Simulação de uma fila:

A simulação por computador consiste na execução do modelo de um sistema real (ou ate mesmo teórico) seguida de uma analise dos resultados obtidos [FISHWICK1996]. Em outras palavras trata-se da execução de um programa que imita o comportamento de um sistema. Na medida em que as filas surgem com frequência nas simulações por computador, especialmente quando se trata de sistemas que envolvem a "prestação" de um serviço, aqui serão discutidos alguns aspectos da implementação de simulações por computador.

O "comportamento" real de um sistema normalmente envolve atividades realizadas em paralelo (enquanto alguém entra em uma fila, outra pessoa está saindo do banco, outra está saindo da fila, um atendimento está sendo realizado, etc...). É portanto natural que se espere a extensa utilização de programação paralela (eventualmente com recursos de computação distribuída, grids, etc) na execução de simulações. Essa, no entanto, não será a abordagem aqui utilizada.

Novamente, em busca da simplificação, será abordado o modelo de execução serial, que é mais simples na sua implementação e não exige recursos especiais. Na simulação serial as ações de simulação ocorrem em seqüência e para criar a ilusão de paralelismo é realizado um cadenciamento dissociado do tempo real. As ações independentes são realizadas em sequência sob a cadência de um relógio global de simulação. Diferentemente de uma simulação em tempo real, esse relógio só é incrementado quando forem finalizadas todas as ações previstas para um ciclo de simulação.

Essa estratégia é implementada na simulação de uma fila para atendimento nos caixas de um banco, conforme Pereira [PEREIRA2003], o qual propõe esta simulação e implementa a solução em Pascal. A ideia aqui consistirá na utilização de uma fila dinâmica cuja operação de inserção será por prioridade (baseada no tipo de serviço ou na idade, por exemplo).

Simulando a Fila dos Caixas Eletrônicos [PEREIRA2003]

Pretendemos simular uma situação que nos permita determinar qual o tempo médio que um cliente aguarda numa fila, para realizar uma transação no caixa eletrônico de uma agência bancária.

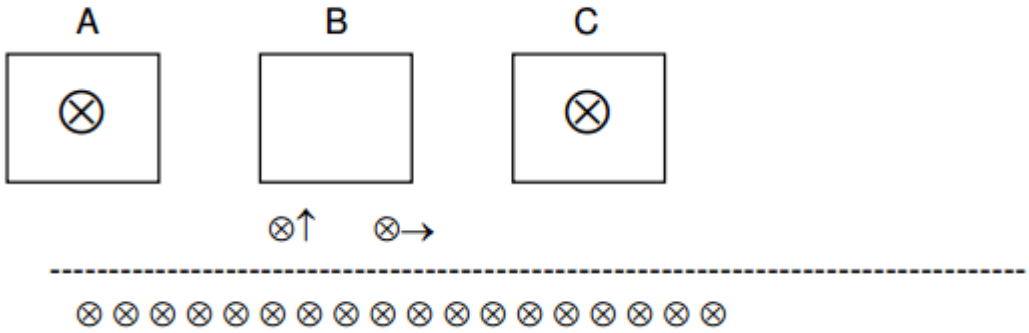

Figura 1: Disposição dos caixas.

O caixa eletrônico oferece cinco opções ao cliente e, através de estatísticas, chegamos ao tempo médio necessário para realizar cada uma das transações possíveis:

Transação	Código	Tempo
Saldo	0	10 s
Saque	1	20 s
Aplicação	2	30 s
Extrato Semanal	3	40 s
Extrato Mensal	4	50 s

Quando um cliente entra na fila, o horário é anotado. Quando ele sai, verifica-se quanto tempo ele aguardou e este valor é acumulado. Também o total de clientes é registrado, de modo que no final do expediente, seja possível determinar o tempo médio que um cliente permanece na fila para utilizar o caixa.

Duas entidades concretas estão envolvidas neste problema: caixas e clientes. Vamos abstrair

destas entidades apenas os atributos essenciais para a resolução do problema, objetivando uma implementação simples e funcional.

Tudo que precisamos saber sobre um caixa é se ele está ocupado ou não. Caso esteja, precisamos Ter uma ”noção” de por quanto tempo ele será usado pelo cliente. Assim, cada caixa será representado por uma variável inteira, cujo valor indica por quantos minutos ele permanecerá ocupado. Se o valor desta variável for 0, então o caixa está livre. Inicialmente, todos os caixas estão livres. Quando um cliente inicia uma transação num caixa, o tempo necessário para realizar a transação escolhida por ele é que determinará quanto tempo o caixa permanecerá ocupado.

A outra entidade que precisamos representar é o cliente. Dele só nos interessa o momento em que ele entrou na fila para que, ao sair, possamos calcular quanto tempo ele aguardou. Portanto, cada cliente será representado por um número inteiro correspondente ao horário em que ele entrou na fila.

Na simulação ora apresentada baseada em Pereira [PEREIRA2003], dois eventos que podem ocorrer são de interesse particular:

- œUm cliente chega à agência e entra na fila;
- œUm caixa é liberado, alguém sai da fila e o utiliza.

Para sincronizar estes eventos, vamos assumir que temos um cronômetro que marca unidades de tempo. Durante uma unidade de tempo, que poderia ser um segundo, qualquer combinação dos eventos pode ocorrer, até mesmo nenhuma.

Para simular a dinâmica com que os eventos ocorrem, procedemos como segue:

- œZerar o cronômetro;
- œEnquanto não terminar o expediente;
  a) se um cliente chegou, então entra na fila;
  b) se fila não está vazia e um caixa está livre, então cliente inicia transação;
  c) para cada caixa ocupado, decrementar o seu tempo de uso;
  d) incrementar o cronômetro.

Alguns pontos do algoritmo acima devem ser detalhados para que possamos programá-los:

Terminou o expediente?

Para determinar o final do expediente, usamos o próprio cronômetro. Para isto, basta definir o período de atendimento da agência em termos das unidades de tempo marcadas pelo cronômetro. Quando o cronometro atingir o valor definido, então terminou o expediente.

Chegou um cliente?

Numa situação real, não teremos um cliente chegando à agência a cada minuto para usar o caixa eletrônico. Para implementar esta parte do algoritmo, de modo que tenhamos algo não sistemático, podemos utilizar uma função que a cada chamada retorne um valor aleatório.

```
function ClienteChegou : boolean;
begin
        ClienteChegou := (random(3) = 0);
end;
```

Note que a chamada random(n) retorna um valor aleatório entre 0 e n-1, sendo que qualquer valor tem a mesma probabilidade de ser sorteado. Desta forma, a função ClienteChegou(...) responderá de forma afirmativa, em média, uma vez em cada três que for chamada.

Cliente entra na fila.

Convencionamos representar um cliente apenas pelo horário em que ele entrou na fila. O tempo está sendo controlado pelo cronometro. Logo, colocar um cliente na fila equivale a simplesmente armazenar nela o valor corrente do cronometro, no momento que ele chega.

Inicia transação.

Quando um caixa é liberado e um cliente começa a utilizá-lo, precisamos saber por quanto tempo ele estará ocupado. O tempo de uso do caixa depende da transação que é iniciada pelo cliente. Aqui também é interessante que as transações iniciadas sejam aleatórias:

```
function Transacao : integer;
begin
        case random(5) of
                0 : Transacao := 10;
                1 : Transacao := 20;
                2 : Transacao := 30;
                3 : Transacao := 40;
                4 : Transacao := 50;
        end;
end;
```

Cada vez que a função Transacao(...) é chamada, ela "sorteia" uma transação e retorna o tempo médio necessário para realizá-la. Uma vez que um caixa é representado por uma variável que especifica quanto tempo ele estará ocupado, para indicar que um cliente está usando um caixa, basta selecionar uma transação e armazenar o tempo correspondente na variável que representa o caixa.

Veja a seguir, a codificação completa para o simulador de filas de caixa:

```
program FilaCaixa;
uses  Filas;
const  FINAL = 100;
var  crono, espera, totcli, client : integer;
caixa : array[1..3] of integer;
filcx : Fila;
{ insira aqui as rotinas ClienteChegou() e Transacao() }
begin
        crono  := 0;
        espera := 0;
        totcli := 0;
        Qinit(filcx);
        for i :=1 to 3 do
                caixa[i] := 0;
        while crono<FINAL do
        begin
                if ClienteChegou then
                begin
                        Enqueue (filcx, crono);
                        inc(totcli);
                end;
                for i:=1 to 3 do
                        if (not QisEmpty(filcx)) and (caixa[i]=0) then
                        begin
                                client := Dequeue (filcx);
                                inc(espera, (crono-cient));
                                caixa[i] := Transacao;
                        end;
                for i:=1 to 3 do
                        if caixa[i]<>0 then dec(caixa[i]);
                        inc(crono);
        end;
        writeln;
        writeln('Tempo de atendimento: ', FINAL);
        writeln('Total de clientes ..: ', totcli);
        writeln('Tempo tot. de espera: ', espera);
        writeln('Tempo med. De espera: ', (espera/totcli):0:2);
end.
```

Comentários Finais

Geração de números pseudo-aleatórios em C

#include <time.h>

Para inicializar a semente de geração de números aleatórios, antes de chamar qualquer função, insira no corpo da função main( ), uma chamada à função srand((unsigned)time(NULL));

Chegou cliente

Numa situação real, não teremos um cliente chegando no sistema a cada minuto ou a cada intervalo de tempo conhecido. Para implementar esta parte do algoritmo, de modo que tenhamos algo não sistemático, podemos utilizar uma função que a cada chamada retorne um valor aleatório. Dependendo do valor retornado consideraremos que um novo cliente chegou ou não.

```
int chegou(void)
{ if (rand( )%3 ==0)
        return SUCESSO;
  else
        return FRACASSO;
}
```

Note que a chamada rand( )%N retorna um valor entre 0 e N-1, da forma como a função chegou( ) utiliza tal valor a probabilidade de SUCESSO (chegada de cliente) é 1/3. Desta forma, a função ClienteChegou() responderá de forma afirmativa, em média, uma vez em cada três que for chamada.

Tempo de transação

Cada cliente trás serviços a serem executados pelo caixa, tais serviços demandarão um tempo o qual poderá ser calculado pela função abaixo.

```
int tempoTransação(void)
{
        int x,t;
        t=(rand( )%5); /* geração de número aleatório 't' irá assumir um valor entre 0 e 5
                        (inclusive) tais valores serão equiprováveis */
        switch (t) {
        case 0:  /* transação tipo zero */
                x = 10;
```

```
                break;
        case 1: /* transação tipo 1 */
                x = 20;
                break;
        .....
        }
        return x;
}
```

No corpo da simulação. Recomenda-se a declaração da variável “espera” como inteiro longo “long int”.

Referências Bibliograficas

[FISHWICK1996] Fishwick, P. A. Computer Simulation Potentials, IEEE , Volume: 15 , Issue: 1 , P:24–27.Feb.-March.1996

[PEREIRA2003] Pereira, Sílvio L. Estruturas de Dados Fundamentais – Conceitos e Aplicações. 7a. ed. Érica, 2003, p. 73