CENTRO DE CIÊNCIA E TECNOLOGIA
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS
UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

Aplicações de Filas

Disciplina: Estrutura de Dados I

Prof. Fermín Alfredo Tang Montané

Curso: Ciência da Computação

Aplicações de Filas

Descrição

- Apresentamos duas aplicações de filas:
  - Classificar dados em categorias;
  - Simulação de filas;
- A primeira aplicação é muito utilizada como uma pre-ordenação, inclusive dentro de outras aplicações. A segunda aplicação é utilizada para estudar o desempenho de qualquer aplicação de fila.

Aplicações de Filas

Classificar dados em Categorias

- Em muitos aplicações é necessário rearranjar a ordem dos dados sem destruir a sequência básica.
- Como exemplo considere a seguinte sequência de números:

```
3 22 12 6 10 34 65 29 9 30 81 4 5 19 20 57 44 99
```

- Desejamos agrupar os números em categorias preservando a ordem original deles dentro de cada grupo. Trata-se de uma aplicação em que são necessárias várias filas.
- Deseja-se classificar os números em quatro categorias:
  - Grupo1: menores que 10;
  - Grupo2: entre 10 e 19;
  - Grupo 3: entre 20 e 29;
  - Grupo 4: maiores ou iguais que 30.

Aplicações de Filas

Classificar dados em Categorias

- Como exemplo considere a seguinte sequência de números:

```
3 22 12 6 10 34 65 29 9 30 81 4 5 19 20 57 44 99
```

- Como resultado do rearranjo teremos a seguinte sequência de números:

```
| 3 6 9 4 5 | 12 10 19 | 22 29 20| 34 65 30 81 57 44 99 |
```

- Não se trata de um algoritmo de ordenação. A sequência resultante é uma lista de números não ordenada, porém categorizada em grupos de acordo com regras definidas. Os números dentro de cada grupo preservaram a ordem da lista original.

Aplicações de Filas

Classificar dados em Categorias - Algoritmo

- O algoritmo para resolver este problema consiste em construir uma fila para cada uma das quatro categorias. Ler a sequência numérica e armazenar os números na fila apropriada. Após processar todos os números da sequencia, imprimir o conteúdo de cada fila para mostrar que a classificação foi realizada corretamente.

```
Algorithm categorize
   Group a list of numbers into four groups using four queues.
      Written by:
      Date:
1 createQueue (q0to9)
2 createQueue (q10to19)
3 createQueue (q20to29)
4 createQueue (qOver29)
5 fillQueues  (q0to9, q10to19, q20to29, qOver29)
6 printQueues (q0to9, q10to19, q20to29, qOver29)
end categorize
```

- A função fillQueues realiza o preenchimento das filas.
- A função printQueues realiza a impressão do conteúdo das filas.

Aplicações de Filas

Classificar dados em Categorias - Algoritmo

- O algoritmo a função fillQueue() e o seguinte:

```
Algorithm fillQueues (q0to9, q10to19, q20to29, qOver29)
This algorithm reads data from the keyboard and places them in
one of four queues.
  Pre  all four queues have been created
  Post queues filled with data
1 loop (not end of data)
  1  read (number)
  2  if (number < 10)
     1  enqueue (q0to9, number)
  3  elseif (number < 20)
     1  enqueue (q10to19, number)
  4  elseif (number < 30)
     1  enqueue (q20to29, number)
  5  else
     1  enqueue (qOver29, number)
  6  end if
2 end loop
end fillQueues
```

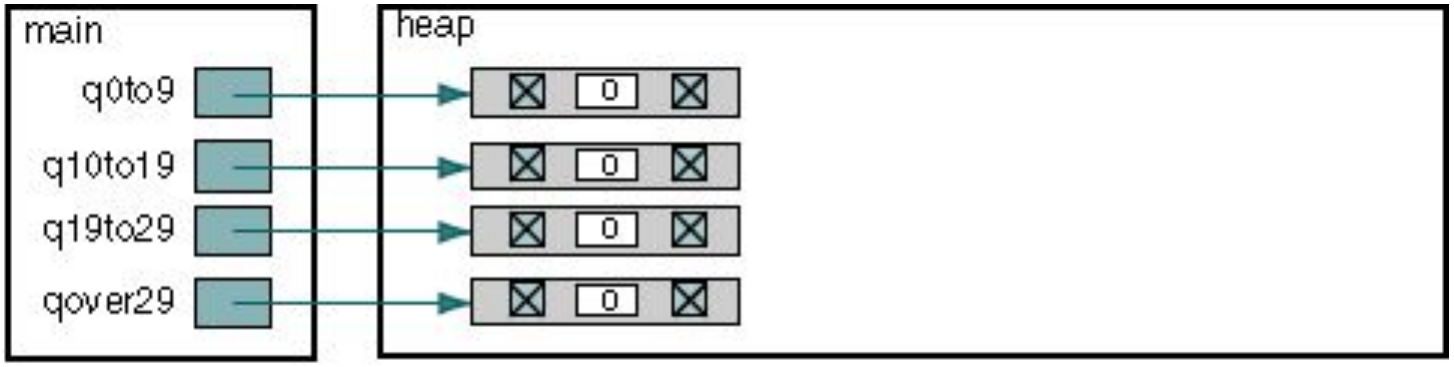
Condição de parada

Leitura dos dados

Aplicações de Filas

Classificar dados em Categorias - Implementação

- Para ilustrar melhor o programa mostra-se as estruturas criadas e seu conteúdo.

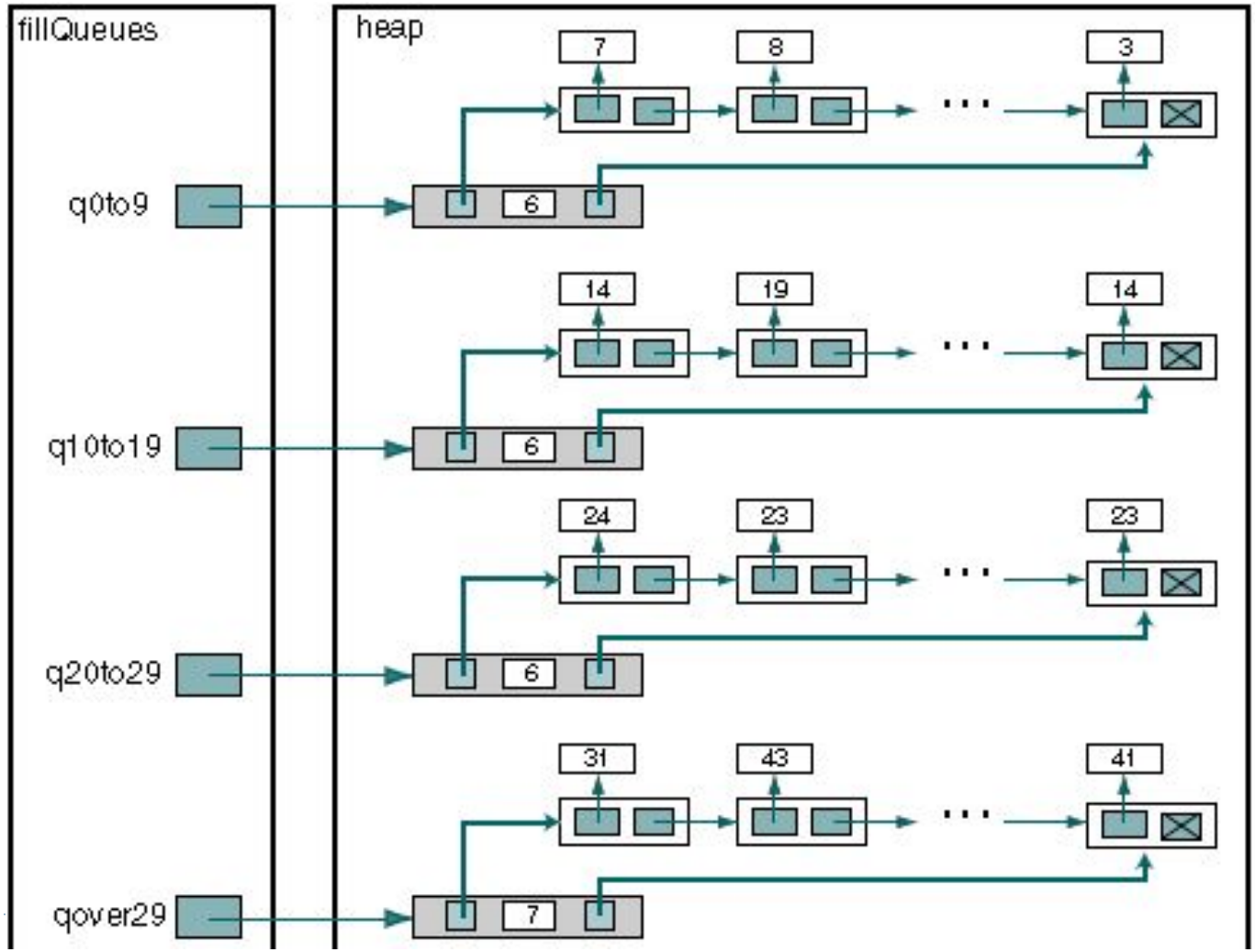


(a) Before calling fillQueues

Aplicações de Filas

Classificar dados em Categorias - Implementação

- Mostra-se as estruturas criadas pelo programa e seu conteúdo.



Aplicações de Filas
Classificar dados em Categorias - Implementação

- Mostra-se as estruturas criadas pelo programa e seu conteúdo.

(b) After calling fillQueues

- O programa começa criando as filas.
- Uma vez que as filas foram criadas, se chama uma função para criar e enfileirar os dados.
- Finalmente se chama uma função para imprimir os dados de cada fila.

Aplicações de Filas

Classificar dados em Categorias - Implementação

- O programa classificador em categorias é mostrado a seguir.

P4-11.c (Classificador em Categorias)

```
1  /*Groups numbers into four groups using four queues.
2       Written by:
3       Date:
4  */
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <stdbool.h>
8  #include "queues.h"
9
10 //Prototype Statements
11 void fillQueues  (QUEUE*, QUEUE*, QUEUE*, QUEUE*);
12 void printQueues (QUEUE*, QUEUE*, QUEUE*, QUEUE*);
13
14 void printOneQueue (QUEUE* pQueue);
15
16 int main (void)
17 {
18 //Local Definitions
19   QUEUE* q0to9;
20   QUEUE* q10to19;
21   QUEUE* q20to29;
22   QUEUE* qOver29;
```

Inclusão da TAD Fila

Preenchimento das Filas

Impressão das Filas

Impressão de uma Fila

Protótipo das funções da aplicação

Declaração das das filas para as 4 categorias

Aplicações de Filas

Classificar dados em Categorias - Implementação

- O programa classificador cria 4 filas, uma para cada categoria (ou intervalo de valores aceito na categoria); depois processa um conjunto de dados e preenche as filas de acordo com os valores; finalmente, imprime o conteúdo de cada fila.

P4-11.c (Classificador em Categorias - Continuação...)

```
23
24  //Statements
25    printf("Welcome to a demonstration of categorizing\n"
26           "data. We generate 25 random numbers and then\n"
27           "group them into categories using queues.\n\n");
28
29    q0to9   = createQueue ();
30    q10to19 = createQueue ();
31    q20to29 = createQueue ();
32    qOver29 = createQueue ();
33
34    fillQueues  (q0to9, q10to19, q20to29, qOver29);
35    printQueues (q0to9, q10to19, q20to29, qOver29);
36
37    return 0;
38  } // main
```

Cria as filas para as 4 categorias

Preenchimento das Filas

Impressão das Filas

Aplicações de Filas

Classificar dados em Categorias - Implementação

- A função **fillQueues()** usa um gerador de números aleatórios para criar números entre 0 e 50. O número gerado é impresso na tela e inserido na fila apropriada de acordo com o seu valor.

P4-12.h (Função fillQueues())

```
1  /*================ fillQueues ================
2    This function generates data using rand() and
3    places them in one of four queues.
4      Pre:  All four queues have been created
5       Post: Queues filled with data
6  */
7  void fillQueues (QUEUE* q0to9,   QUEUE* q10to19,
8                   QUEUE* q20to29, QUEUE* qOver29)
9  {
10 //Local Definitions
11   int  category;
12   int  item;
13   int* dataPtr;
14
15 //Statements
16   printf("Categorizing data:\n");
17   srand(79);
18
```

Recebe as 4 filas como parâmetros

Representa a categoria

Usa dados inteiros

Ponteiro a inteiro

Inicializa a semente de números aleatórios

Aplicações de Filas

Classificar em Categorias

P4-12.h (Função fillQueues() – Continuação...)

```
19    for (int i = 1; i <= 25; i++)
20        {
21        if (!(dataPtr = (int*) malloc (sizeof (int))))
22            printf("Overflow in fillQueues\a\n"),
23                    exit(100);
24
25         *dataPtr = item = rand() % 51;
26         category = item / 10;
27         printf("%3d", item);
28         if (!(i % 11))
29            // Start new line when line full
30            printf("\n");
31
32         switch (category)
33            {
34             case 0 : enqueue (q0to9, dataPtr);
35                      break;
36             case 1 : enqueue (q10to19, dataPtr);
37                      break;
38             case 2 : enqueue (q20to29, dataPtr);
39                      break;
40             default: enqueue (qOver29, dataPtr);
41                      break;
42            } // switch
43        } // for
44    printf("\nEnd of data categorization\n\n");
45    return;
46  } // fillQueues
```

Aloca memória para o novo dado inteiro

Gera um número aleatório entre 1 e 50

Define a categoria a que pertence o número

Insere na fila adequada

Aplicações de Filas

Classificar dados em Categorias - Implementação

- A função **printQueues()** realiza a impressão do conteúdo de cada fila.
- Para isso realiza uma chamada a função auxiliar **printOneQueue()**.

P4-13.h (Função printQueues())

```
1  /*================ printQueues ================
2    This function prints the data in each of the queues.
3       Pre  Queues have been filled
4       Post Data printed and dequeued
5  */
6  void printQueues (QUEUE* q0to9,   QUEUE* q10to19,
7                    QUEUE* q20to29, QUEUE* qOver29)
8  {
9  //Statements
10   printf("Data   0.. 9:");
11   printOneQueue (q0to9);
12
13   printf("Data  10..19:");
14   printOneQueue (q10to19);
15
16   printf("Data  20..29:");
17   printOneQueue (q20to29);
18
19   printf("Data over 29:");
20   printOneQueue (qOver29);
21
22   return;
23 } // printQueues
```

Aplicações de Filas

Classificar dados em Categorias - Implementação

- A função **printOneQueue()** imprime o conteúdo de uma fila. Para isso, realiza um loop enquanto houver dados na fila: Primeiro desenfileira um nó e depois imprime o seu conteúdo. Utiliza-se a função **EmptyQueue()** (Fila Vazia) para verificar se o loop deve continuar.

P4-14.h (Função printOneQueue())

```
1  /*================= printOneQueue =================
2    This function prints the data in one queue,
3    ten entries to a line.
4       Pre  Queue has been filled
5       Post Data deleted and printed. Queue is empty
6  */
7  void printOneQueue (QUEUE* pQueue)
8  {
9  //Local Definitions
10   int  lineCount;
11   int* dataPtr;
12
```

Contador de linha

Ponteiro a inteiro

Recebe 1 fila como parâmetro

Aplicações de Filas

Classificar dados em Categorias - Implementação

- A função **printOneQueue()** imprime o conteúdo de uma fila. Para isso, realiza um loop enquanto houver dados na fila: Primeiro desenfileira um nó e depois imprime o seu conteúdo. Utiliza-se a função **EmptyQueue()** (Fila Vazia) para verificar se o loop deve continuar.

P4-14.h (Função printOneQueue() – Continuação...)

```
13  //Statements
14    lineCount = 0;
15    while (!emptyQueue (pQueue))
16          {
17           dequeue (pQueue, (void*)&dataPtr);
18           if (lineCount++ >= 10)
19              {
20               lineCount = 1;
21               printf ("\n              ");
22              } // if
23           printf("%3d ", *dataPtr);
24          } // while !emptyQueue
25    printf("\n");
26
27    return;
28  } // printOne Queue
```

Repete o loop enquanto a fila não estiver vazia

Extrai o elemento da fila. Repassa o endereço do ponteiro ao inteiro (referencia dupla)

Imprime o valor extraído da fila

Aplicações de Filas
Classificar dados em Categorias - Implementação

- Um exemplo dos resultados do programa classificador é o seguinte:

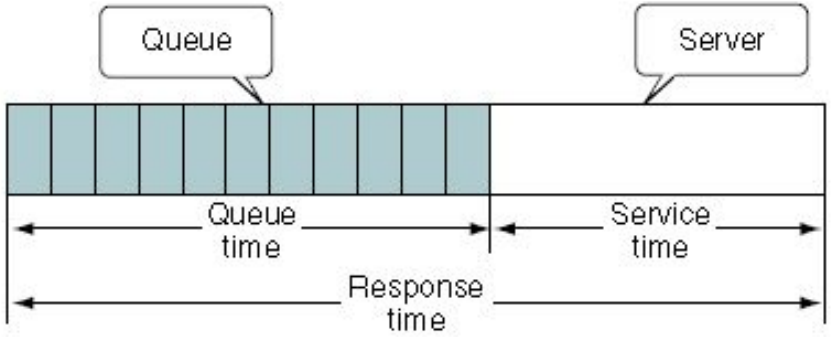
```
Results:
Welcome to a demonstration of categorizing
data. We generate 25 random numbers and then
group them into categories using queues.

Categorizing data:
 24  7 31 23 26 14 19  8  9  6 43
 16 22  0 39 46 22 38 41 23 19 18
 14  3 41
End of data categorization

Data   0.. 9:  7   8   9   6   0   3
Data  10..19: 14  19  16  19  18  14
Data  20..29: 24  23  26  22  22  23
Data over 29: 31  43  39  46  38  41  41
```

- Obs. Este programa utiliza o TAD Fila (arquivos P4-1.h até P4-10.h, além do queues.h). Que podem ser reunidas em um único arquivo.c e seu respectivo.h.
- As funções auxiliares fillQueues(), printQueues() e printOneQueue() nos arquivos P4-12, P4-13 e P4-14, podem ser incorporadas na aplicação.

Aplicações de Filas

Simulação Filas – Modelo de fila única

- Uma outra aplicação importante de filas é a simulação de filas.
- Trata-se da modelagem da atividade de uma ou mais filas usada para gerar estatísticas sobre o funcionamento das filas.



Aplicações de Filas

Simulação Filas – Modelo de fila única

- Vamos construir um modelo de fila única para simular o comportamento de uma loja de doces com um único atendente. O atendente somente pode atender um cliente de cada vez.
- Devido a que os clientes podem fazer pedidos que envolvem vários tipos de doces para serem enviados por encomenda, o tempo de atendimento pode ser demorado, variando de 1 a 10 minutos.
- Queremos estudar a atividade da loja em um dia hipotético. A loja funciona 8 horas por dia. Considerando 60 minutos por hora, o modelo deverá simular 480 minutos.
- A simulação utiliza um **relógio** (variável contador) que controla o tempo. A aplicação monitora a ocorrência de eventos, basicamente o inicio e término de eventos, em intervalos de 1 minuto.
- Cada minuto a simulação precisa verificar três possíveis eventos:
  - Chegada de um cliente;
  - Inicio de atendimento de um cliente;
  - Conclusão do atendimento de um cliente.

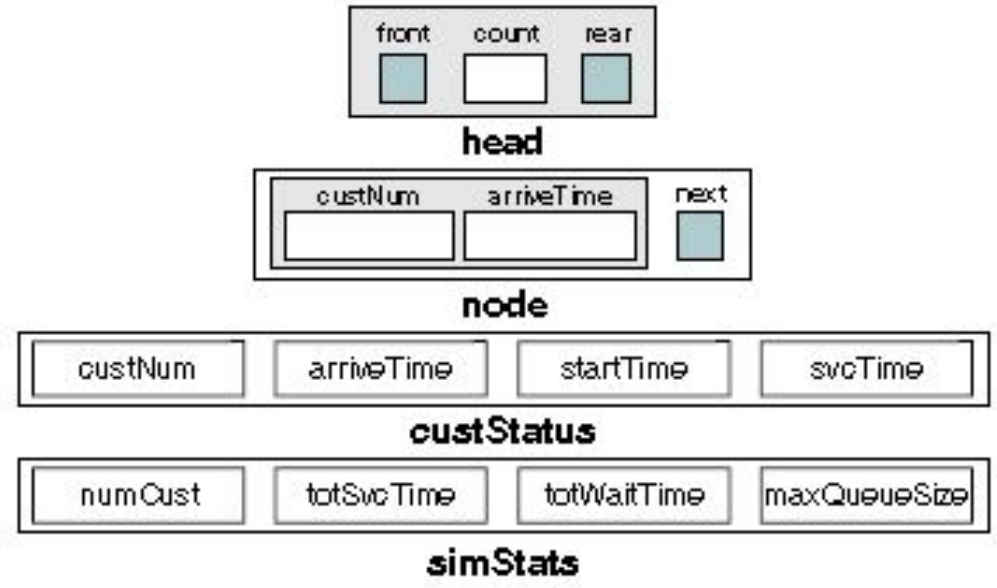
Aplicações de Filas

Simulação Filas – Eventos

- **Chegada de um cliente.-** A chegada de um cliente é verificada por um módulo chamado ***new customer***.
- O proprietário da loja estimou que em média um cliente chega a cada 4 minutos. Simulamos a chegada de um cliente usando um gerador de números aleatórios que retorna um valor entre 1 e 4. Considera-se que uma chegada acontece quando o número é igual a 4.
- **Inicio de atendimento de um cliente.-** O inicio de atendimento de um cliente somente acontece quando o servidor está ocioso. Em cada minuto da simulação o módulo ***server free*** verifica se o atendente está ocupado ou ocioso. Neste último caso, o próximo cliente em espera poderá ser atendido.
- **Conclusão do atendimento de um cliente.-** O módulo ***service complete*** determina se o atendimento do cliente em serviço foi concluído. Para isso, o tempo de atendimento para o cliente atual precisa ser estimado, usando novamente um gerador de números aleatórios. Quando o atendimento é concluído, o aplicativo coleta estatísticas e torna o servidor ocioso novamente.

Aplicações de Filas
Simulação Filas – Estruturas

- Quatro estruturas são necessárias para a aplicação de simulação de filas.
  - **head.-** representa o nó cabeçalho da fila;
  - **node.-** representa o nó cliente que será armazenado na fila. Armazena a identificação do cliente e o seu tempo de chegada;
  - **custStatus.-** armazena dados do cliente atual (identificação, tempo de chegada, tempo de inicio, tempo de atendimento);
  - **simStats.-** armazena as estatísticas da simulação (numero de clientes, tempo total de serviço, tempo total de espera, tamanho máximo da fila).



Cabeçalho

Nó

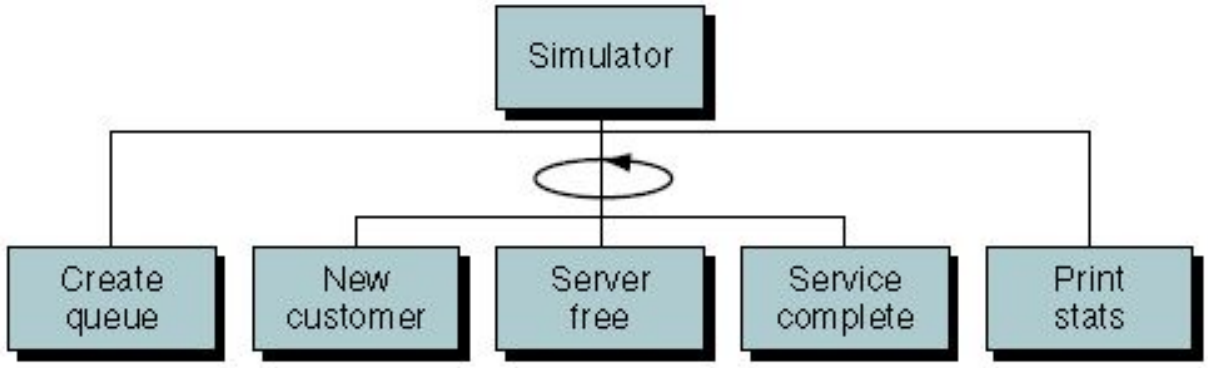
Estado do cliente atual

Estatísticas da simulação

Aplicações de Filas

Simulação Filas – Aplicação

- Módulos da aplicação de simulação:



Aplicações de Filas

Simulação Filas – Aplicação

- O algoritmo principal da aplicação de simulação consiste de um loop que simula a passagem do tempo contabilizado em minutos. O contador clock registra o tempo atual (unidades discretas). O loop termina quando o tempo de simulação (endTime) e não existir mais clientes (moreCusts). Dentro do loop o algoritmo gerencia:
- i) a chegada de novos clientes (newCustomer); ii) a liberação do servidor (serverFree) e inicio de um novo serviço e iii) o fim de serviço (svcComplete)

```
Algorithm taffySimulation
This program simulates a queue for a saltwater taffy store.
     Written by:
     Date:
1 createQueue (queue)
2 loop (clock <= endTime OR moreCusts)
  1  newCustomer  (queue, clock, custNum)
  2  serverFree   (queue, clock, custStatus, moreCusts)
  3  svcComplete  (queue, clock, custStatus,
                   runStats,     moreCusts)
  4  if (queue not empty)
     1  set moreCusts true
  5  end if
  6  increment clock
3 end loop
4 printStats (runStats)
end taffySimulation
```

Cria a fila para armazenar os clientes

Loop principal da simulação

Verifica se existem clientes na fila

Relógio

Imprime estatísticas da simulação

Aplicações de Filas

Simulação Filas – Aplicação

- O algoritmo que gerencia a chegada de novos clientes, **newCustomer**, determina se um novo cliente chegou a loja. A função recebe como parâmetros, a fila de clientes (queue), o relógio (clock) e a identificação do último cliente atendido (custNum).
- Utiliza-se números aleatórios para decidir se um cliente chegou.
- Caso um novo cliente chegue, cria-se um novo nó (custData) com: i) a identificação desse cliente (incrementa-se custNum) e ii) o tempo de chegada (arrivalTime).
- O novo nó é inserido na fila de clientes.

```
Algorithm  newCustomer (queue, clock, custNum)
This algorithm determines if a new customer has arrived.
  Pre    queue is a structure to a valid queue
         clock is the current clock minute
         custNum is the number of the last customer
  Post   if new customer has arrived, placed in queue
1 randomly determine if customer has arrived
2 if (new customer)
  1  increment custNum
  2  store custNum in custData
  3  store arrival time in custData
  4  enqueue (queue, custData)
3 end if
end newCustomer
```

Determina se um novo cliente chegou

Cria a um nó com os dados do novo cliente

Insere o novo nó na fila de clientes

Aplicações de Filas

Simulação Filas – Aplicação

- O algoritmo que gerencia a liberação do servidor, **serverFree**, determina se o servidor está ocioso e nesse caso inicia o serviço de um novo cliente. A função recebe como parâmetros, a fila de clientes (queue), o relógio (clock), os dados do cliente atual (status) e o flag sobre a existência de mais clientes (moreCusts).

```
Algorithm serverFree (queue, clock, status, moreCusts)
This algorithm determines if the server is idle and if so
starts serving a new customer.
  Pre   queue is a structure for a valid queue
        clock is the current clock minute
        status holds data about current/previous customer
  Post  moreCusts is set true if a call is started
1 if (clock > status startTime + status svcTime - 1)
     Server is idle.
   1  if (not emptyQueue (queue))
      1  dequeue (queue, custData)
      2  set status custNum    to custData number
      3  set status arriveTime to custData arriveTime
      4  set status startTime  to clock
      5  set status svcTime    to random service time
      6  set moreCusts true
   2  end if
 2 end if
end serverFree
```

Verifica se existem clientes na fila

Determina se o servidor terminou o atendimento

Remove o próximo cliente da fila

Atualiza status com os dados do novo cliente

Aplicações de Filas

Simulação Filas – Aplicação

- O algoritmo **svcComplete** determina se o cliente atual terminou seu atendimento. A função recebe como parâmetros, a fila de clientes (queue), o relógio (clock), os dados do cliente atual (status), as estatísticas da simulação (stats) e o flag sobre a existência de mais clientes (moreCusts).
- Caso o serviço do cliente tenha concluído, os dados do cliente são impressos (status), as estatísticas da simulação atualizadas (stats) e flag sobre a existência de mais clientes (moreCusts) desativado.

```
Algorithm svcComplete (queue, clock, status,
                       stats, moreCusts)
This algorithm determines if the current customer's
processing is complete.
  Pre    queue is a structure for a valid queue
         clock is the current clock minute
         status holds data about current/previous customer
         stats contains data for complete simulation
  Post   if service complete, data for current customer
         printed and simulation statistics updated
         moreCusts set to false if call completed
```

Aplicações de Filas
Simulação Filas – Aplicação

- Caso o serviço do cliente tenha concluído, os dados do cliente são impressos (status), as estatísticas da simulação atualizadas (stats) e flag sobre a existência de mais clientes (moreCusts) desativado.

Determina se o servidor terminou o atendimento

Calcula o tempo de espera do cliente

Atualiza estatísticas da simulação: número de clientes, tempo total de serviço, tempo total de espera, tamanho máximo da fila.

Atualiza o tamanho máximo da fila

Imprime dados do cliente mais dados de tempo de espera e tamanho da fila

```
1 if (clock equal status startTime + status svcTime - 1)
  Current call complete
  1  set waitTime to status startTime - status arriveTime
  2  increment stats numCust
  3  set stats totSvcTime  to stats totSvcTime + status svcTime
  4  set stats totWaitTime to stats totWaitTime + waitTime
  5  set queueSize to queueCount (queue)
  6  if (stats maxQueueSize < queueSize)
     1  set stats maxQueueSize to queueSize
  7  end if
  8  print (status custNum   status arriveTime
            status startTime status svcTime
            waitTime        queueCount (queue))
  9  set    moreCusts to false
 2 end if
end svcComplete
```

Aplicações de Filas
Simulação Filas – Aplicação

- A tabela ilustra a simulação dos tempos de serviço para 4 clientes. Para cada cliente, mostra-se:
  - i) o tempo de inicio do atendimento; ii) o tempo de serviço; iii) o tempo de término (considera o minuto em que será terminado o serviço) e iv) os minutos de serviço.

Start time	Service time	Time completed	Minutes served
1	2	2	1 and 2
3	1	3	3
4	3	6	4, 5, and 6
7	2	8	7 and 8

Hypothetical Simulation Service Times

Aplicações de Filas
Simulação Filas – Aplicação

- O algoritmo para o módulo de **printStats** é mostrado a seguir.
- Imprime-se dados da simulação como:
  - i) número total de clientes (numCust); ii) tempo total de serviço (totSvcTime); iii) tempo médio de serviço (avrgSvcTime); iv) Tempo médio de espera (avrgWaitTime) e v) Tamanho máximo da fila.

```
Algorithm printStats (stats)
This algorithm prints the statistics for the simulation.
  Pre    stats contains the run statistics
  Post   statistics printed
 1 print (Simulation Statistics:)
 2 print ("Total customers: "    stats numCust)
 3 print ("Total service time: " stats totSvcTime)
 4 set avrgSvcTime to stats totSvcTime / stats numCust
 5 print ("Average service time: " avrgSvcTime)
 6 set avrgWaitTime to stats totWaitTime / stats numCust
 7 print ("Average wait time: " avrgWaitTime)
 8 print ("Maximum queue size: " stats maxQueueSize)
end printStats
```

Calcula o tempo médio de serviço

Calcula o tempo médio de espera

Referências

- Gilberg, R.F. e Forouzan, B. A. Data Structures_A Pseudocode Approach with C. Capítulo 4. Queues. Segunda Edição. Editora Cengage, Thomson Learning, 2005.