MC202GH - Estrutura de Dados - Turmas G e H
Laboratório 7 - *Fila de prioridades*

Docente: Marcelo da Silva Reis

Monitor PED: Matheus Abrantes Cerqueira

Monitores PAD: Andreas Cisi Ramos
Wallace Gustavo Santos Lima

22 de novembro de 2022

Data de entrega: 25/11/2022

Entrega no codePost[1]

Informações gerais

Neste laboratório trabalharemos com fila de prioridades, nas versões *max-heap* e *min-heap*. Para isso, como de praxe serão fornecidos arquivos com protótipos a serem modificados e enviados na plataforma de avaliação. O presente laboratório retoma o tema de administração de restaurantes, para esse fim fazendo uso de uma TAD similar àquelas vistas anteriormente.

Observações importantes:

1. Neste laboratório será permitido o uso apenas das bibliotecas `stdio.h`, `stdlib.h`, `string.h` e `math.h`, além do arquivo de interface do laboratório, que é o `restaurante.h`.
2. O arquivo `restaurante.h` que contém os protótipos de tipos e operadores do nosso TAD. Além disso, tem-se os arquivos de implementação (**restaurante.c** e **questao_x.c**, sendo x o número da questão).
3. Para cada questão, entre na pasta correspondente à mesma e compile o código utilizando o arquivo `Makefile` contindo nela.

[1] https://codepost.io/signup/join?code=ZW239C3IID

Questão 1 (10 pontos) - Lista de espera com pessoas com prioridades

Retomaremos o exemplo de lista de espera em restaurante utilizado anteriormente, porém agora trataremos de uma lista de espera **com prioridades** (por exemplo, idosos, pessoas com crianças pequenas, gestantes, etc.). Para essa tarefa utilizaremos uma fila de prioridades do tipo *max-heap* e *min-heap*; portanto, neste laboratório inserções e remoções da lista de espera deverão ser sempre realizadas em $O(\lg\ n)$, onde n é o número de pessoas na lista.

Para gerenciarmos a lista de espera, faremos uso dos seguintes comandos para chamar operações de manipulação:

- **I** *s p q*: Insere uma nova entrada com os campos sobrenome *s* (string), prioridade *p* (um inteiro entre 1 e 99), quantidade de pessoas *q* (também entre 1 e 99);
- **P**: Imprime na saída padrão todos os registros de entradas na sequência *prioridade*, *quantidade*, *sobrenome*. Exemplo:

  ```
  17 03 Fulano
  ```

  O *max-heap* deve ser impresso do primeiro nível (raiz) até o último, a esquerda para a direita.
- **A**: Libera uma mesa, deve-se então retirar a pessoa atendida;
- **L**: Mostra o primeiro e o último da fila, nessa ordem. Exemplo:

  ```
  Fila: [Fulano 23 ... Ciclano 02]
  ```

  Caso exista uma só pessoa na fila nomes e prioridade do primeiro do último da fila devem ser os mesmos; E se a fila estiver vazia, deve ser impresso:

  ```
  Fila: []
  ```
- **R**: Mostra a pessoa com maior prioridade (raiz do *max-heap*). Exemplo:

  ```
  Raiz: Beltrano 42
  ```
- **M** *s p*: Procura a pessoa com sobrenome *s* e muda sua prioridade para *p*; **Considere que** *s* **e** *p* **serão sempre válidos.**
- **F**: finaliza a execução do programa.

Por exemplo, se o programa desta questão receber:

```
I Fulano 7 1
L
A
L
I Lightyear 20 1
I Mouse 5 3
M Lightyear 2
L
I Donald 4 1
I Patinhas  10 1
P
L
A
I Pardal 19 2
L
A
F
```

a respectiva saída deverá ser:

```
Fila: [Fulano 07 ... Fulano 07]
Mesa disponivel para Sr(a) Fulano
Fila: []
Fila: [Mouse 05 ... Lightyear 02]
10 01 Patinhas
05 03 Mouse
04 01 Donald
02 01 Lightyear
Fila: [Patinhas 10 ... Lightyear 02]
Mesa disponivel para Sr(a) Patinhas
Fila: [Pardal 19 ... Lightyear 02]
Mesa disponivel para Sr(a) Pardal
```

Dica 1: para manter o controle de quem é o último da fila, utilize também um *min-heap*, que sempre deve ter os mesmos itens do *max-heap*.

Dica 2: quando alguém sair da lista de espera, é preciso remover a raiz do *max-heap* e também a mesma pessoa no *min-heap*.

Dica 3: para localizar a pessoa no *min-heap*, e também para localizar em ambos os heaps uma pessoa que terá sua prioridade alterada, é preciso utilizar uma das técnicas de gerenciamento de IDs que discutimos nos final da aula de filas de prioridade (Aula 23).