

Instituto Superior de Engenharia de Coimbra

Eng. Informática

2019/2020



Programação

Trabalho Prático – Simulação da Propagação de Vírus

Realizado por:

Diogo Miguel Pinto Pascoal – 2018019825

Conteúdo

Capítulo 1 – Manual de Utilização.....	3
1.1 – Avançar Iteração.....	5
1.2 – Adicionar Doente	6
1.3 – Estatísticas.....	6
1.4 – Transferir Pessoas.....	7
1.5 – Anular Iterações.....	7
1.6 – Sair.....	8
Capítulo 2 – Principais Estruturas de Dados	10
2.1 – Pessoa.....	10
2.2 – Local.....	10
2.3 – PessoaLocal.....	11
2.4 – TimeMachine.....	11
Capítulo 3 – Principais Estruturas Dinâmicas.....	11
3.1 – main	11
3.1.1 – pLocal espaco.....	11
3.1.2 – pPessoa listaPessoas.....	11
3.1.3 – pTimeMachine timemachine.....	12
3.1.4 – pPessoa listaOGPessoas	12
3.2 – Simulacao.....	12
3.2.1 – pPessoaLocal listaPessoasLocal	12
Capítulo 4 – Justificação para as opções tomadas em termos de implementação.....	13

Capítulo 1 – Manual de Utilização

O programa apresenta uma utilização muito simples e intuitiva: Ao iniciar o programa, somos apresentados com um pedido para inserir o nome do ficheiro onde se encontram os locais para a simulação:



```
trabalhop_diogopascoal_2018019825
*****
A PREPARAR LOCAIS
*****
Insira o nome do ficheiro dos locais:
```

Figura 1 - Inserir nome do ficheiro dos locais

Basta inserir o nome do ficheiro (ou o caminho caso este não se encontre na mesma pasta que o executável) e o programa irá prosseguir para a preparação das pessoas.

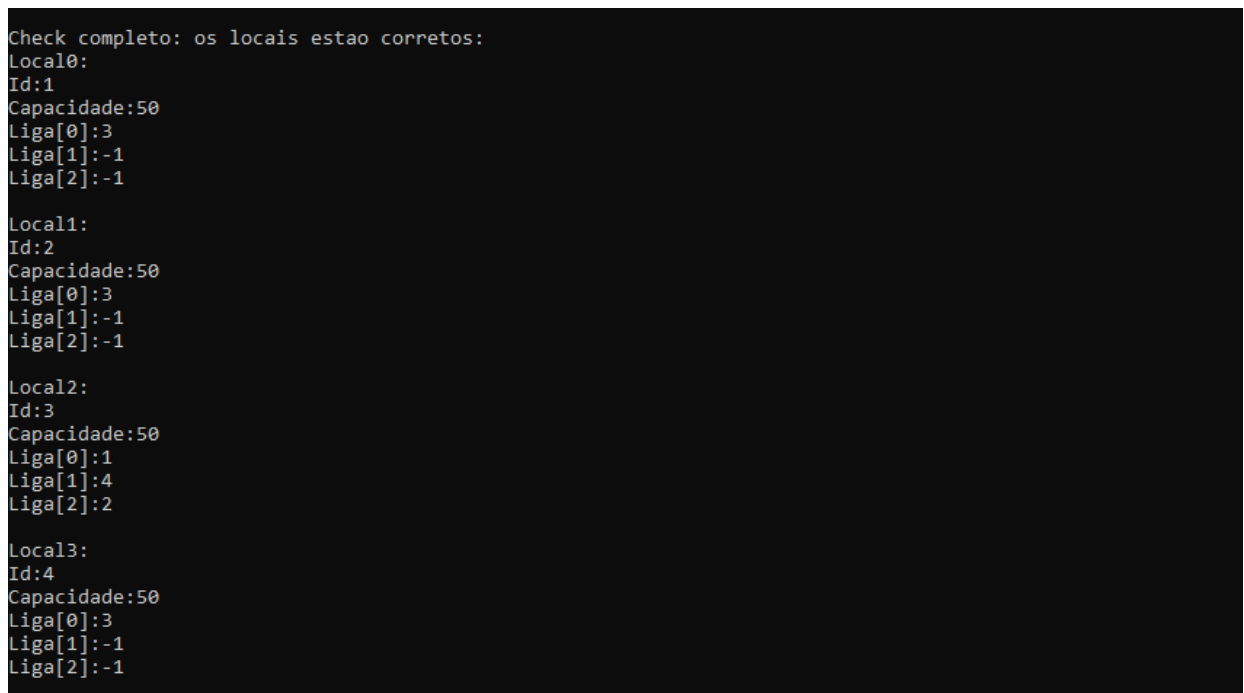
Deste modo, é apresentado um novo ecrã a pedir desta vez o nome do ficheiro onde se encontram guardadas as informações relativas às pessoas:



```
trabalhop_diogopascoal_2018019825
*****
A PREPARAR PESSOAS
*****
Insira o nome do ficheiro das pessoas:
```

Figura 2 - Inserir nome do ficheiro das pessoas

Depois de inseridos os nomes dos dois ficheiros necessários para a inicialização do programa, será apresentada uma lista detalhada dos locais e das pessoas.



```
Check completo: os locais estao corretos:
Local0:
Id:1
Capacidade:50
Liga[0]:3
Liga[1]:-1
Liga[2]:-1

Local1:
Id:2
Capacidade:50
Liga[0]:3
Liga[1]:-1
Liga[2]:-1

Local2:
Id:3
Capacidade:50
Liga[0]:1
Liga[1]:4
Liga[2]:2

Local3:
Id:4
Capacidade:50
Liga[0]:3
Liga[1]:-1
Liga[2]:-1
```

Figura 3 - Informação sobre os locais lidos

```
*****
ID:PauloPires2
Idade:67
Estado:D
Probabilidade Recuperacao:0.010000
Duracao Max Infecao:11
Local Atribuido:3
Dias infetado:10
*****
ID:MarcoMarquesAAA
Idade:40
Estado:I
Probabilidade Recuperacao:0.020000
Duracao Max Infecao:9
Local Atribuido:2
*****
ID:Zulmira2A
Idade:17
Estado:S
Probabilidade Recuperacao:0.050000
Duracao Max Infecao:6
Local Atribuido:2
*****
ID:Olivia1
Idade:12
Estado:S
Probabilidade Recuperacao:0.080000
Duracao Max Infecao:6
Local Atribuido:4
```

Figura 4 - Excerto da Informação sobre as pessoas lidas

De realçar que o programa coloca aleatoriamente as pessoas nos diferentes locais disponíveis. Caso não exista espaço para mais pessoas, o programa irá avisar e ignorar as pessoas que se encontram em excesso.

Após esta apresentação extensiva dos detalhes que o programa recebeu, será recebido com um menu que lhe permite realizar diversas ações:

1. Avançar Iteração
2. Adicionar Doente
3. Estatísticas
4. Transferir Pessoas
5. Anular Iterações
6. Sair

```
Escolha uma opcao:
1 - Avancar uma iteracao
2 - Adicionar Doente
3 - Estatisticas
4 - Transferir pessoas
5 - Anular Iteracoes
6 - Sair
Opcao:
```

Figura 5 – Menu

1.1 – Avançar Iteração

Esta é a função principal do programa. O programa irá simular o avanço do vírus segundo as condições estabelecidas no enunciado. Irá efetuar a simulação isoladamente a cada local, sendo que numa primeira fase irá processar o ato de infetar outras pessoas: as pessoas doentes vão infetar uma certa quantidade de pessoas, quantidade essa definida pela constante *TaxaDeDisseminacao* que corresponde a 5% da quantidade total de pessoas presente no respetivo local¹. Caso o local tenha apenas 1 ou nenhuma pessoa, não irá efetuar este processo visto não haver ninguém para infetar.

O programa irá informar o utilizador de todas as tentativas de infetar alguém, seja esta tentativa efetuada com sucesso ou não, informando porquê é que não foi possível infetar.

De seguida passamos à fase de recuperação, em que o programa irá verificar se as pessoas conseguem curar-se naturalmente. A probabilidade deste evento é calculada segundo a seguinte fórmula: $\frac{1}{idade}$. Esta função está expressa no enunciado. Caso o evento aconteça, existe a possibilidade da pessoa ficar imune ao vírus, sendo que essa probabilidade está definida na constante *TaxaImunidade* presente no ficheiro *main.c*. Também existe um número máximo de dias que uma pessoa pode ficar infetada, definida pela fórmula $duracaoMaxInfecao = 5 + \frac{idade}{10}$, fórmula esta também definida no enunciado do trabalho prático.

```
Escolha uma opcao:
1 - Avancar uma iteracao
2 - Adicionar Doente
3 - Estatísticas
4 - Transferir pessoas
5 - Anular Iteracoes
6 - Sair
Opcao: 1

Opcao 1 escolhida
*****
Local [1]:

Existem de momento 4 pessoas doentes neste local
Cada pessoa vai ter que infetar 0 pessoas
*****
A pessoa com id HugoB ficou curada passado o montante maximo de dias!
*****
Local [2]:
0 local tem apenas 0 ou 1 pessoa, impossivel infetar
*****
Local [3]:

Existem de momento 3 pessoas doentes neste local
Cada pessoa vai ter que infetar 0 pessoas
*****
*****
Local [4]:

Existem de momento 1 pessoas doentes neste local
Cada pessoa vai ter que infetar 0 pessoas
*****
A pessoa com ID PauloPires2 ficou mais um dia infetada
A pessoa com ID SamuelSimoes ficou mais um dia infetada
A pessoa com ID SandraS ficou mais um dia infetada
A pessoa com ID HugoA ficou mais um dia infetada
A pessoa com ID HugoC ficou mais um dia infetada
A pessoa com ID PauloAlves123 ficou mais um dia infetada
A pessoa com ID Carlos1 ficou mais um dia infetada
A pessoa com ID CarlaCardoso ficou mais um dia infetada
```

Figura 6 - Avançar Iteração

¹ Nota: O valor é sempre arredondado para baixo, sendo que se o número de pessoas for abaixo de 20, os doentes não irão infetar ninguém.

1.2 – Adicionar Doente

Assim como o nome indica, esta função permite ao utilizador de criar uma nova pessoa (cujo estado é automaticamente “Doente”) e adicioná-la a um local. Para realizar esta ação, o programa irá pedir ao utilizador as seguintes informações:

- ID do doente
- Idade
- Tempo que esta pessoa se encontra doente
- ID do local onde se deseja inserir o novo doente

```
Escolha uma opcao:
1 - Avancar uma iteracao
2 - Adicionar Doente
3 - Estatisticas
4 - Transferir pessoas
5 - Anular Iteracoes
6 - Sair
Opcao: 2

Opcao 2 escolhida
Inserir novo doente:
Escreva o ID (max 30 char): paciente1

Digite a idade do doente: 20

Ha quantos dias se encontra o doente infetado? 5

Insira o local onde quer inserir o novo doente: 1

Li esta pessoa:
*****
ID:paciente1
Idade:20
Estado:D
Probabilidade Recuperacao:0.050000
Duracao Max Infecao:7
Local Atribuido:1
Dias infetado:5
```

Figura 7 - Adicionar Doente

O programa irá verificar se a informação está correta. Caso esteja então o doente é adicionado à lista.

1.3 – Estatísticas

Esta função fornece ao utilizador a percentagem de pessoas em cada um dos locais no espaço da simulação, assim como a percentagem de pessoas saudáveis, doentes e de pessoas imunes.

```
Escolha uma opcao:
1 - Avancar uma iteracao
2 - Adicionar Doente
3 - Estatisticas
4 - Transferir pessoas
5 - Anular Iteracoes
6 - Sair
Opcao: 3

Opcao 3 escolhida
Percentagem de pessoas por local:
0 local 1 tem 40.91 % das pessoas
0 local 2 tem 18.18 % das pessoas
0 local 3 tem 13.64 % das pessoas
0 local 4 tem 27.27 % das pessoas

*****

Percentagem de pessoas saudaveis: 45.45 %
Percentagem de pessoas infetadas: 45.45 %
Percentagem de pessoas Imunes: 9.09 %
```

Figura 8 – Estatísticas

1.4 – Transferir Pessoas

Com esta função, o utilizador pode transferir pessoas de um local para outro.

Esta função transfere N pessoas de um local para outro, seguindo as seguintes restrições: os locais têm que ter ligação direta e o utilizador apenas pode escolher quantas pessoas serão transferidas, sendo que as pessoas que irão de facto ser transferidas serão escolhidas aleatoriamente.

Ao escolher esta funcionalidade do menu, o programa irá requisitar a seguinte informação:

- ID do local de Origem
- ID do local de Destino
- Número de pessoas a transferir

O programa irá confirmar a informação que recebeu, verificar se os locais têm ligação, e efetuar a transferência.

```
Escolha uma opcao:
1 - Avancar uma iteracao
2 - Adicionar Doente
3 - Estatisticas
4 - Transferir pessoas
5 - Anular Iteracoes
6 - Sair
Opcao: 4

Id Origem: 1

Id destino3

Quantas pessoas: 2

Numero de pessoas a transferir = 2
Ja transferi as pessoas
Ja alterei a capacidade dos locais
```

Figura 9 - Transferir Pessoas

1.5 – Anular Iterações

Esta funcionalidade permite ao utilizador anular iterações passadas, tendo um limite de viajar até 3 iterações atrás. O estado das pessoas e os locais serão armazenados antes de efetuar uma iteração e será possível deste modo aplicar este estado anterior através desta função.

Ao seleccionar esta opção do menu, o programa irá informar de quantas iterações é possível anular.

```
Escolha uma opcao:
1 - Avancar uma iteracao
2 - Adicionar Doente
3 - Estatisticas
4 - Transferir pessoas
5 - Anular Iteracoes
6 - Sair
Opcao: 5

Ainda nao pode voltar atras
```

Figura 10.1 - Anular Iterações - Nenhuma Iteração efetuada

```

Escolha uma opcao:
1 - Avancar uma iteracao
2 - Adicionar Doente
3 - Estatisticas
4 - Transferir pessoas
5 - Anular Iteracoes
6 - Sair
Opcao: 5

Escolha uma opcao:
0 - Sair
1 - Voltar 1 iteracao atras
2 - Voltar 2 iteracoes atras
3 - Voltar 3 iteracoes atras
Escolha uma opcao:

```

Figura 11 - Anular Iterações - 1 ou mais iterações efetuadas

Depois de escolher uma opção, o programa irá reverter para a iteração escolhida. De notar que o uso de qualquer função que não seja a primeira, “Avançar uma iteração”, não cria uma instância para esta funcionalidade, pelo que apenas ao ativar a referida função é que se cria uma cópia para se poder reverter as alterações. As alterações também são perdidas caso o programa seja fechado (mesmo que a lista de pessoas guardada no fim do programa seja reaplicada).

1.6 – Sair

Ao sair do programa através desta opção, são realizadas duas tarefas.

Em primeiro lugar, é criado um relatório, cujo nome será report.txt, com algumas informações principais da simulação realizada, sendo que este relatório segue o seguinte formato:

Número de pessoas no início da simulação: XX

Número de pessoas no final da simulação: XX

Estatísticas:

Início da simulação:

Percentagem de pessoas por local:

O local X tem XX % das pessoas

O local Y tem XX % das pessoas

(...)

Percentagem de pessoas saudáveis: XX %

Percentagem de pessoas infetadas: XX %

Percentagem de pessoas Imunes: XX %

Final da Simulação:

Percentagem de pessoas por local:

O local X tem 31.82 % das pessoas

O local Y tem 18.18 % das pessoas

(...)

Percentagem de pessoas saudáveis: XX %

Percentagem de pessoas infetadas: XX %

Percentagem de pessoas Imunes: XX %

Lista Locais

Local0:

Id:X

Capacidade: X

Liga[0]: X

Liga[1]: -1

Liga[2]: Y

(...)

Lista Inicial de Pessoas:

ID:Pessoa1

Idade:XX

Estado:XX

Probabilidade Recuperação: 0.XXX

Duração Max Infeção:X

Local Atribuido:X

(...)

Lista Final de Pessoas:

(...)

FIM RELATÓRIO

Também é realizada uma cópia da lista final das pessoas, guardando toda a informação que também está presente no relatório **exceto o local atribuído**. Isto deve-se ao facto que é sempre atribuído um local aleatório ao ler a informação relativa às pessoas no início do programa como antes referido. O nome desta cópia é definido pelo utilizador, sendo que ao chamar a funcionalidade aparece uma mensagem a pedir o nome a registar.

```
Escolha uma opcao:
1 - Avancar uma iteracao
2 - Adicionar Doente
3 - Estatisticas
4 - Transferir pessoas
5 - Anular Iteracoes
6 - Sair
Opcao: 6

Insira o nome do ficheiro para guardar a lista das pessoas:
```

Figura 12 - Pedido para nome do ficheiro da lista das pessoas

Após inserir o nome do ficheiro o programa termina e fecha a consola.

Capítulo 2 – Principais Estruturas de Dados

Neste capítulo será feita uma apresentação breve das diversas estruturas de dados presentes no programa, juntamente com uma breve justificação para a criação das mesmas.

2.1 – Pessoa

Cada estrutura deste tipo contém as seguintes componentes:

- id – char com 30 caracteres no máximo onde se guarda um identificador da Pessoa
- idade – Inteiro com a idade da Pessoa
- estado – Pode ter 1 dos 3 seguintes caracteres: ‘S’, ‘D’ e ‘I’, sendo o significado dos mesmos “Saudável”, “Doente” e “Imune”, respetivamente
- dias_infetado – Caso a pessoa esteja doente, esta variável contém a quantidade de dias deste que esta pessoa foi infetada
- idLocal – Esta variável é inicializada ao iniciar o programa, que irá guardar no percurso do programa o local que lhe foi atribuído
- probabilidadeRecuperacao – Contém a probabilidade em percentagem calculada da pessoa se curar naturalmente
- duracaoMaxInfecao – Duração máxima em dias que a pessoa pode estar doente, caso ultrapasse este valor tem a possibilidade de voltar a estar saudável ou de se tornar imune
- nextPessoa – Ponteiro para a próxima pessoa

Trata-se de uma estrutura usada numa lista ligada, daí a presença do ponteiro nextPessoa.

Esta estrutura irá representar individualmente cada pessoa / paciente presente na simulação.

As componentes id, idade, estado e dias_infetado são todas fornecidas no ficheiro texto da lista das pessoas. Acrescentei a esta estrutura a probabilidadeRecuperacao, duracaoMaxInfecao devido ao facto de ser mais fácil e mais eficiente calcular inicialmente o valor destas duas variáveis e associá-las à respetiva pessoa, sendo que quando for necessário verificar se a pessoa fica curada ou não basta extrair os valores da mesma

2.2 – Local

A estrutura local é já definida no enunciado com as seguintes propriedades:

- id – Inteiro que identifica o local
- capacidade – Valor inteiro que indica a capacidade máxima do Local
- liga[3] – Array de tamanho 3 que contém os id’s dos locais a que este se encontra ligado. Caso alguma destas ligações não se ligue a outro local, o valor dessa ligação é -1.

Esta estrutura representa os diferentes locais e espaços onde as pessoas são inseridas. Esta estrutura apresenta este aspeto devido às restrições impostas pelo enunciado.

2.3 – PessoaLocal

Esta estrutura é composta por apenas dois elementos:

- pessoa – Ponteiro do tipo pPessoa, que aponta para uma pessoa na lista de pessoas
- next – Ponteiro do tipo pPessoaLocal, que aponta para a próxima estrutura deste tipo

Esta estrutura serve o propósito de criar listas ligadas com pessoas que pertençam ao mesmo local, sendo que quando é usada todas as pessoas associadas encontram-se no mesmo local. Apliquei este método devido a ser uma maneira fácil de associar pessoas do mesmo local.

2.4 – TimeMachine

Esta estrutura é o fundamento da função **Anular Iterações**, é composta por 3 ponteiros do mesmo tipo: pPessoa iteracaoX, sendo que X varia entre 1 e 3. Cada um destes ponteiros aponta para um backup da lista ligada de pessoas original, de modo a ser usada na função antes referida

Capítulo 3 – Principais Estruturas Dinâmicas

3.1 – main

3.1.1 – pLocal espaco

Esta estrutura dinâmica do tipo **Local**, armazenada na forma de um array dinâmico. Foi usado esta forma por diversos motivos: o primeiro ser um requisito do enunciado esta estrutura ser desta forma; Também de notar que o formato da estrutura não permite o uso de listas ligadas. Tendo em conta que esta estrutura também não é alterada ao longo do programa, o mais simples é criar um array com os diferentes locais, criando assim o espaço da simulação.

Realçar que esta estrutura tem que ser dinâmica pois o ficheiro binário que contém a informação relativa ao espaço da simulação nunca indica com quantos locais estamos a trabalhar, pelo que temos que adaptar a memória alocada à quantidade de locais que vamos encontrando ao analisar este ficheiro.

3.1.2 – pPessoa listaPessoas

Esta estrutura dinâmica é do tipo **Pessoa**, sendo que a informação está armazenada numa lista ligada, daí a presença do ponteiro pPessoa nextPessoa na estrutura de dados usada. Ao inicializar o programa é criada esta lista, sendo que ela contém a informação de todas as pessoas lidas no ficheiro fornecido pelo utilizador. Também é, neste preciso momento, atribuído um local a cada pessoa. De notar que pessoas que não podem ser inseridas na simulação (caso a informação esteja inválida ou não existe espaço para inserir as outras pessoas) não estarão presentes nesta lista e serão simplesmente ignoradas.

Esta tipo de estrutura dinâmica é bastante adequado ao contexto da situação. Temos funções que acrescentam pessoas à simulação, pelo que basta percorrer a lista e acrescentar um nó no final, sendo esse nó a

nova pessoa, evitando deste modo ter que realocar memória para a estrutura toda e apenas é necessário alocar memória para a nossa nova pessoa. Também existem outras estruturas onde é mais fácil ter um ponteiro a apontar para o mesmo endereço de memória de um dos nós da lista do que copiar a pessoa em si.

3.1.3 – pTimeMachine timemachine

Esta estrutura é do tipo TimeMachine, sendo que esta estrutura *per se* não é dinâmica, mas as suas componentes sim. Esta estrutura é apenas criada uma vez no código do programa de modo a ter instâncias das suas componentes: timemachine.iteracao1 , timemachine.iteracao2 e timemachine.iteracao3. Estas componentes são listas ligadas do mesmo tipo da listaPessoas, sendo que elas vão ser uma cópia exata da listaPessoas em determinados momentos do programa.

3.1.4 – pPessoa listaOGPessoas

Esta estrutura dinâmica é do mesmo tipo que pPessoa listaPessoas, sendo que apenas serve para fazer uma cópia da lista de pessoas original carregada a partir do ficheiro para se poder apresentar no relatório final várias informações, sendo uma delas a lista de pessoas em concreto.

3.2 – Simulacao

3.2.1 – pPessoaLocal listaPessoasLocal

Esta estrutura dinâmica é uma componente fundamental para a função Avançar Iteração, sendo que é uma lista ligada em que os seus nós são sempre pessoas que estão no mesmo local. É uma lista que é recriada para cada local a cada iteração do programa (de modo a evitar perdas de informação como novas pessoas no local ou pessoas transferidas). Deste modo podemos ao chamar a função “infeta²” sem nos preocupar se de facto todas as pessoas pertencem ao mesmo local.

² Simulacao.c -> avancalteracao -> infeta

Capítulo 4 – Justificação para as opções tomadas em termos de implementação

A primeira grande decisão que implementei neste trabalho prático foi a estrutura de dados PessoaLocal. Tendo em conta que as pessoas já estavam guardadas numa lista ligada devido às restrições do enunciado, assumi que faria mais sentido qualquer secção do programa que tivesse que interagir com esta lista trabalhasse em lista ligada igualmente, e foi o que fiz. A lista que é transferida para as funções de infeta² e recuperacao³ é sempre uma lista deste tipo, deste modo nunca foi preciso verificar ID's, simplesmente tenho que fazer iterativamente o ato de infetar pessoas e o ato de verificar se a pessoa fica curada ou não para cada local individualmente.

Como referido ao apresentar a estrutura de dados Pessoa, decidi também calcular logo ao carregar a lista de pessoas a probabilidade de recuperar naturalmente e a duração máxima da infeção de cada pessoa, permitindo que a função recuperacao³ seja muito mais simples e basta chamar a função probEvento presente no utils.c com o parâmetro da pessoa sem ser necessário mais nenhum cálculo.

Também assumi que as pessoas ainda podem infetar no seu último dia de infeção. Em outras palavras, primeiro é processada a fase de infetar pessoas e só depois é que é verificado se fica curado, sendo que o a secção do código que acrescenta um dia aos dias_infetado só acontece depois de realizar o processo de infetar e recuperar a todos os locais (ou seja mesmo no final da função avancaralteracao).

Criei um array chamado capacidadeLocais[nLocais] para ter sempre presente os lugares livres em cada local, evitando alterar o array dinâmico original, sendo que qualquer função que altere a quantidade de pessoas irá atualizar a capacidade neste array e nunca tem que alterar a composição do array dinâmico espaço. Este array também só é criado após a função startupLocais⁴ pois só assim é que temos o valor de nLocais e a informação para preencher este array.

Quis que o ficheiro report.txt fosse o mais completo possível, mas sem ter informação desnecessária, pelo que inseri informação que achei relevante para se poder tirar qualquer tipo de conclusões sobre um possível estudo sobre a simulação, daí a criação da lista ligada listaOGPessoas e da variável nPessoasInicial, que me permitiu obter dados iniciais da simulação mesmo após a lista listaPessoas e todas as instâncias da timemachine terem perdido essa informação inicial.

Para a timemachine, inicialmente criei funções para ela própria fazer uma copia das listas, e para verificar o valor da variável times para saber em que local (timemachine.iteracaoX $1 \leq X \leq 3$) teria que fazer dita cópia. Depois de criar as funções para realizar o report.txt, e após alguns bugs em termos de memória na timemachine, decidi refazer esta última funcionalidade, o que me permitiu torná-la muito mais simples e eficiente, sendo que agora esta funcionalidade se resume a apenas duas funções e uma estrutura de dados. A primeira função simplesmente tem que chamar a função copiaLista presente em Pessoas.c e evita-se problemas com alocação de memória, tornando todo o processo mais simples. A verificação da variável times é realizada diretamente nos switch cases dentro do main, para evitar confusão ao procurar onde é que cada função está e qual é ativada: em todos os casos é chamada a função presente em timemachine.c e as particularidades são resolvidas dentro do próprio switch case (limpar a lista de pessoas anterior, copiar a lista para a listaPessoas, limpar os snapshots da timemachine necessários, etc.).

De realçar que as limpezas de memória das listas ligadas de pessoas (seja da timemachine seja as outras listas) são limpas iterativamente, para garantir o uso do free() a todos os nós da lista. Para os arrays dinâmicos basta usar diretamente o free(). Também existe uma variante desta função para as listas PessoaLocal.

A capacidade dos locais também é atualizada cada vez que se chama uma das iterações da timemachine.

³ Simulacao.c -> avancalteracao -> **recuperacao**

⁴ Pessoas.c -> **startupPessoas**