa ou dias de infecção negativa não vai ser adicionado.

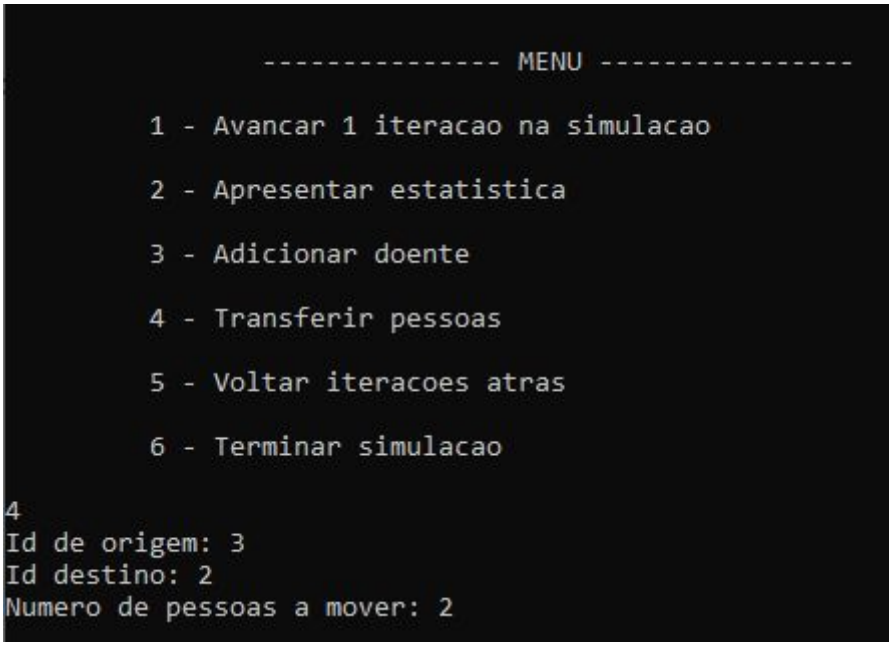
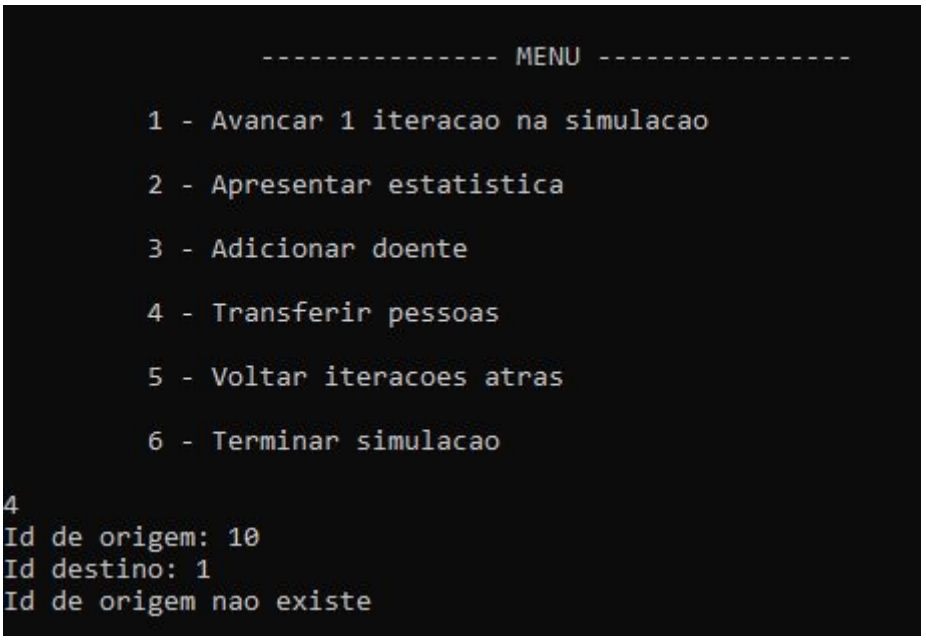


Fig. 7 - Transferir pessoas. Se o utilizador colocar id de origem não existente ou id de destino não existente nada irá acontecer. É também garantido que se tentar mover para uma sala que está cheia não terá efeito nenhum, se não houver pessoas no id de origem também não irá acontecer nada, e se tentar mover um número superior aos que estão no

local, irão ser movidos todos os que estão no local. É garantido que as pessoas no id origem são escolhidas aleatoriamente.



Conclusão

Com a realização deste projeto, aprofundei os meus conhecimentos na linguagem C, aprendendo a manipular listas dinâmicas, trabalhar com estruturas, leitura e escrita de ficheiros.
Gostava de ter feito a nova funcionalidade que foi pedida pelos docentes de forma recursiva, mas devido a falta de tempo não o realizei.
Estou bastante satisfeito com o resultado final deste projeto, e considero que adquiri as competências da cadeira de Programação.