Departamento de Matemática da Universidade de Coimbra		
2011/2012	**Programação Orientada para os Objectos**	**Projecto 2**

Simulador de um Sistema de Filas de Espera

Descrição Sumária: Uma empresa proprietária de uma cadeia de hipermercados, pretende construir um simulador de um sistema de filas de espera das caixas num supermercado. A empresa pretende abrir uma nova unidade e está interessada em saber quantas caixas necessita de instalar para que seja sempre possível atender os seus clientes sem que haja longas filas de espera (filas com quatro ou mais clientes). Para isso necessita de saber qual o número mínimo de caixas que serão necessárias de forma a evitar filas longas.

Um dos objectivos do simulador é determinar esse número.

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 hipermercados, número de caixas registadoras abertas num determinado período do dia. 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 das caixas num supermercadode sabe-se que o processo de chegada dos clientes à zona das caixas 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 que o tempo de serviço é exponencial negativo de média μ.

Considera-se que o sistema começa vazio (sem pessoas nas caixas) e com um certo número C_i de caixas em funcionamento. Pretende-se saber qual o número mínimo de caixas necessárias para atender os clientes sem que haja congestionamentos (filas de espera longas).

Podemos considerar que os clientes optam por uma fila de espera em função do número de elementos que esta tem (ou seja, escolhem a que tiver menos pessoas na fila). Uma vez escolhida uma fila, esta não é abandonada pelo cliente.

Considere também que cada nova caixa absorve o cliente que desencadeia a sua abertura. Uma caixa (vazia) fecha sempre que o número médio de caixas vazias nos últimos $t_{\min}$ é superior a C_x.

O horário de funcionamento do hipermercado é das 9 às 23 horas. O simulador deverá estar preparado para assumir três períodos de funcionamento distintos: o período das 9 às 12, o período das 12 às 17 e finalmente o período das 17 às 23. Em cada um destes períodos o valor de λ é diferente.

Como resultado de cada simulação deverá encontrar-se o número mínimo, máximo e médio de caixas que estiveram em funcionamento, em cada período, após a abertura.

Para além do registo da actividade nas filas de espera para cada caixa, o simulador deverá contabilizar o tempo médio de espera. Esta média deve incluir apenas os clientes que não foram imediatamente atendidos.

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

- o número mínimo de caixas que serão necessárias para que não haja filas de espera longas;

- o número mínimo, máximo e médio de caixas que estiveram em funcionamento, em cada período, após a abertura;
- o tempo médio que um cliente espera para ser atendido, por caixa.

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 disco 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:

- número C_i de caixas abertas no início da simulação;
- o valor da taxa de chegada de clientes λ em cada um dos períodos de funcionamento;
- a média do tempo de serviço (atendimento) μ em cada caixa;
- os valores de $t_{\min}$ e C_x (valores por período).

a simulação deverá correr durante um total de 14 horas (das 9 às 23), ou seja 50400 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. Podemos constatar que durante os 50400 segundo de duração da simulação, em grande parte deles não ocorre qualquer actividade com interesse. Numa simulação que dure s segundos e inclua e eventos, seré 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 das caixas do hipermercado) 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 cliente. Para tal, 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 cliente para o instante $t = 17$. De igual forma vamos gerar todos os eventos de entrada até se atingir a hora de fecho. Por exemplo, para a chegada do próximo cliente. O processo é idêntico ao descrito anteriormente. Suponhamos que agora o valor escolhido aleatoriamente era o 21. Neste caso seria necessário inserir um novo evento relativo à chegada do próximo cliente no instante $t = 17 + 21 = 38$s.

- 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 cliente no instante $t = 17$s. Como tal, actualizamos o instante actual com o valor 17 e executamos o evento. Como existem caixas disponíveis, o cliente passa automaticamente para o atendimento. Para determinar o tempo de duração do atendimento, escolhemos um número aleatório seguindo uma distribuição exponencial negativa com média μ, por exemplo 75. Desta forma criamos um novo evento (cliente despachado) no instante de tempo $t = 17 + 75 = 92$s.

- O próximo evento a ocorrer é a chegada de um novo cliente 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).

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.

Outro aspecto que também deverá ter notado no exemplo acima é a necessidade de definir vários tipos diferentes de eventos. Para cada um desses tipos é necessário determinar qual a informação a armazenar e quais as acções a realizar quando o evento é executado. A título de exemplo apresentamos de seguida o evento «chegada de cliente» e as acções a realizar na execução desse evento:

- O evento associado à chegada de um novo cliente, como qualquer evento, deverá estar programado para ocorrer num determinado instante de tempo.

- Na execução do evento relacionado com a chegada de um novo cliente deveremos:

  - em primeiro lugar verificar se existe uma caixa disponível. Se tal for o caso o cliente é atendido imediatamente. Gera-se um novo evento: atendimento terminado;

- se não existir nenhuma caixa disponível, selecciona a caixa com menor fila e espera pela sua vez na fila de espera. Se o número de clientes à sua frente for igual ou superior a três, então caso o número de caixas disponíveis ainda não tenha sido atingido, abre-se mais uma caixa e este cliente passa a ser atendido nessa caixa;
- os eventos do tipo atendimento terminado podem, por sua vez, gerar novos eventos de atendimento terminado, basta para tal que haja outros clientes nessa fila de espera.

O programador terá que decidir que outros tipos de eventos considerar, quais os seus parâmetros e qual o procedimento adequado à sua execução.

Geração de números aleatórios: Como referido anteriormente, quer o tempo entre chegadas de dois clientes, quer o tempo que dura o atendimento em cada caixa seguem uma distribuição exponencial negativa de 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 cliente. Imaginemos que λ é 0,05 (que significa uma taxa de 3 clientes 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 $x = -\ln(1-0{,}38) \times 20 = 9{,}56$ 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 cliente" ; 345 ; "escolheu a caixa" ; 2 ; "aguarda"
 ...
12:32:47 ; "cliente" ; 345 ; "abandona a caixa" ; 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, 2011/2012.

- 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 organizar o seu código de forma a que os ficheiros referentes ao T.A.D Filas (classe genérica) e ao T.A.D. Lista (classe genérica) e o referente ao programa de simulação estejam em ficheiros separados.
- A especificação algébrica dos tipos abstractos de dados filas e listas é:

  Filas=({vazia,Elemento:Fila},{cria, insere, retira, topo, comprimento, vazia})

  Listas=({vazia,Elemento:Lista},{cria, insere(`int`,Elemento), retira(`int`), ve(`int`), vazia});

  respectivamente.

  Nota a operação `insere` refere-se à inserção de um dado elemento na posição dada («empurrando» os outros elementos para a direita). As operações `retira` e `ve` referem-se às posições dadas.

  O programa principal deve ser independente da implementação realizada de tal forma que as diferentes implementações das classes respectivas, pelos diferentes grupos, possam ser intermutáveis.

  A classificação dos trabalhos vai reflectir isso.

  Os métodos devem ter os nomes referidos acima e os ficheiros deverão ter, obrigatoriamente, os seguintes nomes: `filasGenericas.cpp` e `listasGenericas.cpp`.
- É obrigatório a apresentação de uma `Makefile` que automatize o processo de compilação.
- Data de realização: 27 de Abril a 11 de Junho.
- Deve entregar (por correio electrónico) um arquivo (`zip, tgz, rar`), 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 `proj2POO1112GrpN.{zip|tgz|rar}`, com N o número do grupo de trabalho.