Programação

Simulação da Propagação de Vírus

Docente: Francisco Pereira

José Fernando Esteves de Almeida nº 2019129077 – LEICE

Coimbra, 07 de Junho de 2020

Índice

1. Introdução

Este trabalho surge no âmbito da Unidade Curricular de Programação, no 2º Semestre do ano letivo de 2019/2020. Feito em linguagem C standard, respeitando a norma C99, tem como objetivos principais:

1. Simular, a um nível básico (cientificamente não preciso, e com intuito apenas didático), a simulação e propagação de um vírus numa população, que se encontra dividida por espaços/locais;
2. Ser de fácil utilização, através da implementação de uma interface simples e amigável, e que esclareça o utilizador em tudo o que pode fazer num determinado momento.

Para alcançar estes objetivos, foram usados conceitos da linguagem C, como estruturas dinâmicas, estruturas ligadas, leitura de ficheiros binários e leitura e escrita de ficheiros de texto, entre outros.

2. Código Fonte

O código fonte do programa encontra-se dividido por quatro ficheiros de código (extensão .c) e três ficheiros do tipo *header* (extensão .h), separados em menus e preparação, utilidades gerais, utilidades de estruturas e utilidades de ficheiros:

- main.c
  - Contém o "esqueleto" do programa: dois menus e a preparação da simulação;
  - Aqui se encontra a maioria do código com que o utilizador vai diretamente interagir.
- utils.c/.h
  - Para além das estruturas de dados, contêm as funções de randomização fornecidas;

        - Contêm as funções a que acede diretamente o menu de simulação (no main): avancarIteracao, adicionarDoente, transferirPessoas, recuarIteracoes e terminarSimulacao.
- utilStructs.c/.h
    - Contêm um bom número de funções que servem para o tratamento de estruturas de dados;
    - Estas incluem: as funções de verificação de informação (verificarPessoas e e verificarLocais), as funções que trabalham com as stat (gerarStat, apresentarStat, inserirStat), funções de libertação de memória (freeLigada, freeLocais, freeStat), entre outras.
- utilFiles.c/.h
    - Contêm as funções relacionadas com a manipulação de ficheiros: pedirFicheiro, lerBinario, lerTxt, criarFicheiro, listarFicheiro e verificarFicheiro.

3. Estruturas de Dados

Neste trabalho, foram utilizadas, no total, quatro estruturas diferentes:

- individuo - renomeada para pessoa e pPessoa;
- enunciado - renomeada para sler;
- sala - renomeada para local e plocal;
- estatistica - renomeada para stat e pstat;

Estas estruturas foram todas declaradas no ficheiro utils.h. As suas declarações e usos vão ser apresentados nas próximas páginas.

3.1 Estrutura individuo

```
typedef struct individuo pessoa, *pPessoa;
struct individuo {
   char id[MAX];
   int idade;
   char estado;
   int duracao;

   pPessoa prox;
};
```

A estrutura individuo é usada em várias listas ligadas ao longo do programa. Guarda a informação associada a uma pessoa e o ponteiro prox, que liga à próxima pessoa da lista.

3.2 Estrutura enunciado

```
typedef struct enunciado sler; // Usada apenas para a leitura do ficheiro .bin
struct enunciado {
   int id; // id numérico do local
   int capacidade; // capacidade máxima
   int liga[3]; // id das ligações (-1 nos casos não usados)
};
```

A estrutura enunciado é, como o nome indica, a estrutura que consta do enunciado do trabalho, embora com o seu nome mudado. Apenas serve para ler a informação do ficheiro binário, informação esta que é logo depois transferida da sler para um local.

3.3 Estrutura sala

```
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)

   pPessoa lista; // Lista das pessoas que se encontram no local
};
```

A estrutura sala é quase uma cópia da estrutura enunciado, apenas adicionando um campo: um ponteiro para pessoa, que permite associar uma lista ligada de pessoas a cada local. Tendo a população diretamente associada ao local desta forma, é fácil e rápido aceder e/ou modificar a informação necessária.

3.4 Estrutura estatistica

```
typedef struct estatistica stat, *pstat;
struct estatistica {
   plocal locais; // Informação da população e dos locais numa determinada iteração
   int tam; // Tamanho do vetor locais
   int nPessoas; // Numero de pessoas na iteracao
   int iteracao; // Iteracao a que se refere a estatistica
   float tSaudaveis, tInfetados, tImunes; // Taxa de pessoas saudáveis, infetadas e
imunes

   pstat prox;
};
```

A estrutura estatistica guarda todas as estatísticas/informação de iterações passadas, incluindo: vetor dinâmico dos locais (com listas ligadas de população associadas), número de pessoas da população, iteração a que a informação se refere e taxa de pessoas saudáveis, doentes e imunes.

Inclui também um ponteiro para outra estatística, de modo a possibilitar a formação de uma lista ligada de estatísticas.

4. Decisões

4.1 Estruturas Dinâmicas

Foram, ao longo deste trabalho, implementadas diferentes estruturas dinâmicas e alteradas várias vezes. Decidi, por fim, utilizar as estuturas de dados já referidas, formando as seguintes estruturas dinâmicas:

4.1.1 Vetor Dinâmico de Locais

Criado a partir da estrutura sala, armazena a informação relativa a todos os locais. Quando é criado, mantém-se com o ponteiro para pessoa a NULL, até lhe ser designada a população que vai conter na simulação (ver anexo 1). Toma a posição de estrutura mais importante do programa, sendo que é ela que guarda toda a informação necessária e é a ela que vamos fazer alterações diretas. De modo a manipular esta estrutura, as seguintes funções foram implementadas (entre outras):

- listPopulacao - Escreve a informação da população (de todos os locais do vetor) num ficheiro ou consola;
- designarPessoas - Função que associa cada pessoa da população aos respetivos locais que vão preencher durante a simulação. Funciona da seguinte forma: para cada pessoa, encontra aleatoriamente um local para a colocar, através da posição que ocupa no vetor. Se esse local estiver cheio, continua a tentar até encontrar um vazio. Se nunca encontrar um vazio, ignora toda a restante população (ou seja, se não houver espaço suficiente nos locais para toda a população, a população restante é ignorada). Se a capacidade do local for superior a zero, insere a pessoa nesse local (inserção no fim de uma lista ligada), e passa à próxima;
- verificarLocais - Efetua a verificação dos locais. As condições verificadas são as seguintes:
    - Não existência de locais com IDs repetidos ou negativos;

- Capacidades positivas de locais;
- Ligações a espaços existentes;
- Ligações mútuas a espaços;
- Não existência de ligações de espaços a eles próprios.

- freeLocais: Liberta a memória associada ao vetor dinâmico, mas apenas após libertar todas as listas ligadas associadas.

4.1.2 Lista Ligada de Pessoas (não designadas)

É criada ao ler o ficheiro de texto com a população. Apenas contém a população, não inserida em qualquer espaço. Está ordenada de forma igual ao ficheiro de onde foi retirada a informação. Tem dois usos: servir de apoio temporário, até a população ser designada e, antes disso, efetuar a verificação da população através da função verificarPessoas. As condições verificadas são as seguintes:

- Não existência de pessoas com IDs repetidos;
- Idades de pessoas positivas;
- Estados de saúde corretos: 'S', 'I' ou 'D' e, se doente, a duração da infeção ser positiva.

4.1.3 Lista Ligada de Estatísticas

Serve para guardar a informação da simulação, sendo criado um novo nó a cada nova iteração (incluindo logo após a fase de preparação).

A decisão de criar esta estrutura foi talvez das mais significantes do trabalho, pois, apesar de possivelmente ocupar grandes quantidades de memória se existir um elevado número de iterações guardadas, considero que os benefícios superam as desvantagens: o acesso a informações como o número de pessoas e as taxas é feito rapidamente e sem complicações de percorrer vetores e listas, logo o código torna-se mais rápido.

O uso desta estrutura possibilita, também, o fácil recuo de iterações, não só limitada ao recuo de três iterações, podendo assim gerar um relatório final de simulação mais informativo.

De modo a manipular esta estrutura, as seguintes funções foram implementadas:

- gerarStat: Cria uma cópia exata do vetor dinâmico de locais e calcula a restante informação necessária para gerar uma nova estatística: número de pessoas da população, taxas de infetados, doentes e saudáveis;
- inserirStat: Insere a estatística gerada no início da lista ligada. Considero que esta foi a melhor opção, pois é a que torna mais fácil o recuar de iterações: basta avançar nós na lista ligada, e carregar na simulação a informação da iteração para onde queremos recuar;
- apresentarStat: Apresenta a estatística escolhida na consola ou ficheiro;
- freeStat: Liberta a lista de ligada de estatísticas, incluindo a libertação de todos os locais e listas ligadas de pessoas.

4.2 Simulação

A simulação foi codificada da seguinte forma:

- Ao começar, verifica os ficheiros de pessoas e locais, através das estruturas já mencionadas. É feita a contagem do número de infetados, saudáveis e imunes, que serão associados a um vetor de 3 elementos nTotais, que vai sendo atualizado à medida que vão sendo infetados/curados novos indivíduos ao longo da simulação (facilita o cálculo das taxas). É também guardada uma cópia da informação da iteração (na lista ligada de stat);

- Ao avançar uma iteração, primeiro é percorrida toda a população e, para cada doente, é feita uma tentativa de cura, respeitando as regras fornecidas (inclusive a da chance do indivíduo se tornar imune). Se não se curar, é incrementado o contador de duração de infeção e é feita a disseminação da infeção no local onde está, se existirem pessoas suficientes. Depois de percorrer todos os indivíduos, gera e guarda a informação da nova iteração na lista ligada de stats. Nota: é possível que um indivíduo se cure e seja infetado novamente na mesma iteração, por outro doente;

- Para transferir pessoas, pede-se ao utilizador um local de origem, que vai ser pesquisado no vetor. Se não existirem, pede-se outro, até o utilizador fornecer um que exista. O mesmo acontece com o local de destino. Ao obter um local de destino que exista, verifica se esse o local de origem está ligado a ele (continua a pedir locais até o utilizador introduzir um que esteja ligado). Seguidamente, pede o número de pessoas ao utilizador. Se o local de origem tiver pessoas suficientes e o local de destino tiver capacidade suficiente, é feita a transferência das pessoas (aleatoriamente escolhidas). Se não, continua a pedir um número de pessoas, até ser possível. Por fim, é gerada e guardada novamente a informação da iteração alterada, que substitui a mais recente na lista ligada;

- Para recuar iterações, apenas avança na lista ligada de stats, e carrega a informação do vetor dinâmico dos locais. O vetor nTotais também é carregado, a partir das restantes informações da estrutura stat. Não considerei justificável, tendo em conta a estrutura usada, apenas poder recuar 3 iterações, optando por dar possibilidade ao utilizador de recuar quantas iterações desejar (desde que existam).

5. Manual de Utilização

5.1 Menu Pré-Preparação

Ao executarmos o programa, deparamo-nos com o primeiro menu (ver anexo 2): o menu pré-preparação. Contém várias opções que nos permitem criar, ler e verificar os ficheiros de locais e população, algo que foi muito útil no *debugging* e teste de código relacionado com a simulação em si.

Opções:

1. Iniciar Simulação: O programa leva-nos para a fase de preparação;
2. Criar Novo Ficheiro de Espaços: Pede ao utilizador toda a informação necessária à criação de um ficheiro de espaços/locais. No entanto, não efetua qualquer verificação, de modo a ser possível testar outras funções;
3. Listar Espaços: Apresenta na consola um ficheiro de espaços (escolhido pelo utilizador);
4. Verificar Ficheiros de Espaços: Verifica se um determinado ficheiro de espaços (escolhido pelo utilizador) se encontra no formato pretendido (locais com IDs não repetidos e positivos, capacidades positivas, ligações a espaços existentes, ligações mútuas e sem ligações de um espaço a ele próprio);
5. Criar Novo Ficheiro de Espaços: Pede ao utilizador toda a informação necessária à criação de um ficheiro de pessoas/população. No entanto, não efetua qualquer verificação, de modo a ser possível testar outras funções;
6. Listar Pessoas: Apresenta na consola um ficheiro de pessoas (escolhido pelo utilizador);
7. Verificar Ficheiros de Pessoas: Verifica se um determinado ficheiro de pessoas (escolhido pelo utilizador) se encontra no formato pretendido (pessoas com IDs não repetidos, idades positivas, estado ‘S’, ‘I’ ou ‘D’ e, se doente, a duração ser positiva);
8. Sair: Terminar o programa.

5.2 Fase de Preparação

Na fase de preparação, vão ser pedidos ao utilizador e verificados o ficheiro de locais e o ficheiro de pessoas que serão usados na simulação.

São efetuadas as verificações seguintes (devolve erro e a causa dele se pelo menos uma destas condições não for verdade):

1. Ficheiro de Locais:
    - Não existência de locais com IDs repetidos ou negativos;
    - Capacidades de locais positivas;
    - Ligações a espaços existentes;
    - Ligações mútuas a espaços;
    - Não existência de ligações de espaços a eles próprios.

2. Ficheiro de Pessoas:
    - Não existência de pessoas com IDs repetidos;
    - Idades de pessoas positivas;
    - Estados de saúde corretos: ‘S’, ‘I’ ou ‘D’;
    - Se doente, a duração da infeção ser positiva.

5.3 Menu Simulação

O segundo e último menu (ver anexo 3) é usado para manipular a simulação em si, oferecendo ao utilizador o leque de opções que pode tomar.

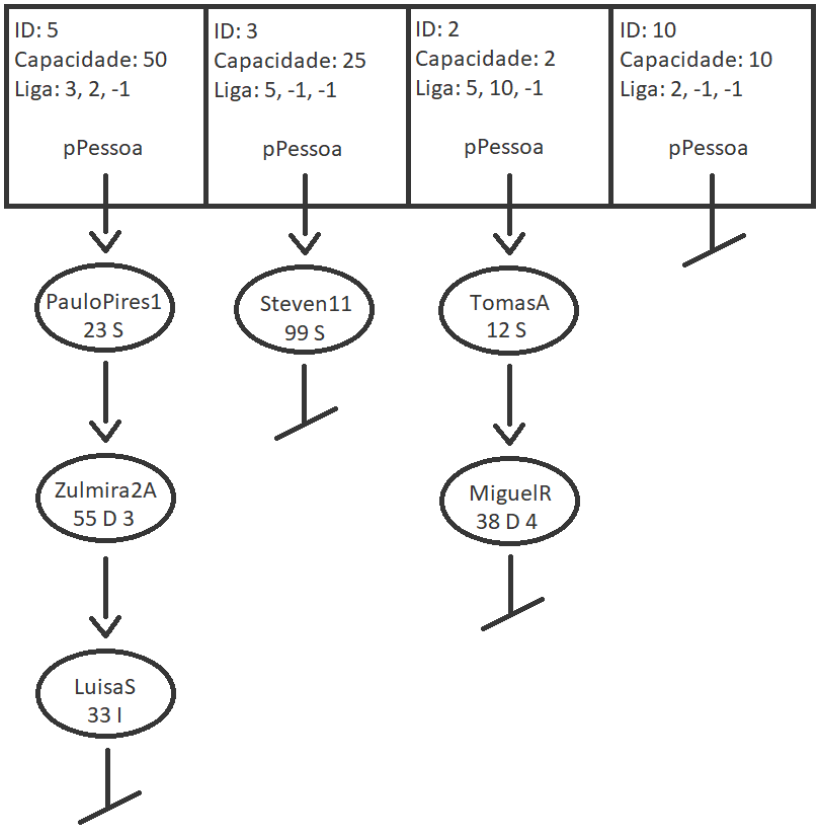
Opções:

1. Avançar 1 Iteração na Simulação: Como o nome indica, efetua todos os cálculos e manipulações necessárias (explicadas no ponto 4.2);
2. Apresentar Estatística: Apresenta a informação mais recente guardada na lista ligada de estatísticas (ponto 4.1.3);
3. Adicionar Doente: Adiciona um doente, cuja informação é dada pelo utilizador, a um local com capacidade suficiente (se o local de destino não tiver capacidade suficiente, informa o utilizador e pede um novo ID para local de destino). Também faz com que a estatística mais recente da lista ligada de estatísticas seja substituída pela nova, que inclui o novo doente;
4. Transferir Pessoas: Transfere um número de doentes de um local para outro. O local de origem, destino e número de doentes é pedido ao utilizador, e o programa verifica se essa transferência é possível e, no caso de não ser, informa o utilizador e pede nova introdução de dados;
5. Voltar Atrás X Iterações (Undo X): Pede ao utilizador um número X de iterações a recuar. Se não for possível recuar esse número de iterações, pede ao utilizador um novo número. Se for possível, atualiza o vetor das estatísticas, apagando as que já não são necessárias;
6. Terminar Simulação: Termina a simulação, gerando um ficheiro de texto com a população (cujo nome é fornecido pelo utilizador) e um ficheiro de texto *report.txt* com a informação da estatística mais recente da lista ligada, assim como:
    - Local com maior número de pessoas;
    - Local e iteração do maior número de doentes, imunes e saudáveis;
    - Iteração com a maior taxa de doentes, imunes e saudáveis.

6. Anexos

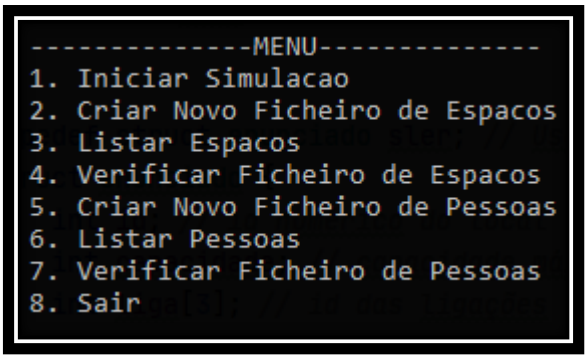
6.1 Anexo 1 – Vetor Dinâmico de Locais

Exemplo de um Vetor Dinâmico de Locais, com 4 locais e uma população já atribuída de 6 pessoas:



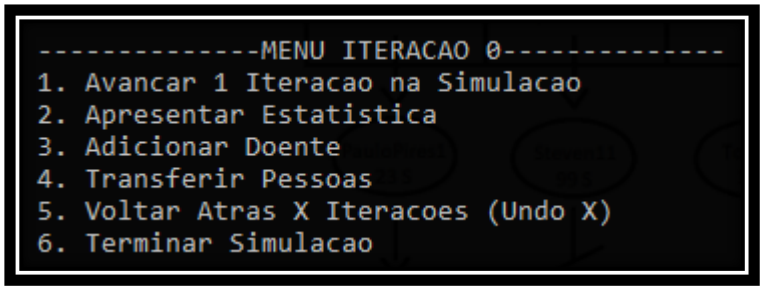
6.2 Anexo 2 – Menu Pré-Preparação

O Menu Pré-Preparação tem o seguinte aspeto:



6.3 Anexo 3 – Menu Simulação

O Menu Simulação tem o seguinte aspeto (onde '0' é a iteração atual):



7. Referências

- https://inforestudante.ipc.pt
- https://stackoverflow.com
- https://www.youtube.com

Consultados em maio e junho de 2020.