Programação

Trabalho Prático

LEI; LEI-PL; LEI-CE: 1º ano - 2º semestre 2019/2020

Simulação da Propagação de Vírus

Notas prévias:

- O modelo descrito não pretende simular de forma precisa a propagação do COVID-19. É apenas um enunciado do trabalho prático de Programação.
- O enunciado é possivelmente vago, genérico e incompleto em alguns pontos. O que se pretende é que os alunos avaliem as opções existentes e escolham a que considerarem mais apropriada para cada uma das situações com que se depararem. Todas as escolhas devem ser referidas e justificadas no relatório.
- O programa deve ser implementado em C *standard*, *i.e.,* não deve ter instruções que o tornem específico para um determinado ambiente/plataforma de desenvolvimento. Deverá ser respeitada a norma C99.
- O programa entregue deve ter uma interface simples e amigável, indicando o que pode ser feito em cada situação. Não são valorizados programas com interfaces gráficos.
- Deve distribuir o código fonte por vários ficheiros. Para além do ficheiro com código disponibilizado com este enunciado, deverão existir, no mínimo, outros dois ficheiros com código fonte.
- Deverá utilizar *header files* para gerir as dependências entre os vários ficheiros com código fonte.
- Todos os ficheiros devem ter a identificação do aluno (nome completo e número), em comentário, nas linhas iniciais dos ficheiros.

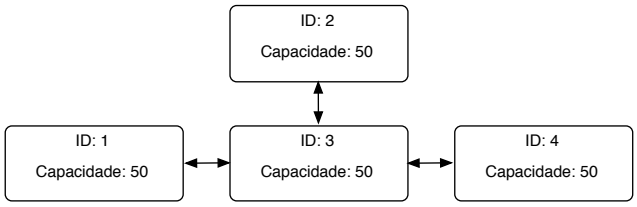
1. Introdução

Pretende-se que seja implementado um programa que simule, de forma simplificada, a propagação de um vírus entre uma população. Os indivíduos da população encontram-se num espaço constituído por locais, interligados entre si. A execução do programa consiste em 2 fases sequenciais:

1. **Preparação:** No início da execução, a aplicação carrega todos os dados necessários a partir de ficheiros: Configuração do espaço e dimensão e características da população inicial.
2. **Simulação:** Quando termina a fase de preparação, inicia-se um processo de simulação iterativo que representa a propagação do vírus ao longo do tempo (1 iteração de cada vez, correspondendo cada uma destas iterações a 1 dia). No final da simulação, o programa apresenta um resumo dos resultados e grava informação relevante em ficheiros.

2. Espaço

O espaço em que decorre a simulação é constituído por um conjunto de locais interligados entre si, por exemplo, várias salas de um mesmo edifício.



A figura mostra um exemplo de um espaço. É constituído por 4 locais, existindo algumas ligações bidirecionais entre eles. Cada local tem um *id* numérico positivo (único), uma capacidade máxima e algumas ligações diretas a outros locais. No máximo, cada local pode ter 3 ligações.
Os possíveis espaços para realizar a simulação estão armazenados em **ficheiros binários** (1 ficheiro por espaço). Cada um destes ficheiros contém várias estruturas do tipo *local*, uma para cada um dos locais existentes nesse espaço:

```
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)
};
```

No início da execução, o utilizador indica o nome do ficheiro onde se encontra a informação do espaço a usar. Antes de se iniciar a fase de simulação, esta informação é obrigatoriamente transferida para um **vetor dinâmico de estruturas**.
O programa deve validar a organização do espaço: todos os locais devem ter *ids* positivos únicos e as ligações devem estar corretas (por exemplo, se o local A tiver ligação para o local B, então B também deve ter ligação para A). Caso exista algum erro detetado na validação, deverá ser apresentada uma mensagem adequada e o programa termina sem iniciar a fase de simulação.
Juntamente com o enunciado são disponibilizados 3 ficheiros binários com possíveis espaços para simulação. São apenas exemplos e durante a defesa poderão ser utilizados outros ficheiros, desde que sigam a mesma organização indicada neste enunciado.

3. Pessoas

Existe um conjunto inicial de pessoas que deve ser distribuída pelos locais onde será realizada a simulação. Cada uma das pessoas deste conjunto inicial tem as seguintes características:
- Identificador único alfanumérico (1 palavra)
- Idade
- Estado: pode ser **S**audável, **D**oente, **I**mune
- Caso esteja doente, existe a informação sobre há quantos dias foi infetado

Ficheiros de texto armazenam informação sobre populações de pessoas. Cada **ficheiro de texto** tem obrigatoriamente a informação armazenada no formato descrito a seguir.
Uma linha tem toda a informação de uma pessoa, de acordo com a seguinte ordem: identificador, idade, estado (S, D ou I). Caso o estado seja D, surge ainda um inteiro

indicando há quantos dias foi infetado. A informação numa linha está separada por um ou mais espaços em branco. Os dados sobre as várias pessoas surgem em linhas consecutivas. A seguir ilustra-se um exemplo de um ficheiro de texto com 6 pessoas.

```
PauloPires1   23   S
AnaLebre34A   55   I
Tomas111      12   S
PauloPires2   67   D    10
LuisaSantos   40   D    3
Zulmira2A     17   S
```

Durante a fase de preparação, o utilizador indica o nome do ficheiro de texto onde se encontra a informação das pessoas com as quais se vai iniciar a simulação. Cada uma destas pessoas deve ser colocada num dos locais existentes. A seleção do local para cada pessoa deve ser feita de forma aleatória, satisfazendo a restrição de capacidade. Se existirem mais pessoas do que a capacidade total do espaço, algumas destas pessoas não participam na simulação.

Durante a execução, a informação sobre as pessoas que estão a participar na simulação está armazenada em **estruturas dinâmicas do tipo lista ligada**. Fica ao critério do aluno escolher a melhor organização para as estruturas dinâmicas, sujeita apenas às restrições do enunciado.

O programa deve validar a informação lida do ficheiro de pessoas. Caso esteja num formato inválido, deverá ser apresentada uma mensagem adequada e o programa termina sem iniciar a fase de simulação.

Juntamente com o enunciado são disponibilizados 2 ficheiros de texto com possíveis populações para simulação.

4. Configuração do Modelo de Propagação do Vírus

Para que o modelo de propagação do vírus possa ser simulado ao longo do tempo (os detalhes são apresentados na secção 5), é necessário que sejam definidos alguns parâmetros. Este modelo tem os seguintes parâmetros, cujos valores são fixos[1]:

- Taxa de disseminação: Número de pessoas que um doente infeta em cada iteração da simulação. Neste modelo, o valor corresponde a 5% do número de pessoas que se encontram no mesmo local (valor arredondado para baixo). Por exemplo, numa sala com 75 pessoas, um doente transmite o vírus a 3 outras pessoas em cada iteração da simulação. As pessoas afetadas são escolhidas aleatoriamente, de entre todas as que se encontram no local. Pode acontecer que o vírus seja transmitido a outra pessoa que já está doente. Neste caso, a infeção não tem qualquer efeito.
- Probabilidade de recuperação: Probabilidade que um doente tem de recuperar em cada iteração da simulação. Neste modelo, a probabilidade é 1/idade, ou seja, para um doente com 50 anos, a probabilidade de recuperar em cada iteração é de 1/50 = 0.02.
- Duração máxima da infeção: Número de dias em que uma pessoa está doente. No final deste período, qualquer pessoa volta a ficar saudável. Neste modelo, o valor é de 5 + 1 dia por cada dezena de anos de vida (arredondado para baixo). Por exemplo, uma pessoa infetada com 45 anos fica curada ao fim de 9 dias.
- Taxa de imunidade: Quando uma pessoa infetada fica curada (por recuperação ou por ter ultrapassado a duração máxima da infeção), existe a probabilidade de ficar

[1] Estes valores são constantes e apenas poderão ser alterados através de alteração direta no código e recompilação do programa.

imune à doença. Se isto acontecer, mesmo que volte a ser infetado no futuro, não contrai novamente a doença. Neste modelo, a taxa de imunidade é de 20%.

5. Simulação do Modelo de Propagação do Vírus

A fase de simulação inicia-se depois de terminar a leitura de informação do espaço e do grupo de pessoas a considerar. A simulação decorre ao longo de várias iterações sucessivas, cada uma delas correspondendo a um dia. Durante a fase de simulação, a aplicação deve disponibilizar um menu com, pelo menos, as seguintes opções:

1. **Avançar 1 iteração na simulação**: a aplicação aplica o modelo de propagação à população e atualiza o estado das pessoas.
2. **Apresentar estatística:** a aplicação apresenta dados estatísticos do estado atual da simulação. Entre outros resultados, deve mostrar: distribuição das pessoas por sala, taxa de saudáveis, taxa de infetados, taxa de imunes, ...
3. **Adicionar doente:** 1 nova pessoa doente é adicionada a um local. O utilizador especifica o *id* do local e o identificador, a idade e o número de dias de infeção da nova pessoa.
4. **Transferir pessoas:** N pessoas são movidas de um local para outro dentro do espaço, desde que estes locais tenham uma ligação direta. Os *ids* dos locais de origem e de destino e o valor N são indicados pelo utilizador (por exemplo, mover 3 pessoas do local 1 para o local 3). O programa escolhe N pessoas de forma aleatória, de entre as que se encontram no local de origem.
5. **Terminar Simulação:** Ao terminar a simulação, a aplicação gera um relatório final. Fica ao critério do aluno decidir que informação deve ser incluída, mas convém que seja completa. Este relatório é guardado num **ficheiro de texto** com nome *report.txt*. Além disso, a população existente na última iteração deve ser guardada num ficheiro de texto. O nome deste ficheiro é indicado pelo utilizador.

6. Código Disponibilizado

O ficheiro *utils.c* contém algumas funções auxiliares que podem ser úteis durante a implementação do trabalho:

void initRandom();

Inicializa o gerador de números aleatórios. Deve ser chamada apenas uma vez, no início da execução do programa.

int intUniformRnd(int a, int b);

Devolve um valor inteiro aleatório distribuído uniformemente no intervalo *[a, b]*.

int probEvento(float prob);

Devolve o valor 1 com probabilidade *prob*. Caso contrário, devolve 0.

O ficheiro *utils.c* contém igualmente uma função *main()* que serve apenas para ilustrar alguns exemplos de chamadas das funções disponibilizadas.

7. Normas para a realização do trabalho

O trabalho deve ser **realizado individualmente**. O trabalho só pode ser entregue uma vez e a nota obtida é válida em todas as épocas de avaliação do ano letivo 2019/2020.

Data provisória para entrega do trabalho prático: **23.55** do dia **7 de Junho de 2020**.

Material a entregar:

- **Até às 23.55 do dia 07/06/2020**: entregar através do InforEstudante um ficheiro compactado em formato ZIP, contendo o relatório, o código fonte comentado e os ficheiros de dados necessários para o funcionamento do programa. Caso tenham usado o Netbeans ou o CodeBlocks para a implementação, deverão incluir no ZIP todos os ficheiros do respetivo projeto.
- O nome do ficheiro ZIP deve obrigatoriamente ter o seguinte formato: `Prog_pX_NumAluno.zip`, em que X é a identificação da turma prática que frequenta.

- **No início da defesa**: entregar ao professor responsável pela defesa uma cópia impressa do relatório. O conteúdo do documento deve ser igual ao submetido anteriormente.

Defesa:

Os trabalhos serão sujeitos a **defesa obrigatória,** em data a anunciar. As defesas poderão incluir:

i) Demonstração do funcionamento do programa
ii) Explicação detalhada do código
iii) Implementação de alterações / novas funcionalidades

As defesas serão feitas nos computadores dos laboratórios, utilizando o Netbeans ou o Codeblocks em ambiente Windows. Antes da submissão deverão certificar-se que o trabalho submetido está pronto para correr num destes ambientes.

Relatório

Deve ser entregue um relatório contemplando os seguintes pontos:

- Apresentação das principais estruturas de dados, justificando as escolhas feitas
- Apresentação detalhada das estruturas dinâmicas implementadas
- Justificação para as opções tomadas em termos de implementação
- Pequeno manual de utilização.

Avaliação

A cotação do trabalho é de **6 valores**.

Esta componente da avaliação não tem nota mínima.

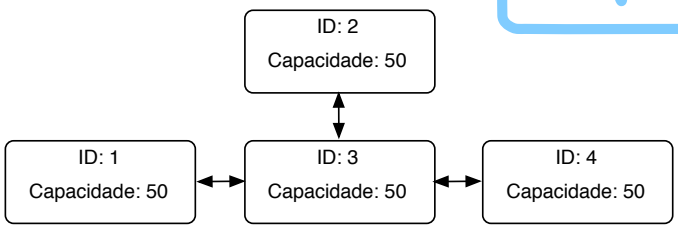
A deteção de plágio parcial ou total implica a anulação imediata de todos os trabalhos envolvidos.

Critérios de Avaliação para as Funcionalidades Implementadas

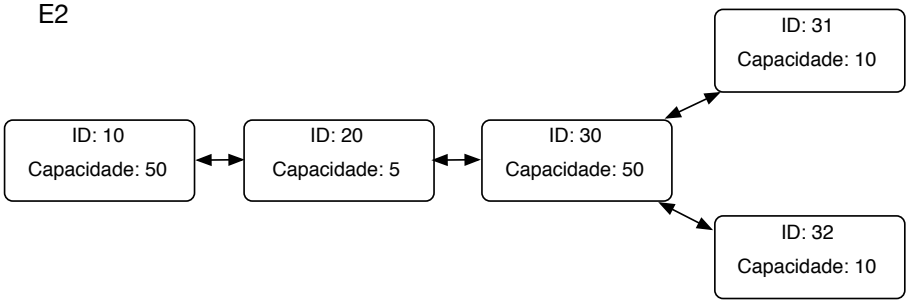
- Definição das estruturas de dados
- Correção das funcionalidades implementadas
- Manipulação de estruturas dinâmicas
- Manipulação de ficheiros
- Simplicidade/funcionalidade do interface com o utilizador

Anexo: Mapas dos locais armazenados nos 3 ficheiros binários disponibilizados como exemplos

E1



E2



E3