Instituto Superior de Engenharia de Coimbra

Programação

Trabalho Prático

Simulação da Propagação de Vírus

Licenciatura em Engenharia Informática

2019 / 2020

TheForgotten

someone@isec.pt

Conteúdo

1. Principais Estruturas de Dados

No programa existem 2 estruturas de dados principais: a struct person e a struct sala.

1.1. struct person

```
typedef struct person person, *nperson;

struct person {
    char name[100];
    int age;
    char state;
    int sickD;
    int idlocal;

    nperson next;
};
```

Esta estrutura serve para guardar dados sobre uma pessoa.

- `char name[100]` – nome (ou identificador alfanumérico) da pessoa com um máximo de 99 caracteres
- `int age` – idade dessa mesma pessoa
- `char state` – estado de saudade da pessoa em relação ao vírus
- `int sickD` – número de dias doente (se não estiver doente toma o valor de 0)
- `int idlocal` – id do local / sala frequentado pela pessoa (no caso de não frequentar nenhum espaço toma o valor de 0)
- `nperson next` – ponteiro para a próxima pessoa (usado na lista ligada; no caso de ser a última pessoa da lista toma o valor NULL)

1.2. struct sala

```
typedef struct sala local, *plocal;

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

Esta estrutura serve para guardar dados sobre um determinado local / sala do espaço onde a simulação vai ocorrer.

- `int id` – id do local
- `int capacidade` – número máximo de pessoas permitidas no local

- `int liga[3]` – id de outros locais aos quais este local está ligado (no caso de não existir ligação toma o valor de -1)

2. Estruturas dinâmicas implementadas

No programa existem apenas 2 estruturas dinâmicas: o vetor de structs que contém os locais do espaço e uma lista ligada para guardar a população da simulação.

Apesar de ter levantado alguns obstáculos achei que seria mais prático e mais eficiente trabalhar apenas com 2 estruturas dinâmicas para toda a simulação. Estas 2 guardam toda a informação essencial e tudo o resto é obtido a partir da manipulação das mesmas.

2.1. Vetor env[]

Este vetor de structs é responsável por guardar todos os locais / salas do espaço onde a simulação ocorre. Este é definido dinamicamente durante a leitura do ficheiro binário de espaço. Para poder acompanhar este vetor dinâmico é definido um inteiro `envSize` para guardar o tamanho do mesmo.

2.2. Listas Ligadas nextperson, nextperson1, nextperson2 e nextperson3

Estas listas ligadas são responsáveis por guardar a população. O nextperson é a lista ligada que representa a população no "presente" da simulação. As restantes representam, respetivamente, a população há 1 / 2 / 3 dias atrás.

A lista ligada inicial é obtida através da leitura do ficheiro de população. Nesta lista as pessoas são inseridas por ordem de leitura, ou seja, a primeira lida é a primeira pessoa da lista e a última lida é a última pessoa da lista.

Apesar de trazer dificuldades mais tarde, achei por bem manter a lista organizada desta maneira desde o início até ao final do programa. Decidi que seria mais correto fazer desta maneira para mexer o mínimo com as probabilidades (apesar de ser aleatório, nunca é verdadeiramente aleatório) caso rode o programa de novo com a mesma população.

3. Funções não usadas na versão final do programa

3.1. void printPeople(nperson p)

Esta função foi usada para debugging do programa. É responsável por imprimir uma lista ligada de structs person de modo a possibilitar descobrir se há algo de errado com a leitura do ficheiro ou se durante a simulação eram alterados parâmetros que se deviam manter inalteráveis.

3.2. void printEnv(local *v, int size)

Esta função foi usada para debugging do programa. É responsável por escrever na consola cada local presente no vetor espaço v de tamanho size de modo a facilitar perceber se há algo de errado com a leitura do ficheiro ou se durante a simulação é alterado algum dado deste vetor (que da maneira que o programa esta concebido deve manter-se inalterado).

4. EnvironmentGenerator

Este miniprograma serve para gerar ficheiros binários de espaço. Foi feito principalmente para facilitar a criação de espaços novos para poder dar debug no programa principal. Este não faz quaisquer verificações nos dados inseridos, ou seja, permite que sejam inseridos locais com o mesmo ID ou com IDs não positivos, por exemplo.

5. Mini Manual de Utilização

Apesar de a interface ser bastante user friendly e simples, há alguns aspetos relevantes a notar.

Este programa começa por pedir o nome dos ficheiros a ler para a população e outro para o espaço. Tal como é indicado pelo programa, o nome deve incluir a extensão do ficheiro. No caso de haver erro de leitura o programa fecha imediatamente.

No final de conseguir ler com sucesso os ficheiros e verificar que está tudo bem com eles, este finalmente inicia a fase de simulação. Nesta fase a consola mostra-nos um menu que contém todas as opções disponíveis durante a simulação de maneira simples e direta.

Na primeira linha temos o dia em que a simulação está de momento e, de seguida, temos os menus com as seguintes opções:

1. Avançar uma iteração na simulação – esta opção avança um dia na simulação e aplica o modelo de propagação do vírus fornecido
2. Apresentar estatística – apresenta um resumo do estado atual do espaço tanto em números como em percentagem

3. Adicionar doente – adiciona um doente a um local à escolha (verifica, entre várias coisas, por exemplo, que o nome (identificador alfanumérico) do doente é único e ainda não existe)
4. Transferir pessoas – transfere aleatoriamente um número de pessoas indicado dum local para o outro (para isto acontecer os locais devem ter ligação)
5. Voltar atrás X iterações (máximo 3) – aqui podemos recuar no máximo 3 dias na simulação
6. Terminar simulação – como o nome indica, termina a simulação e solicita ao utilizador um nome (com extensão) para o ficheiro que vai conter a população atual