DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
MATA40 – Estrutura de Dados e Algoritmos I
Prof. Roberto Freitas Parente
roberto.parente@ufba.br

Trabalho 01
Versão: 19 de junho de 2018

1 Instruções:

- O trabalho é em equipe de no máximo **2 alunos**. O nome dos integrantes devem estar no cabeçalho comentado em todos os arquivos.
- Será aplicado detector de plágio! Uma vez detectado plágio, a **nota da disciplina** será **zerada**.
- Data de entrega: ~~17 de junho~~ **24 de junho** no sistema do moodle até as 23:59.
- A entrega deve ser no formato ZIP com o seguinte conteúdo:

  + Codigo-fonte (bem como os arquivos `.c` e `.h` referentes aos TAD utilizados);

  + Arquivo Makefile[1] associado para compilar o código.
- O código deve estar escrito em `Linguagem C` e as únicas bibliotecas permitidas são: `stdio.h`, `stdlib.h` e `string.h` caso contrário o trabalho será invalidado.
- Para validação do código será utilizado um ambiente GNU/Linux com o compilador `GCC` (versão 7+) dado os seguintes parâmetros[2]:
  `gcc -Wall -Wextra -Werror -Wpedantic`
- Para verificação de vazamento de memória e gerência dos ponteiros, será utilizado o software `Valgrind`[3].

2 Correção

Serão utilizados os seguintes critérios para correção:

2.1 Compilação

- Uso correto do Makefile.
- Compilação com parâmetros requeridos.

2.2 Entrada/Saída

- Cada entrada gera uma saída correta.

2.3 TAD / Estrutura de dados

- Implementação correta das estruturas de dados.
- Implementado um TAD por estrutura.
- Implementação correta do conceito de TAD.

2.4 Gerência de memória

- Teste de alocação.
- Remoção correta da memória alocada.
- Vazamentos de memória.
- Controle de ponteiros.

2.5 Organização do código

- Funções separadas para cada uma das operações.
- Comunicação por retorno de função.
- Fluxo da simulação.
- Documentação do código.

[1] Veja mais sobre em `https://pt.wikibooks.org/wiki/Programar_em_C/Makefiles`.
[2] Leia mais sobre em: `https://gcc.gnu.org/onlinedocs/gcc-6.4.0/gcc/Warning-Options.html`.
[3] `http://valgrind.org`

3 Simulação de um cartório

Um cartório é uma repartição pública ou privada que tem como principal função a custódia de documentos com fé pública, ou seja, é uma organização burocrática que organiza documentos importantes ☺☻. Desta forma, nosso objetivo é simular uma versão simplificada de um cartório que gerencia a aquisição e transferência de bens através do atendimento de clientes pelos guichês do cartório.

Cada simulação deve receber como entrada a quantidade de guichês e clientes bem como informações das operações a serem executadas. Ademais, ao final a simulação deve gerar um relatório parcial informando a situação de cada guichê e um relatório final sobre todos bens dos CPFs envolvidos na simulação.

3.1 A simulação

Para cada simulação, informamos que o cartório contém M guichês, recebe N clientes durante um dia e L níveis de prioridade. Cada cliente tem os seguintes dados: Código de pessoa física (CPF), operação, o bem, CPF de terceiro envolvido na operação e sua prioridade. Cada dado tem a seguinte forma:

- CPF cliente: inteiro;
- Operação: caractere 'A' para aquisição e 'T' para transferência;
- Bem: string;
- CPF terceiros: inteiro;
- Prioridade: inteiro.

A entrada de cada cliente é gerenciada por L filas, onde cada fila representa uma prioridade. Deve-se escolher o cliente referente a fila de maior prioridade e alocá-lo em um guichê disponível utilizando a estratégia de round–robin iniciando do guichê 0 (zero), ou seja, se o cliente anterior foi alocado no guichê $k \mod M$, então o presente cliente de ser alocado no guichê $k + 1 \mod M$ ficando a seguinte ordem dos guichês selecionados: $0, 1, 2, \ldots, M - 1, 0, 1, 2, \ldots, M - 1, 0, \ldots$.

Cada cliente realiza apenas uma operação, mas não existe restrição da quantidade de vezes que um cliente é atendido. Cada guichê atende a um cliente por vez e armazena os documentos de cada cliente em uma pilha.

Ao finalizar os atendimentos, deve ser impresso o relatório parcial de cada guichê, onde será informado a quantidade de documentos armazenados no guichê seguido dos dados de cada documento. Após a impressão dos relatórios parciais, deve-se gerar o relatório final que informa a quantidade de CPFs envolvidos nas operações e imprimir as operações de cada CPF, onde a chave para ordenação é o CPF.

3.2 Entrada

A entrada é constituída por uma única simulação enviada pela entrada padrão. A primeira linha contém três inteiros L, M e N separados por um espaço em branco, onde $1 \leq L, M, N \leq 2^{32} - 1$. L será a quantidade de níveis de prioridades, M será a quantidade de guichês e N a quantidade de clientes envolvidos na simulação. As próximas N linhas deve conter os seguintes valores C, CT, P, O e B onde $1 \leq C \leq 2^{32} - 1$ representa o CPF do cliente, $1 \leq CT \leq 2^{32} - 1$ o CPF do terceiro envolvido, $1 \leq P \leq L$ a prioridade do cliente C, $O \in \{A, T\}$ o caractere referente a operação e B a string de tamanho no máximo 25 referente ao nome do bem envolvido na transação.

Exemplos:

- A linha "2 123 23 A `Carro 23`" significa que o cliente com CPF 2 adquiriu 'Carro 23' do CPF 123 e tem prioridade 23 na fila de entrada do cartório.
- A linha "23 1223 3 T `CaSaA`" significa que o cliente com CPF 23 transferiu 'CaSaA' para o CPF 1223 e tem prioridade 3 na fila de entrada do cartório.

Ademais, teremos a seguinte estrutura geral da entrada:

Entrada:
$L\ M\ N$
$C_1\ CT_1\ P_1\ O_1\ B_1$
$C_2\ CT_2\ P_2\ O_2\ B_2$
$\vdots$
$C_N\ CT_N\ P_N\ O_N\ B_N$

3.3 Saída

A saída é constituída por dois relatórios para que devem ser enviados pela saída padrão seguindo a formatação descrita nessa seção.

Relatório Parcial

Antes do relatório parcial deve ser impressa a mensagem "`-:| RELATÓRIO PARCIAL |:-`" e seguido de um inteiro M referente a quantidade de guichês envolvidos na simulação, na linha seguinte deve ser impresso o texto "`Guiche` k`:` P_k" onde k é o número do guichê e um inteiro P_k referente a quantidade de documentos empilhados no guichê k e nas próximas P_k linhas informaŕa os valores de cada operação na formatação "`[CPFc,CPFt,O,B]`" onde "CPFc" é referente ao CPF do cliente atendido, "CPFt" o cpf do terceiro envolvido na operação, "O" o caractere referente a operação e "B" uma string referente ao nome do bem onde deve-se substituir espaços em brancos por underlines e utilizar os símbolos em maiúsculos (ex: entrada: "CasA b" relatório parcial: "CASA_B"). ~~A ordem e impressão de cada guichê deve ser a mesma de atendimento.~~ A ordem e impressão de cada guichê deve ser a mesma da remoção da pilha individual.

Relatório Final

Antes do relatório final deve ser impressa a mensagem "`-:| RELATÓRIO FINAL |:-`". Na linha seguinte um inteiro K informa a quantidade de CPFs envolvidos na simulação e o relatório deve seguir a formatação "`-:[ CPF`$_\ell$`:` B_ℓ" , onde "CPF$_\ell$" um inteiro que indica o ℓ-ésimo CPF e B_ℓ um inteiro referente a quantidade de bens envolvidos nas operações do CPF$_\ell$ seguindo por uma lista ordenada crescente (alfanumericamente) dos B_ℓ bens com a seguinte formatação: `\tSBem`, onde \t é o símbolo de tabulação, "S" deve ser `+` para aquisição e `-` para transferência, e "Bem" o nome do bem com espaços em brancos substituídos por underlines e símbolos em maiúsculos.

Exemplo: O CPF 12623 adquiriu os bens "CasA V", "Piscina" e transferiu os bens "sapato podre" e "aviao". O relatório final deve constar da seguinte forma:

```
-:[ 12623:  4
  -AVIAO
  +CASA_V
  +PISCINA
  -SAPATO_PODRE
```

Observação: `O relatório final deve constar TODOS os CPFs envolvidos na simulação e não apenas os CPFs de clientes.`

Saída final

Como descrito acima, segue os seguintes parâmetros para formatação da saída:

- M: Quantidade total de guichês.
- P_k: Quantidade de documentos armazenados no guichê k.
- $CPFc^i_j$: CPF do cliente referente ao j-ésimo documento desempilhado do guichê i.
- $CPFt^i_j$: CPF de terceiros referente ao j-ésimo documento desempilhado do guichê i.
- O^i_j: Operação do j-ésimo documento desempilhado do guichê i.
- B^i_j: Nome formatado do bem envolvido no j-ésimo documento desempilhado do guichê i.
- K: Quantidade total de CPF's distintos envolvidos (cliente e terceiros).
- CPF_ℓ: ℓ-ésimo CPF envolvido na simulação (ordenados).
- B_ℓ: Quantidade de bens que envolveram o ℓ-ésimo CPF.
- $\text{S}^i_j \in \{\texttt{+},\texttt{-}\}$: Sinal referente a operação do j-ésimo bem do i-ésimo CPF da simulação.
- Bem^i_j: Nome formatado do j-ésimo bem do j-ésimo CPF da simulação.

Ademais, teremos a seguinte estrutura:

Saída:

-:| RELATÓRIO PARCIAL |:-
M
Guiche 1: $P1$
[CPFc^1_1,CPFt^1_1,O^1_1,B^1_1]
[CPFc^1_2,CPFt^1_2,O^1_2,B^1_2]
⋮
[CPFc^1_{P1},CPFt^1_{P1},O^1_{P1},B^1_{P1}]
⋮
Guiche M: PM
[CPFc^M_1,CPFt^M_1,O^M_1,B^M_1]
[CPFc^M_2,CPFt^M_2,O^M_2,B^M_2]
⋮
[CPFc^M_{PM},CPFt^M_{PM},O^M_{PM},B^M_{PM}]
-:| RELATÓRIO FINAL |:-
K
-:[CPF_1: $B1$
$\text{S}^1_1\text{Bem}^1_1$
$\text{S}^1_2\text{Bem}^1_2$
⋮
$\text{S}^1_{B1}\text{Bem}^1_{B1}$
⋮
-:[CPF_K: BK
$\text{S}^K_1\text{Bem}^K_1$
$\text{S}^K_2\text{Bem}^K_2$
⋮
$\text{S}^K_{BK}\text{Bem}^2_{BK}$

4 Restrições e informações adicionais

Estruturas de dados & TAD's

1. Para organizar a entrada dos clientes:
   - TAD para fila com alocação encadeada: Deve ser criada uma fila para cada prioriedade L mesmo que a fila esteja vazia.
   - TAD para fila de prioridade: Deve-se criar um TAD que irá gerenciar as L filas de entrada de cliente.

   Obs: São dois TAD diferente e o TAD de fila de prioridade deve utilizar o TAD da primeira fila descrita para gerenciar as L filas.
2. Cada guichê deve conter uma pilha implementada utilizando alocação sequencial. A simulação deve utilizar apenas o TAD da pilha.
3. Lista ordenada: A lista do relatório final deve ser uma lista duplamente encadeada circular ordenada pelo CPF.

Observação: É obrigatório que a simulação utilize o conceito de Tipo Abstrato de Dados (TAD), ou seja, os códigos dos TAD's e da simulação sejam totalmente independentes.

Modularização & Funções

A utilização do conceito de Tipos Abstratos de Dados (T.A.D.) fornece um código bastante modularizado. Porém é importante que dentro do arquivo da simulação existam funções específicas para cada uma das operações da simulação, por exemplo: Receber cliente (ler linha de entrada e alocá-los na fila de entrada), atender cliente (retirar cliente da fila e atribuí-lo a um guichê).

É importante que as funções façam suas operações independentes. Ex: não passar por parâmetro o guichê destino. Para um código bem organizado é importante que o mesmo não contenha variáveis globais e que as funções retornem valores que indiquem o seu comportamento. É bastante comum definir padrões para o retorno das funções de acordo com os possíveis comportantes da mesma. Por exemplo: remoção de um elemento: null significa que não houve remoção. Ou mesmo um inteiro que mapeie os possíveis comportamentos.

Por fim, espera-se que a simulação implemente as seguintes funções:

1. Função para ler linhas da entrada;
2. Função de inserir um cliente na fila de entrada;
3. Função de remover cliente da fila de entrada;
4. Função de enviar um cliente para atendimento;
5. Função para gerar e imprimir relatório parcial;
6. Função para gerar relatório final;
7. Função para imprimir relatório final.

Lembre-se: Tudo que é alocado, deve ser desalocado!

☻ Boa diverSão!!!! ☺