Sistema de Gestão de Filas em Supermercado

O objetivo deste projeto é o desenvolvimento de um sistema de gestão de filas para caixas de um supermecado. Neste sistema as filas são ordenadas por prioridade, sendo elas 1, 2 e 3. O sistema foi desenvolvido na linguagem C utilizando filas com prioridades para organizar o atendimento dos clientes.

Implementação

Foram criados os seguintes arquivos, que estão na pasta `src` para separa o código para ficar mais intuitivo:

- `cliente.h` e `cliente.c` - Tratam sobre os dados do cliente e a implementação de seus metódos;
- `filaPrioridade.h` e `filaPrioridade` - Tratam sobre a fila com prioridade e sua implementação;
- `caixa.h` e `caixa.c` - Tratam sobre os dados do caixa e implementação dos metódos;
- `menu.h` e `menu.c` - Tratam sobre a implementação dos metódos de um menu;
- `main.c` - Implementa o método main e chama o método que cria o menu.

Cliente

Os dados de cada cliente do sistema foram armazendados em um *struct* chamado `Cliete` que possui os seguintes atributos:

- **Nome** - *array* de *chars* com tamanho máximo de 100 carateres que guarda o nome do clinte.
- **Cpf** - inteiro de 11 dígitos que aramzena o CPF do cliente;
- **Prioridade** - inteiro que representa a prioridade do cliete, pode ser 1 para alta prioridade, 2 para proridade média e 3 para prioridade média;
- **Num_Itens** - inteiro que guarda a quantidade de itens no carrinho do cliente.

O cliente possui os seguintes métodos:

- **CadastrarCliente** - utilizado para cadastrar um cliente. Recebe os dados do cliente e retorna `Cliente`;
- **ImprimirCliente** - utilizado para imprimir os dados de um cliente, recebe um `Cliente`.

CadastrarCliente

O método recebe os dados do cliente a ser cadastrado e segue a seguinte lógica:

1. Instância um novo cliente chamado `novo`;
2. Inseri os dados do cliente em `novo`;
3. Retorna `novo`.

ImprimirCliente

O método recebe um cliente cadastrado e imprime no terminal a seguinte linha com os dados do cliente:

- `$Nome$ - $Cpf$ - $Prioridade$ - $Num_Itens$`

Fila com Prioridade

A fila com prioridade é formada por dois *struct* que são utilziados para estruturar a mesma. O primeiro *struct* é o `Bloco` que onde os dados da fila são armazenados, ele possui os seguintes atributos:

- **cliente** - um *struct* de Cliente onde é armazenado o cliente na fila;
- **prox** - um ponteiro para o próximo `Bloco` da fila.

O segundo *struct* é o `FilaPrioridade` que aramzena os dados gerais da fila, ele possui os seguintes atributos:

- **first** - um ponteiro para o primeiro `Bloco` da fila;
- **last** - um ponteiro para o último `Bloco` da fila;
- **qtd** - inteiro que guarda a quantidade de elementos na fila.

A fila com prioridade possui os seguintes métodos:

- **EsvaziarFila** - utilizado para esvaziar uma fila, recebe a fila;
- **InserirFila** - utilizado para inserir um elemento na fila, recebe a fila e o cliente;
- **RemoverFila** - utilizado para remover o primeiro elemento de uma fila, recebe a fila;
- **ImprimirFila** - utilizado para imprimir os dados de uma fila, recebe a fila.

EsvaziarFila

O método recebe uma fila com prioridade e segue a seguinte lógica:

1. Seta o `first` da fila como um ponteiro para um novo `Bloco` alocado;
2. Seta o `last` da fila para ser igual ao `first`;
3. Seta o próximo do `first` como nulo;
4. Seta o `qtd` da fila como 0.

InserirFila

O método recebe uma fila com prioridade e o cliente a ser inserido na fila, depois segue a seguinte lógica:

1. Se `first` igual ao `last` da fila, ou seja, fila vazia:
    1. Seta o próximo do `last` como um ponteiro para um novo `Bloco` alocado;
    2. Seta o `last` como o próximo do `last`;
    3. Seta o cliente do `last` como o cliente passado;
    4. Seta o próximo do `last` como nulo;
2. Senão estiver vazia:
    1. Seta um ponteiro temporário `tmp` como um ponteiro para um novo `Bloco` alocado;
    2. Seta o cliente de `tmp` como o cliente passado;
    3. Seta um ponteiro `aux` como o `first` da fila;
    4. Percorre a fila até o último elemento da fila ou achar um elemento que a prioridade do cliente é maior que a do cliente de `tmp` e armazena em `aux` aonde parou percorrer a fila;
    5. Seta o próximo de `tmp` como o próximo de `aux`;
    6. Seta o próximo de `aux` como `tmp`;
    7. Se o próximo de `tmp` é nulo, ou seja, `tmp` é o último elemento da fila:
        1. Seta o `last` da fila como `tmp`;
3. Aumenta em 1 o `qtd` da fila.

RemoverFila

O método recebe uma fila com prioridade e segue a seguinte lógica:

1. Verifica se a fila está vazia, se sim não faz nada;
2. Instância dois ponteiros `aux` e `tmp`;
3. `tmp` recebe o `first` da fila;
4. `aux` recebe o próximo de `tmp`;
5. Próximo de `tmp` recebe o próximo do `aux`;
6. Libera a memória de `aux`;
7. Diminui em 1 o `qtd` da fila.

ImprimirFila

O método recebe uma fila com prioridade e segue a seguinte lógica:

1. Imprime no terminal: `Nome - OPF - Prioridade - Itens:`;
2. Para cada cliente da fila:
    1. Chama o método `ImprimirCliente` para imprimir os dados do cliente.

Caixa

Os dados de um caixa do sistema são armazenados em um *struct* chamado **Caixa** que guarda os seguintes atributos:

- **Num_Id** - inteiro que guarda o número identificador do caixa.
- **Estado** - booleana que gurada se o caixa está aberto (**true**) ou feichado (**false**).
- **Fila** - `FilaPrioridade` que guarda a fila com prioridade de clientes do caixa.

O caixa possui os seguintes métodos:

- **CadastrarCaixa** - cadastra um caixa fechado e vazio no sistema, recebe o identificador e retorna `Caixa`.
- **CaixaCadastrarCliente** - cadastra um cliente na fila do caixa, recebe o caixa e os dados do cliente.
- **AtenderCliente** - atende e remove o primeiro cliente de sua fila, recebe o caixa.
- **AbrirCaixa** - abre um caixa do sistema, recebe o caixa.
- **FecharCaixa** - fecha um caixa do sistema retornando sua fila, recebe o caixa e retorna uma cópia da fila do caixa.
- **CaixaImprimirFila** - imprime a fila do caixa, recebe o caixa.
- **ImprimirEstado** - imprime o estado do caixa, recebe o caixa.

CadastrarCaixa

O método recebe o inteiro identificador do caixa e segue a seguinte lógica:

1. Instância um novo caixa chamado `novo`;
2. Inseri os dados do caixa em `novo`;
3. Seta o `Estado` do caixa como fechado (**false**);
4. Esvazia a fila do caixa usando `EsvaziarFila`;
5. Retorna `novo`.

CaixaCadastrarCliente

O método recebe um caixa e os dados do cliente a ser cadastrado, e segue a seguinte lógica:

1. Instância um novo cliente usando `CadastrarCliente` passando os dados do cliente;
2. Inseri o cliente em `Fila` do caixa usando `InserirFila`.

AtenderCliente

O método recebe um caixa e segue a seguinte lógica:

1. Utiliza o método `RemoverFila` para remover o primeiro cliente em `Fila` do caixa.

AbrirCaixa

O método recebe um caixa e segue a seguinte lógica:

1. Seta `Estado` do caixa como aberto (**true**);
2. Esvazia a fila do caixa usando `EsvaziarFila`.

FecharCaixa

O método recebe um caixa e segue a seguinte lógica:

1. Cria uma cópia da fila em `Fila` do caixa e guarda em `copia`;
2. Seta o Seta o `Estado` do caixa como fechado (**false**);
3. Esvazia a fila do caixa usando `EsvaziarFila`;
4. Retorna `copia`.

CaixaImprimirFila

O método recebe um caixa e segue a seguinte lógica:

1. Imprime no terminal: `Caixa $Num_Id$`;
2. Usa `ImprimirFila` para imprimir todos os clientes na fila do caixa.

ImprimirEstado

O método recebe um caixa e imprime no terminal a seguinte linha com os dados do caixa:

- `Caixa $Num_Id$ - $Aberto/Fechado$ - $qtd$`

Menu

O menu do sistema é gerado pelo método `Menu` que dá ao usuário do sistema as seguintes opções de operações:

1. **Cadastrar um Cliente** - Opção para cadastrar um cliente em um caixa do sistema, feita pelo método `OptionCadastrarCliente`.
2. **Atender um Cliente** - Opção para atender o cliente de um caixa, feita pelo método `OptionAtenderCliente`.

3. **Abrir ou Fechar um Caixa** - Opção para abrir ou fechar um dos caixas do sistema, feita pelo método `OptionAbrirFecharCaixa`.
4. **Imprimir a Lista de Clientes em Espera** - Opção para imprimir os clientes em espera de cada caixa do sistema, feita pelo método `OptionImprimirFilas`.
5. **Imprimir o Status dos Caixas** - Opção para imprimir o estado e o número de clientes de cada caixa do sistema, feita pelo método `OptionImprimirStatusCaixas`.

Além destes métodos foram criados dois métodos auxiliares:

- **ClearScreen** - limpa o terminal de execução antes de mostrar o menu ou antes de executar uma ação do sistema.
- **Message** - mensagem que aparece após a execução de qualquer ação do sistema.

OptionCadastrarCliente

O método recebe um *array* com todos os caixas do sistema e segue a seguinte lógica:

1. Pergunta em que caixa o cliente vai ser cadastrado;
2. Se o caixa existir contiuna, senão mostra um erro;
3. Verifica se o caixa está aberto, se estiver continua senão mostra um erro;
4. Pergunta os dados do cliente, verficando as restrições dos dados;
5. Cadastra o cliente no caixa usando o método `CaixaCadastrarCliente`;
6. Mostra uma mensagem que o cliente foi cadastrado.

OptionAtenderCliente

O método recebe um *array* com todos os caixas do sistema e segue a seguinte lógica:

1. Pergunta em que caixa o cliente vai ser cadastrado;
2. Se o caixa existir contiuna, senão mostra um erro;
3. Utiliza o método `AtenderCliente` para atender o primeiro cliente da fila do caixa;
4. Mostra uma mensagem falando que cliente do caixa escolhido foi atendido.

OptionAbrirFecharCaixa

O método recebe um *array* com todos os caixas do sistema e segue a seguinte lógica:

1. Pergunta em que caixa o cliente vai ser cadastrado;
2. Se o caixa existir contiuna, senão mostra um erro;
3. Pergunta se quer abir ou fechar o caixa;
4. Caso abrir:
    1. Verifica se caixa já estado aberto;
    2. Se não está aberto abri usando `AbrirCaixa` e mostra um mensagem falando que o caixa foi aberto;
    3. Se já estiver aberto mostra uma mensagem falando que o caixa já está aberto;
5. Caso fechar:
    1. Verifica se é o último caixa aberto se for não deixa fechar até atender todos os clientes;
    2. Verifica se o caixa está fechado;
    3. Se estiver fechado mostra um mensagem falando que o caixa já está fechado;
    4. Se não estiver fechado:

1. Usa `FecharCaixa` e armazena a fila do caixa fechado em `clientes`;
2. Sorteia os clientes de `clientes` para os outros caixas abertos de forma aleatória;
3. Cada cliente é cadastrado nos caixas sorteados usando `CaixaCadastrarCliente`.

OptionImprimirFilas

O método recebe um *array* com todos os caixas do sistema e segue a seguinte lógica:

1. Imprime a mensagem: `Clientes em Espera`;
2. Para cada caixa:
    1. Usa `CaixaImprimirFila` para imprimir a fila de clientes do caixa.

OptionImprimirStatusCaixas

O método recebe um *array* com todos os caixas do sistema e segue a seguinte lógica:

1. Imprime a mensagem: `Estados dos Caixas`;
2. Para cada caixa:
    1. Usa `ImprimirEstado` para imprimir o estado do caixa.

Compilação e Execução

O progama disponibilizado possui um arquivo Makefile que realiza todo o procedimento de compilação e execução. Para tanto, temos as seguintes diretrizes de execução:

Comando	**Função**
`make clean`	Apaga a última compilação realizada contida na pasta build
`make`	Executa a compilação do programa utilizando o gcc, e o resultado vai para a pasta build
`make all`	Executa os dois comandos anteriores de uma vez
`make run`	Executa o programa da pasta build após a realização da compilação