Departamento de Matemática da Universidade de Coimbra		
2016/2017	**Programação Orientadas para os Objectos**	**Projecto 2**

Simulador de um Sistema de Filas de Espera

Descrição Sumária: Pretende-se construir um simulador de um sistema de semáforos para um cruzamento de duas estradas (ver Figura 1). Pretende-se saber os tempos de «abertura» (luz verde) para cada uma das vias de acordo com o tráfego esperado para cada uma das vias (ambos os sentidos), evitando longas filas de carros.

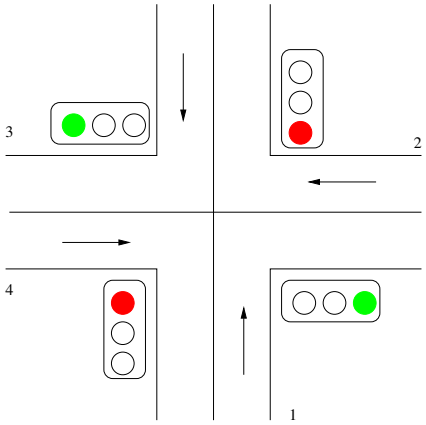

Figura 1: Cruzamento com Semáforos

Os objectivo do simulador é determinar o tempo de abertura para cada um dos pares de semáforos.

Descrição da Simulação: a modelação de sistemas de filas de espera é de grande utilidade na tomada de decisões sobre o dimensionamento de diversos serviços. No caso dos semáforos em cruzamentos de estradas, o tempo de abertura de cada um deles. A modelação destes sistemas tem como objectivo o melhorar do funcionamento de um dado serviço, encontrando soluções equilibradas entre dois cenários possíveis: situações de congestionamento e de desaproveitamento de recursos.

No caso das filas de espera de carros, a chegada a um cruzamento segue uma distribuição de *Poisson* de intensidade λ, ou seja o tempo entre chegadas sucessivas segue uma distribuição exponencial negativa de média $\frac{1}{\lambda}$, e o tempo de «saída», isto é, o tempo que uma dada viatura demora a arrancar e passar pelo cruzamento é exponencial negativa de média μ.

Considera-se que o sistema começa vazio (sem carros). Pretende-se saber qual o número mínimo de tempo de abertura para cada uma das vias de forma a que não haja congestionamentos (filas de espera com mais de 10 carros).

Para além do registo da actividade nas filas de espera para cada semáforo, o simulador deverá contabilizar o tempo médio de espera. Esta média deve incluir apenas os carros que tiveram de esperar nos semáforos.

Em conclusão. O simulador deve dar resposta às seguintes questões:

- o número mínimo de tempo de abertura dos semáforos, sem que haja congestionamentos.
- o tempo mínimo, máximo e médio de espera nos semáforos.

Configuração do Simulador: a simulação deverá permitir definir, de forma dinâmica, todos os parâmetros nela envolvidos. A ideia é possibilitar o teste de uma grande variedade de combinações diferentes. Os parâmetros da simulação deverão ser pedidos ao utilizador através de uma interface

própria para o efeito. Deverá existir a possibilidade de gravar os parâmetros em ficheiro, para mais tarde serem recuperados em simulações futuras. A interface com o utilizador não necessita de ser complexa, devendo contudo ser suficientemente flexível de forma a permitir uma correcta parametrização de todas as variáveis definidas no cenário de simulação. Os parâmetros principais a considerar no contexto da simulação são os seguintes:

- o tempo de abertura em cada uma das vias, no início da simulação;
- o valor da taxa de chegada de carros λ em cada uma das entradas (ambos os sentidos, para cada uma das vias);
- a média do tempo de saída μ em cada via/sentido;

a simulação deverá correr durante um total de 24 horas, ou seja $24 \times 60 \times 60 = 86400$ segundos.

O simulador deverá correr com uma granularidade de 1 segundo, ou seja, todos os tempos são registados em função do número de segundos a partir do início da simulação.

Detalhes de Implementação: uma primeira aproximação para a implementação do simulador seria a utilização de um ciclo de controlo, onde cada iteração corresponderia à unidade temporal mínima pretendida na simulação (no nosso caso 1 segundo). Para além de ser uma solução simples de implementar esta seria uma aproximação ideal para simulações com muita actividade em cada ciclo (ocorrência de vários eventos em cada ciclo).

Contudo, considerando o tipo de actividade envolvida, este não é o caso da nossa simulação. É previsível que durante os 86400 segundo de duração da simulação, em grande parte deles não ocorre qualquer actividade. Numa simulação que dure s segundos e inclua e eventos, será desejável que o tempo de execução da simulação seja determinado pela densidade da actividade e não pelo tempo de duração da simulação. Uma vez que, no caso da simulação pretendida, e é muito menor do que s, será preferível utilizar um mecanismo que não envolva uma iteração por cada segundo. Desta forma será possível acelerar os resultados da simulação.

Uma solução muito mais adequada para este problema consiste na utilização de um esquema baseado em eventos. Em vez de iterarmos sobre cada segundo da simulação poderemos iterar sobre a sequência de eventos que presumivelmente irá ocorrer. Isto irá permitir-nos saltar sobre todos os espaços temporais da simulação onde não ocorre nenhuma actividade com interesse. A seguir descrevemos uma possível abordagem para o algoritmo a utilizar na simulação:

```
inicializa o contador de tempo a 0;
gera a lista de eventos para a duração da simulação;
WHILE (a lista de eventos não estiver vazia) {
  obtém o próximo evento "e" a partir da lista de eventos;
  coloca o contador de tempo com o valor do tempo do evento "e";
  executa "e";
}
```

As estruturas de dados auxiliares para a implementação do sistema de simulação são as filas (referentes às filas de espera dos carros nas diferentes vias) e as listas (referente à lista dos eventos);

Vejamos um exemplo, baseado no conjunto de parâmetros exemplificados anteriormente:

- No início da simulação, programamos a chegada do primeiro carro a uma das vias. Para tal, começamos por escolher um número aleatório gerado de acordo com uma distribuição normal, para escolher em qual das vias de acesso ao cruzamento o carro está a chegar, suponhamos que o valor obtido era 3. De seguida escolhemos um número aleatório gerado de acordo com uma distribuição exponencial negativa, com média $\frac{1}{\lambda}$. Suponhamos que o valor obtido era 17s. Assim programávamos a chegada do primeiro carro para o instante $t = 17$, na fila de espera 3. De igual forma vamos gerar todos os eventos de entrada até se atingir a hora final. Por exemplo, para a

chegada do próximo carro. O processo é idêntico ao descrito anteriormente. Suponhamos que agora os valores escolhidos aleatoriamente eram o 2 e o 21. Neste caso seria necessário inserir um novo evento relativo à chegada do próximo cliente no instante $t = 17 + 21 = 38$s. Note-se que as mudanças de estado dos semáforos são também considerados eventos, os quais ocorrem de forma regular de acordo com os tempos estipulados ao início da simulação.

- De seguida entramos no ciclo de resposta aos eventos e processamos o primeiro evento da lista. O primeiro evento existente da lista é a chegada do primeiro carro no instante $t = 17$s, à via/sentido 3. Como tal, actualizamos o instante actual com o valor 17 e executamos o evento. O carro irá para a via de acesso que lhe está associada, ficando à espera, no semáforo, conforme este esteja «fechado» (luz vermelha) ou com carros ainda à espera, ou passando de imediato o semáforo se este estiver aberto e sem carros à espera para arrancar.

- O próximo evento a ocorrer é a chegada de um novo carro no instante $t = 38$. E o processo repete-se sempre da mesma forma: seleccionando-se o próximo evento a ocorrer (o que tiver um instante de tempo de execução mais baixo).

- Alguns dos eventos não se referem à chegada de mais carros mas sim à mudança de estado dos semáforos. Neste caso, além da mudança de estado do semáforo, podem ser gerados novos eventos. No caso em que o semáforo «abre» (passa a verde), é necessário gerar os eventos de saída das viaturas dessa via/sentido.

A simulação prossegue repetindo o mesmo procedimento até que a lista de eventos fique vazia. A estrutura de dados a ser projectada deve permitir implementar a gestão dos vários eventos criados ao longo da simulação. De notar que a inserção de eventos na lista poderá ocorrer em qualquer ordem, mas a pesquisa pelos eventos deverá ser ordenada pelo tempo programado para a sua ocorrência.

Geração de números aleatórios: como referido anteriormente, quer o tempo entre chegadas de dois carros, quer o tempo que dura a saída dos mesmos de um dado semáforo, seguem uma distribuição exponencial negativa de uma dada média μ. Coloca-se a questão de saber como gerar números aleatórios que sigam esta distribuição. Na realidade, o processo é bastante simples de conseguir à custa de números aleatórios com uma distribuição uniforme. Computacionalmente a geração de números pseudo-aleatórios com uma distribuição uniforme é possível, basta recorrer à função `rand()`. Assim, o algoritmo de geração de números aleatórios com uma distribuição exponencial negativa é o seguinte:

1. gera-se um número aleatório u entre 0 e `RAND_MAX` com uma distribuição uniforme (utilizando o `rand()`);

2. ajustamos esse valor para o intervalo entre 0 e 1;

3. O número aleatório x com uma distribuição exponencial negativa de média μ será então igual a:

$$x = -\ln(1-u)\mu$$

.

Por exemplo, imaginemos que queremos escolher o intervalo de tempo que decorre até aparecer o primeiro carro. Imaginemos que λ é 0,05 (que significa uma taxa de 3 carros por minuto: $\frac{3}{60}$s). Como vimos, neste caso, o tempo entre chegadas sucessivas segue uma distribuição exponencial negativa de média $\frac{1}{\lambda}$, ou seja $\mu = \frac{1}{\lambda} = 20$. Escolhemos um número pseudo-aleatório entre (após o ajustamento) 0 e 1, por exemplo $u = 0{,}38$. Com esse número podemos obter o valor aleatório $x = -\ln(1-0{,}38) - 20 = 9{,}56$ ou seja, arredondando, 10s.

Outros requisitos: Para além das funcionalidades já mencionadas o simulador deverá ser capaz de armazenar um relatório da simulação num ficheiro (nome a ser indicado pelo utilizador e em formato *CSV*). O relatório deverá ter uma linha para cada evento que ocorre, com a indicação do instante temporal em que ele ocorre. Por exemplo:

```
12:25:34 ; "chegada de um novo carro" ; 345 ; "na via" ; 2 ; "aguarda"
 ...
12:32:47 ; "carro" ; 345 ; "abandona a via" ; 2 ; "esperou" ; 120
```

O programa deverá apresentar ao utilizador todos os menus achados necessários para a realização das várias tarefas especificadas neste documento.

Projecto 2, 2016/2017.

- Documente o seu programa. Tanto em termos de documentação interna, como de documentação externa na forma de um pequeno manual de utilização. Utilize os diagramas *UML* para representar a modelização da hierarquia de classes.
- É obrigatório a organização do código em termos de uma hierarquia de classes.
- Deve usar as implementações *STL* para as Filas, assim como para as Listas.
- É obrigatório a apresentação de uma `Makefile` que automatize o processo de compilação.
- Data de realização: 29 de Abril a 2 de Junho.
- Deve entregar (por correio electrónico) um arquivo (`zip`, contendo os ficheiros referentes ao programa (`Makefile, .cpp, .hpp`), assim como o ficheiro referente ao relatório (formato `PDF`), até às 24h00 do último dia do prazo. O nome do ficheiro a entregar deve ser `proj2POO1617GrpN.zip`, com N o número do grupo de trabalho.