Raul Julian Rocha Silva

Trabalho Prático de **AEDS2**

Introdução

Problemas são questões propostas em busca de uma solução. Com o propósito de conceder uma solução para certo problema, existem os algoritmos, cada problema que é decidível possui um algoritmo que determina uma solução para cada instância desse problema.

Algoritmos descrevem passo a passo os procedimentos para chegar a uma solução de um problema e podem ser representados de três formas:

A forma de descrição narrativa, na qual se usa a linguagem nativa de quem escreve. Essa forma não segue um padrão definido e pode sofrer várias interpretações por quem lê;

O objetivo deste trabalho, portanto, é apresentar diferentes soluções alterando-se variáveis para solucionar o seguinte problema:

"Todos os dias, muitas pessoas almoçam no restaurante da cantina do ICEX. Um dos problemas enfrentados pelos usuários, no entanto, são as grandes filas enfrentadas para pagar pela comida e se servir. Além disso, a quantidade de bandejas e pratos disponíveis é limitada, podendo ocorrer falta desses recursos dependendo da situação Implementação. A pessoa que gerencia a cantina quer saber quanto tempo, em média, que os usuários levam para completar seu atendimento (entrada na fila até terminar de servir os alimentos) em um espaço de 4 h.";

Desenvolvimento

Para a implementação do trabalho foi foi criado uma série de variáveis para controlar o sistema, que são:

#define MAX_MESSAGES 10
Define o máximo de mensagens padrões do sistema

#define MAX_TRAYS 30
Define o máximo de pratos que pode conter na pilha

#define RESTITUTION_AMOUNT_FOOD 10
Define a quantidade de pratos que são repostos na pilha

#define CLIENTS_PER_TIME 2
Define a quantidade de cliente que chega na fila a cada instante de tempo

#define RESTITUTION_INTERVAL_SERVICE 12
Define o intervalo de reposição dos pratos na pilha

#define SYSTEM_INTERVAL_TIME 240
Define o espaço de tempo do sistema

#define SERVICE_INTERVAL_TIME 1
Define o tempo para atendimento do cliente

#define FOOD_INTERVAL_TIME 4
Define o tempo que o cliente demora para se servir

#define AMOUNT_TAB_QUEUE 2
Define a quantidade de filas para fichas

#define AMOUNT_SERVICE_QUEUE 2
Define a quantidade de filas para servir

Funções:

bool stack_full(Stack *stack);
Retorna se pilha cheia
O(1)

bool stack_empty(Stack *stack);
Retorna se pilha vazia
O(1)

Stack* stack_create();
Retorna criação de Pilha
O(1)

void stack_up(Stack *stack);
Empilha
O(1)

void stack_remove(Stack *stack);
Desempilha
O(1)

void stack_print(Stack *stack);
Printa pilha
O(N)

Node* queue_createHead();
Cria cabeça da fila
O(1)

void queue_createRandomNode(Node *head);
Insere elemento aleatório
O(N)

void queue_addNewNode(Node *head, Node *node) ;
Adiciona novo nó.
O(N)

Node* queue_removeNode(Node *head);
Remove um nó.
O(1)

void queue_printList(Node *head);
Printa os elementos da fila
O(N)

float queue_length(Node *head) ;
Retorna a quantidade de elementos da fila.
O(N)

void checkServiceQueue(Node *head, Stack *p);
Checa a fila de serviços e atualiza.
O(N)

void checkSystem(int cTime, Node **f1, Node **f2, Stack *p);
Chega as funcionalidades do sistema.
O(N^2)

Programa Principal
O programa principal se baseia em uma sequência de passos, que são :

1) Cria as variáveis **tab_queue** e **service_queue** que são responsáveis por armazenar as N filas que o usuário definiu.
2) Plates_stack e clients_served inicia a pilha de prato e os clientes que foram atendidos com sucesso.
3) 2 FOR criam as N filas solicitados e para cada intervalo de tempo (t = 1), checkSystem() é requisitada para verificar o que está acontecendo no intervalo de tempo passado.
4) Dentro da função, é adicionado CLIENTS_PER_TIME pessoas nas filas disponíveis. Lembrando que as filas são preenchidas na posição de acordo com o resto do tamanho máximo. Isso significa que à medida que tem fila disponível ele insere um novo cliente na fila (Quanto maior o número de filas, maior será a performance); Os elementos são inseridos de forma totalmente aleatória.
5) Se o tempo atual for igual ao intervalo para atender o cliente (ou seja, se um cliente foi atendido), ele retira este cliente da fila de fichas e o adiciona nas N filas do self-service;
6) Feito isso, para cada fila de self-service, ele chama o checkServiceQueue que é responsável por analisar esta fila e ver se alguma coisa aconteceu neste exato momento.
7) É criado 3 condições dentro da função (Cliente movendo para pegar prato, cliente pegando prato e cliente finalizado). Para essas 3 situações, lógicas diferentes são executadas em cada fila. Para analisar essas situações, eu criei um count->sTime que começa de zero e vai até FOOD_INTERVAL_TIME, para controlar se o cliente acabou ou não de servir a comida.
8) Feito isso, é analisado se é o exato momento para reabastecer a pilha (RESTITUTION_INTERVAL_SERVICE) e assim é inserido, caso necessário, a quantidade de pratos definida na variável.
9) Estes pratos são inseridos somente se a pilha não ultrapassa sua capacidade.
10) Caso não exista pratos na pilha, o count->sTime do cliente permanece inalterado até que novos pratos apareçam na pilha.

1. Testes
Para analisar, não colocarei o LOG completo nesses casos maiores, apenas resultados finais:

Configuração **PADRÃO**:

#define MAX_MESSAGES 10
#define MAX_TRAYS 30
#define RESTITUTION_AMOUNT_FOOD 10
#define CLIENTS_PER_TIME 2
#define RESTITUTION_INTERVAL_SERVICE 12
#define SYSTEM_INTERVAL_TIME 240
#define SERVICE_INTERVAL_TIME 1
#define FOOD_INTERVAL_TIME 1

#define AMOUNT_TAB_QUEUE 1
#define AMOUNT_SERVICE_QUEUE 1

1 filas de fichas e 1 filas de self-service

========== DATA ANALYSIS ==========

TAB QUEUE 1 LENGTH: 240

SERVICE QUEUE 1 LENGTH: 69

CLIENTS SERVED QUEUE LENGTH: 171

TIME TO ATTEND 1 CLIENT: 1.4 minutes

#define AMOUNT_TAB_QUEUE 2
#define AMOUNT_SERVICE_QUEUE 2

2 filas de fichas e 2 filas de self-service

========== DATA ANALYSIS ==========

TAB QUEUE 1 LENGTH: 0

TAB QUEUE 2 LENGTH: 0

SERVICE QUEUE 1 LENGTH: 88

SERVICE QUEUE 2 LENGTH: 240

CLIENTS SERVED QUEUE LENGTH: 152

TIME TO ATTEND 1 CLIENT: 1.6 minutes

6 filas de fichas e 4 filas de self-service

========== DATA ANALYSIS ==========

TAB QUEUE 1 LENGTH: 0

TAB QUEUE 2 LENGTH: 0

TAB QUEUE 3 LENGTH: 0

TAB QUEUE 4 LENGTH: 0

TAB QUEUE 5 LENGTH: 4

TAB QUEUE 6 LENGTH: 4

SERVICE QUEUE 1 LENGTH: 4

SERVICE QUEUE 2 LENGTH: 4

SERVICE QUEUE 3 LENGTH: 8

SERVICE QUEUE 4 LENGTH: 8

CLIENTS SERVED QUEUE LENGTH: 456

TIME TO ATTEND 1 CLIENT: 0.5 minutes

LOG COMPLETO PARA PEQUENO CASO:

Neste exemplo é apresentado PASSO A PASSO de todas as mudanças dentro do sistemas, desde à configuração das filas/pilha até o final do sistema com a análise de dados.

Método que eu adotei para facilitar os testes e entendimento por parte do professor em relação ao projeto.

CONFIGURAÇÃO:

#define MAX_MESSAGES 10
#define MAX_TRAYS 30
#define RESTITUTION_AMOUNT_FOOD 2
#define CLIENTS_PER_TIME 2
#define RESTITUTION_INTERVAL_SERVICE 3
#define SYSTEM_INTERVAL_TIME 10
#define SERVICE_INTERVAL_TIME 1
#define FOOD_INTERVAL_TIME 1

//VARIABLES TO CONTROL
#define AMOUNT_TAB_QUEUE 1
#define AMOUNT_SERVICE_QUEUE 1

TIME 1
Adding 2 new peoples to tab queue...
Attending the Client 1 from queue 1 and add him to service queue...
Client 1 is moving to get plate

TIME 2
Adding 2 new peoples to tab queue...
Attending the Client 2 from queue 1 and add him to service queue...
Client 1 can't get plate because the stack is empty!
Client 2 is moving to get plate

TIME 3
Adding 2 new peoples to tab queue...
Attending the Client 3 from queue 1 and add him to service queue...
Client 1 can't get plate because the stack is empty!
Client 2 can't get plate because the stack is empty!
Client 3 is moving to get plate

Puting 2 plates on stack

TIME 4
Adding 2 new peoples to tab queue...
Attending the Client 4 from queue 1 and add him to service queue...
Removing one plate of the stack of Client 1
Client 2 can't get plate because the stack is empty!
Client 3 can't get plate because the stack is empty!
Client 4 is moving to get plate

TIME 5
Adding 2 new peoples to tab queue...
Attending the Client 5 from queue 1 and add him to service queue...
The client 1 was served!
Client 2 can't get plate because the stack is empty!
Client 3 can't get plate because the stack is empty!
Client 4 can't get plate because the stack is empty!
Client 5 is moving to get plate

TIME 6
Adding 2 new peoples to tab queue...
Attending the Client 6 from queue 1 and add him to service queue...
Client 2 can't get plate because the stack is empty!
Client 3 can't get plate because the stack is empty!
Client 4 can't get plate because the stack is empty!
Client 5 can't get plate because the stack is empty!
Client 6 is moving to get plate
Puting 2 plates on stack

TIME 7
Adding 2 new peoples to tab queue...
Attending the Client 7 from queue 1 and add him to service queue...
Removing one plate of the stack of Client 2
Client 3 can't get plate because the stack is empty!
Client 4 can't get plate because the stack is empty!
Client 5 can't get plate because the stack is empty!
Client 6 can't get plate because the stack is empty!
Client 7 is moving to get plate

TIME 8

Adding 2 new peoples to tab queue...
Attending the Client 8 from queue 1 and add him to service queue...
The client 2 was served!
Client 3 can't get plate because the stack is empty!
Client 4 can't get plate because the stack is empty!
Client 5 can't get plate because the stack is empty!
Client 6 can't get plate because the stack is empty!
Client 7 can't get plate because the stack is empty!
Client 8 is moving to get plate

TIME 9
Adding 2 new peoples to tab queue...
Attending the Client 9 from queue 1 and add him to service queue...
Client 3 can't get plate because the stack is empty!
Client 4 can't get plate because the stack is empty!
Client 5 can't get plate because the stack is empty!
Client 6 can't get plate because the stack is empty!
Client 7 can't get plate because the stack is empty!
Client 8 can't get plate because the stack is empty!
Client 9 is moving to get plate
Puting 2 plates on stack

TIME 10
Adding 2 new peoples to tab queue...
Attending the Client 10 from queue 1 and add him to service queue...
Removing one plate of the stack of Client 3
Client 4 can't get plate because the stack is empty!
Client 5 can't get plate because the stack is empty!
Client 6 can't get plate because the stack is empty!
Client 7 can't get plate because the stack is empty!
Client 8 can't get plate because the stack is empty!
Client 9 can't get plate because the stack is empty!
Client 10 is moving to get plate

TAB QUEUE 1

Client 11
Client 12
Client 13
Client 14
Client 15
Client 16

Client 17
Client 18
Client 19
Client 20

SERVICE QUEUE 1

Client 3
Client 4
Client 5
Client 6
Client 7
Client 8
Client 9
Client 10

CLIENTS SERVED QUEUE

Client 1
Client 2

========== DATA ANALYSIS ==========

TAB QUEUE 1 LENGTH: 10

SERVICE QUEUE 1 LENGTH: 8

CLIENTS SERVED QUEUE LENGTH: 2

TIME TO ATTEND 1 CLIENT: 5.0 minutes

Conclusão

Com este método que utilizei, a melhor combinação seria uma quantidade maior de filas, tanto de self-service quanto de fichas, pois o cliente seria atendido mais rápido e existiria várias ramificações para que ele pudesse se servir.

Como foi demonstrado no exemplo acima, à medida que vai aumentando as variáveis, o tempo de atendimento do cliente vai diminuindo, ou seja, vai ficando mais eficiente o atendimento.

Outras varíáveis acima podem ser alteradas, como o intervalo de tempo em relação às mudanças no projeto, mas para a análise preferi manter todas elas iguais e alterar apenas as filas. Mas, caso o professor deseje, pode alterar normalmente as variáveis e rodar o projeto novamente.

Para versões futuras, é uma intenção minha separar o arquivo main.c em outros arquivos .c com seus respectivos headers; Não consegui fazer isso porque perdi muito tempo na lógica para análise do instante de tempo, o que era prioritário para o funcionamento perfeito do meu trabalho e apresentou alguns problemas para o modelo dinâmico que criei, sendo o trabalho inteiro controlado por variáveis do sistema.