Projeto Lava-lava

Descrição

A cadeia Lava-lava vai instalar uma máquina de lavagem de automóveis e pretende estudar a sua operação.

A Lava-lava tem as seguintes regras/dados:

- A máquina fica aberta um dado número de horas.
- Pôr a máquina a funcionar tem um custo fixo de 200 euros (não considerando o custo das lavagens).
- Um carro ao permanecer 10 ou mais minutos em fila desiste e sai da fila.
- Após chegar a hora de fecho, impede-se a entrada de novos carros, mas lavam-se os restantes carros ainda na fila de espera e os carros já não desistem.
- Quando termina uma lavagem, se existir algum carro em espera, nesse mesmo minuto inicia-se a lavagem do primeiro carro na fila.
- O intervalo de tempo entre chegadas consecutivas (em minutos) tem uma distribuição uniforme num determinado intervalo dado.

Máquina de lavagem "neutral - the kings computer wash the car machine"[1]

A Lava-lava disponibiliza três tipos de lavagem, com os seguintes nomes, duração em minutos, respectivamente.

Tipo de lavagem	Duração (em minutos)	Percentagem relativa de nº de lavagens
lavagem básica (tipo 0)	5	50
lavagem com cera (tipo 1)	7	30
lavagem *deluxe* (tipo 2)	10	20

A cada automóvel está associada a sua ordem de chegada (1º, 2º, etc), o minuto de chegada e o tipo de lavagem pretendida, que são definidos pelo cenário de chegadas. Um cenário de chegadas especifica de quantos em quantos minutos chega um novo carro. A cadência de pedidos para cada tipo de lavagem está previamente definida conforme a tabela acima.

Objectivos do projecto

Pretende-se desenvolver uma aplicação que, dado um cenário de chegadas, faça a gestão da máquina de lavagem, fazendo uso dos tipos de dados abstractos (TDA) *Fila de Carros com Desistências* e *Máquina*, a desenvolver pelos alunos, e forneça um conjunto de indicadores operacionais e financeiros sobre o funcionamento da máquina.

[1] Todos os direitos reservados ao autor da imagem.

Funcionamento

A aplicação inicia com a máquina vazia e no minuto zero (vamos considerar como unidade de tempo o minuto). As chegadas ocorrem de acordo com um determinado cenário (conforme se descreve mais abaixo) e o comportamento relativamente à fila de espera é conforme descrito nas regras da Lava-lava. A aplicação deve usar um "relógio" que faz parte da máquina e executa uma simulação conforme se descreve. A cada minuto do relógio é inspeccionado se há algum evento (ou vários eventos) a ocorrer nesse minuto.

1. Se ocorre uma chegada,
    a. gera-se o tipo de lavagem que o cliente vai escolher (recorrendo a um gerador que se disponibiliza com este enunciado);
    b. actualizam-se a ordem e o minuto de chegada do carro (é minuto actual no relógio da máquina);
    c. faz-se entrar na fila esse carro;
    d. gera-se o minuto em que vai ocorrer a próxima chegada e guarda-se esta informação na máquina;
    e. actualiza-se o indicador relativo ao número de carros que entraram no serviço.
2. Se ocorre um fim de lavagem,
    a. a máquina passa a desocupada;
    b. actualizam-se o número total de carros atendidos, o tempo total dos atendidos no sistema, o número de lavagens do tipo do carro atendido.
3. Se ocorre o início duma lavagem – ocorre quando a máquina está desocupada e a fila não está vazia:
    a. guarda-se na máquina a informação relativa ao carro a lavar;
    b. agenda-se o minuto de próximo fim de lavagem, de acordo com o tipo de lavagem do carro que se vai lavar (por exemplo, se no minuto 32 se inicia uma lavagem básica, o próximo fim de lavagem é o minuto 37);
    c. faz-se sair da fila o primeiro carro;
    d. actualiza-se o indicador relativo ao tempo total de espera dos atendidos.

A cada minuto do relógio, os carros que estão há 10 ou mais minutos na fila desistem e abandonam a fila. A máquina termina a sua operação quando já foi atingido o tempo de operação diário e a fila está vazia.

A máquina dispõe dum parâmetro "traço" (dado como *input*) que permite controlar as alterações que vão ocorrendo, conforme os exemplos de output fornecidos com este enunciado.

Antes de terminar, a aplicação escreve os indicadores correspondentes à simulação, conforme o exemplo abaixo e liberta a memória que tiver sido reservada durante a sua execução.

Exemplo de um relatório de operação da máquina (output)

```
Relatorio
Minuto actual: 633
Conteudo da fila:carros[]
Lucro: 229 euros
Numero de carros que entraram no sistema: 110
Numero de carros atendidos: 93
Numero de lavagens atendidas por tipo 0: 48, 1: 25, 2: 20
Numero de carros que desistiram: 17
Tempo medio no sistema: 10.774
Tempo medio de espera dos atendidos: 4.161
```

TDA Fila de Carros com Desistências

Para a concretização deste TDA deve ser usada uma implementação dinâmica (pode basear-se na implementação apresentada nas aulas). Deve considerar o tipo *Carro* anexo a este enunciado (carro.h).

Operações

filac_criar – cria uma fila nova.

filac_destruir – destrói uma fila.

filac_apagar – faz sair todos os carros da fila sem a destruir.
filac_n_elems – número de elementos na fila.
filac_entrar – faz entrar um carro na fila.
filac_primeiro – o primeiro carro da fila.
filac_sair – faz sair o primeiro carro da fila.
filac_esta_vazia – determina se a fila está vazia.
filac_desistir – faz sair da fila todos os carros cujo tempo de espera é maior ou igual a k minutos, isto é, o tempo limite de espera. Um dos parâmetros desta operação é o minuto actual, outro será o tempo limite de espera.
filac_imprimir – imprime o conteúdo da fila no formato
n_elem [ordem_de_chegada minuto_de_chegada; ...; ordem_de_chegada minuto_de_chegada]
filac_erro – o valor do erro da fila.
filac_reset_erro – reinicia o erro da fila.

TDA Maquina

Tipo

Este tipo deve ser concretizado recorrendo a uma estrutura onde os campos guardam:

- o relógio da máquina;
- a fila de espera;
- os lucros por tipo de lavagem;
- se a máquina está ou não ocupada;
- o número de carros que entram no sistema e o número de carros que são atendidos por tipo de lavagem;
- o tempo total de espera dos atendidos e o tempo total no sistema (tempo de espera mais tempo de lavagem) dos atendidos;
- e os restantes indicadores que considerar necessários.

Como é a máquina que controla a simulação, todas as variáveis que dizem respeito à simulação devem ser guardadas na estrutura máquina, por exemplo, o minuto da próxima chegada e o minuto do próximo fim de lavagem.

Operações

maquina_criar – cria um máquina nova com :uma fila vazia, a informação dos valores de lucro por cada tipo de lavagem, a informação relativa ao cenário de chegadas (limite inferior e limite superior do intervalo que define o número de minutos entre chegadas sucessivas), o tempo total da simulação, a informação relativamente ao traço da simulação estar ou não activo, o minuto da primeira chegada que deve estar inicializado (próxima chegada) e todos os indicadores a zero.
maquina_destruir – destrói a máquina.
maquina_chegar – processa uma chegada de acordo com as regras.
maquina_iniciar_lavagem – processa o início duma lavagem.
maquina_sair – processa o fim de uma lavagem.
maquina_desistir – processa as desistências dos carros que estão na fila da máquina.
maquina_processar – processa a simulação recorrendo às outras operações do TDA.
maquina_printf – escreve o conteúdo da máquina apresentando o conteúdo da fila, quantos carros entraram no sistema, quantos carros desistiram, o tempo médio de espera por carro atendido, o tempo médio no sistema por carro atendido (inclui o tempo de espera e o tempo de atendimento) e o lucro diário da máquina (lucro total das lavagens efectuadas - custo fixo de operação), conforme se exemplifica no ficheiro de output.
maquina_erro – o valor do erro do máquina.
maquina_reset_erro – reinicia o erro do máquina.

Formato do ficheiro de dados
O ficheiro de dados inclui os lucros por tipo de lavagem e os limites do intervalo que define o tempo entre chegadas (ao minuto)
1ª linha – lucros por tipo de lavagem
2ª linha – limite inferior e limite superior do intervalo de tempo que decorre entre chegadas sucessivas e que define o cenário de chegadas
3ª linha – número de horas de operação da máquina
4ª linha – 1, se pretendemos ver o traço da simulação; 0, caso contrário.

Exemplo de ficheiro de dados:
```
3 5 8
1 5
2
1
```
representa que

- a lavagem do tipo básica tem um lucro de 3 euros, a lavagem com cera tem um lucro 5 e a *deluxe* tem um lucro de 8;
- os carros chegam com um tempo entre chegadas consecutivas dado por um número pseudo-aleatório inteiro entre 1 e 5 minutos (*inclusive*);
- a simulação deve estudar duas horas de operação da máquina
- pretendemos ver o traço da simulação.

Com este enunciado, são disponibilizados alguns ficheiros de dados e respetivos outputs resultantes de execuções que não activam a opção *inicia* do gerador de números pseudo-aleatórios (para permitir a comparação dos resultados.)

O que devem fazer
Implementar os TDA acima e testá-los com a aplicação cliente "runMaquina". Deve fazer o desenvolvimento do seu projecto de uma forma incremental. Para tal, deve executar as seguintes tarefas.

1. Criar o TDA Fila de Carros com Desistências (fila_carros.h e fila_carros.c) e adaptar o programa usaFila.c na página da disciplina para usar testar este novo TDA.

2. Criar o TDA Maquina.

3. Testar o TDA Maquina com a aplicação dada runMaquina.c. Note que esta aplicação necessita da entrada do nome de um ficheiro de dados na linha de comandos. Para tal, execute

```
$ runMaquina dados.txt
```

***Guidelines* para o desenvolvimento**

Quando precisar de usar mais do que um TDA deve incluir os respectivos ficheiros "ficheiroTDA.o".

Tenha em atenção o seguinte:
- O tipo *Carro* tem de ser conhecido simultaneamente pelo TDA *Fila de Carros com Desistências* e pelo TDA *Maquina.*
Deve, por isso, fazer #include "carro.h" nos ficheiros adequados.
- A leitura de um carro a partir do teclado pode ser feita com as seguintes instruções:
```
printf( "Introduza carro: ordem_chegada minuto_chegada tipo_lavagem\n" );
scanf( "%d %d %d", &ordem_chegada, &minuto_chegada, &tipo_lavagem);
```

É disponibilizado um módulo para fazer a geração de números pseudo-aleatórios, relativamente ao qual deve fazer #include "gerador.h" nos ficheiros adequados.
Esse módulo disponibiliza as seguintes funções

- `void inicia( void )` – que permite inicializar a sequência de números gerados em função do tempo do sistema. Se esta função não for invocada, a sequência de números gerados é sempre a mesma.
- `int geraInteiro( int min, int max)` – que permite gerar um número pseudo-aleatório inteiro no intervalo [min;max].
- `int geraTipoLavagem( void )` – que gera um tipo de lavagem de acordo com as percentagens por ocorrência de tipo de lavagem definidas neste enunciado.

Enquanto estiver a fazer a depuração do seu programa deve usar sempre a mesma sequência de números pseudo-aleatórios. Para isso, no ficheiro `runMaquina.c` não deve fazer a invocação da função `inicia`. Só quando pretender testar diferentes simulações deve invocar a função `inicia`. O mesmo se passa para a opção traço, que deve usar com o valor 1 enquanto estiver a realizar testes.

Onde e até quando devem entregar

Devem entregar o resultado de cada fase do desenvolvimento, na página da disciplina no mocho (mocho.di.fc.ul.pt), no sítio "Entrega do projecto – milestone X", de acordo com o seguinte calendário:
tarefa 1 – *milestone* 1 - **22 de Abril de 2013** às 23h (vale 50%).
tarefa 2 e tarefa 3 – *milestone* 2 - **10 de Maio de 2013** às 23h (vale 50%).

A possibilidade de entrega em cada *milestone* é encerrada automaticamente à hora prevista. O ficheiro a entregar pode ser carregado no mocho por qualquer dos elementos do grupo, mas isso deve ser feito por apenas um deles e não por ambos. O ficheiro pode ser reenviado (re-carregado) várias vezes, mas tem de ser sempre pelo aluno que o submeteu a primeira vez, ficando apenas registada a última versão.
A falta de entrega de uma tarefa até à data prevista não é eliminatória (excepto para a tarefa final). No entanto, será penalizada em 50% da respectiva nota parcial. No *milestone* 2 devem corrigir erros que entretanto detectem no código entregue no *milestone* 1.

O projecto **deve ser feito em grupos de dois elementos**. Projectos entregues individualmente, sem autorização prévia da Professora responsável pela disciplina, serão fortemente penalizados.

O que devem entregar

Em cada *milestone*, devem fazer um ficheiro compactado (.zip) com todos os ficheiros de código (.c e .h) que criaram até ao fim da fase correspondente. No *milestone* 2, devem incluir a versão final de runMaquina.c. Não incluam ficheiros executáveis nem ficheiros de dados/resultados. Não é preciso entregar nada em papel; devem apenas carregar o ficheiro, isto é, fazer o seu *upload*, no mocho. O ficheiro zip a entregar deve ter o nome
fcXXXXX_fcYYYYY.zip, onde XXXXX é o número do 1º elemento do grupo e YYYYY é o número do 2º elemento.
Atenção, não entregue ficheiros .zipx ou .rar.

Todos os ficheiros entregues devem ter no cabeçalho a descrição do ficheiro, assim como o nº e nome de cada autor.
Exemplo para maquina.c

```
/**
 * Implementação do TDA Maquina
 * autores: fc50000 António Vindouro e fc500001 Manuel Vindouro
 */
```

Cuidados a ter com a compilação e execução

Todas as funções devem ser documentadas, através de um comentário antes do início da definição da função e, eventualmente, comentários no seu corpo.

Todo o programa deve estar devidamente indentado e não conter linhas que excedam os 80 caracteres.

Antes de entregar confirme se o seu projecto compila com a versão do compilador gcc instalado nos laboratórios do DI, com as opções –Wall –ansi em ambiente Linux. Programas que dêem erros de compilação

serão avaliados com zero. Projectos com *warnings* serão penalizados. Podem entregar apenas o resultado de alguns dos passos do desenvolvimento caso não tenham completado todo o projecto.

Os ficheiros de implementação dos TDA devem ser compilados com o seguinte comando:

```
gcc -Wall -ansi -c nomeficheiro.c
```

Os ficheiros que implementam aplicações devem ser compilados com o seguinte comando:

```
gcc -Wall -ansi -o nomeFicheiro nomeficheiro.c ficheiroTDA.o
```

Ficheiros

carro.h

```c
/**
 * Interface para o tipo Carro (projecto Lava-lava)
 * @author respicio
 * @Date April 2013
 */

#ifndef _CARRO
#define _CARRO


/* tipo carro */
typedef struct carro {
  int ordemChegada;
  int minChegada;
  int tipoLavagem;
} Carro;

#endif
```

runMaquina.c

```c
/**
 * Aplicacao que permite testar o TDA Maquina
 * @author respicio
 * @Date April 2013
 */

#include <stdlib.h>
#include <stdio.h>
#include "maquina.h"
#include "gerador.h"

int main( int argc, char *argv[] ) {

  Maquina maq;
  FILE *dados;
  int lucro0, lucro1, lucro2, limInfChegadas, limSupChegadas,
          tempoSimulacao, traco;

  if ( argc != 2 )
  {
     printf( "Deve executar o programa indicando o nome do ficheiro \n"
             "de dados a considerar \n"
             "Exemplo: runMaquina dados.txt\n" );
     exit( 1 );
  }

  dados = fopen( argv[ 1 ], "r" );
  if (dados == NULL )
  {
     fprintf( stderr, "O ficheiro %s não existe \n", argv[1] );
     exit( 1 );
  }

  fscanf( dados, "%d%d%d", &lucro0, &lucro1, &lucro2 );
  fscanf( dados, "%d%d%d", &limInfChegadas, &limSupChegadas, &tempoSimulacao );
  fscanf( dados, "%d", &traco );
```

```
  /* cria maquina vazia com os parametros que definem
   * os lucros, o cenario de chegadas e o tempo da simulacao */
  maq = maquina_criar( lucro0, lucro1, lucro2, limInfChegadas,
          limSupChegadas, tempoSimulacao, traco );

  if ( maquina_erro() != MAQUINA_OK )
  {
      fprintf( stderr, "Erro ao criar maquina \n" );
      exit( maquina_erro() );
  }

  /* inicia o gerador - RETIRAR A INVOCACAO DE COMENTARIO APOS TESTES */
  /*inicia(); */

  /* processa a simulacao */
  maquina_processar( maq );

  /* escreve conteudo da maquina */
  maquina_printf( maq );

  /* liberta a memoria afecta a maquina */
  maquina_destruir( maq );

  /* fechar o ficheiro dados */
  fclose( dados );

  return 0;
}
```

gerador.c

```
/**
 * Modulo gerador de numeros pseudo-aleatorios.
 *
 * author: respicio
 * date: April, 2013
 */

#include "gerador.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h> /* para poder usar funcoes sobre o system time */
#define PERCT_0 50
#define PERCT_1 30
#define PERCT_2 20

/**
 * Inicializacao do gerador de numeros pseudo-aleatorios de forma a que
 * a sequencia de chamadas as funcoes geraInteiro e geraLavagem
 * seja dependente do system-time. Se esta funcao nao for previamente
 * invocada, a sequencia de valores obtidos eh sempre a mesma. O algoritmo da
 * funcao rand() usa uma semente para gerar uma sequencia de numeros,
 * por isso eh necessario usar o srand para fazer variar essa semente.
 * A constante RAND_MAX esta definida na standard library (stdlib).
 */
void inicia( void ) {
  srand( (unsigned) time (NULL));
}

/**
 * Retorna um numero inteiro pseudo-aleatorio no intervalo [min, max]
 * @param min o limite inferior do intervalo
 * @param max o limite superior do intervalo
 */
int geraInteiro( int min, int max) {
  double v;
  int na = rand();
  v = (double) na / (double) RAND_MAX; /* Numero pseudo-aleatorio em [0,1] */
  na = (int) ( v * ( max - min + 1 ) + min );
  /* Numero pseudoaleatorio inteiro em [min,max] */
  return na;
}

/**
 * um tipo de lavagem gerado de acordo com as percentagens definidas no
 * projecto Lava-lava
```

```
 */
int geraTipoLavagem( void ) {
  int na = geraInteiro( 0, 100 );
  if ( na <= PERCT_0 )
        return 0;
  else
          if ( na <= PERCT_0 + PERCT_1 )
                return 1;
      else
                return 2;
}
```