NSTITUTO FEDERAL DE MINAS GERAIS (IFMG)

***CAMPUS* SÃO JOÃO EVANGELISTA**

COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

Atividade AED I – Fila

Bianca de Castro Aguiar Fontes

SI 231

---

1. Inclusão das bibliotecas e criação das estruturas necessárias para o programa

```cpp
#include <iostream>
#include <windows.h>
#include <fstream>

using namespace std;

const int MAX_ORDERS = 100;

// estrutura de pedido
You, 20 hours ago | 1 author (You)
struct Order
{
    string customerName;
    string description;
    int tableNumb;
};

// estrutura da fila de pedidos
You, 1 hour ago | 1 author (You)
struct OrdersRow
{
    int end = -1;
    Order ordersList[MAX_ORDERS];
};

OrdersRow orders;
```

2. Cabeçalhos das funções utilizadas

```cpp
int menu();

void insertOrder(Order newOrder);
void markAsDone();
void showOrders();
void showNextOrder();

bool checkOrders();

Order readKeyboard();

void importOrders();
void exportOrders();
```

3. Função main, onde tem o Switch Case que gere todo o programa

```
int main()
{
    UINT CPAGE_UTF8 = 65001;
    UINT CPAGE_DEFAULT = GetConsoleOutputCP();
    SetConsoleOutputCP(CPAGE_UTF8);

    Order order;

    int option;
    do
    {
        option = menu();

        switch (option)
        {
        case 0:
            // sai do programa
            cout << "Saindo ... ";
            Sleep(2000);
            exit(0);

        case 1:
            // lê as informações do pedido
            order = readKeyboard();
            cout << "Anotando pedido ... \n";
            Sleep(2000);
            // insere no vetor
            insertOrder(order);
            break;      You, 20 hours ago • atividade_6 done

        case 2:
            // marca o primeiro pedido da fila como concluído
            markAsDone();
            break;

        case 3:
            showOrders();
            break;
```

```
        case 4:
            showNextOrder();
            break;

        case 5:
            exportOrders();
            break;

        case 6:
            importOrders();
            cout << "Anotando pedidos ... \n";
            system("pause");
            break;

        default:
            cout << "Opção inválida! Tente novamente ... ";
            Sleep(2000);
            break;
        }
        system("cls");
    } while (option != 0);

    return 0;
}
```

4. Exibe o menu

```cpp
// exibe menu para navegação no sistema e retorna a opção
int menu()
{
    int option;

    cout << "»———————————— MENU ————————————«\n\n";
    cout << "01 - Fazer pedido\n";
    cout << "02 - Marcar pedido como concluído\n";
    cout << "03 - Mostrar pedidos\n";
    cout << "04 - Mostrar próximo pedido\n";
    cout << "05 - Exportar pedidos\n";
    cout << "06 - Importar pedidos\n";
    cout << "00 - SAIR\n";
    cout << "\n\n»————————————————————————————————————————«\n\n";
    cout << "Digite: ";
    cin >> option;

    system("cls");

    return option;
}
```

5. Verifica se existem pedidos

```cpp
// verifica se existem pedidos na fila
bool checkOrders()
{
    // se o fim da fila por maior que (-1), ou seja, conteúdo existente
    if (orders.end > -1)
        return true;
    return false;
}
```

6. Insere um pedido manualmente do teclado

```cpp
Order readKeyboard()
{
    Order order;

    cout << "»———————————— ANOTANDO PEDIDO ————————————«\n\n";
    cin.ignore();
    cout << "Nome do cliente: ";
    getline(cin, order.customerName);
    cout << "Descrição do pedido: ";
    getline(cin, order.description);
    cout << "Número da mesa: ";
    cin >> order.tableNumb;

    return order;
}
```

7. Insere de fato o pedido no vetor

```cpp
// insere o pedido na fila
void insertOrder(Order newOrder)
{
    // se o fim da fila for menor que o tamanho máximo do vetor ele anota o pedido
    if (orders.end < MAX_ORDERS - 1)
    {
        orders.end++;
        // insere o novo pedido no fim da fila
        orders.ordersList[orders.end] = newOrder;
        cout << "Pedido anotado com sucesso!\n";
        system("pause");
    }
}
```

8. Marca o primeiro pedido como concluído

```cpp
void markAsDone()
{
    int option, table = orders.ordersList[0].tableNumb;

    // verifica se existem pedidos anotados
    if (checkOrders())
    {

        // verifica se realmente quer marcar como concluido
        cout << "Digite (1) para marcar o pedido da mesa " << table << " como concluído!";
        cin >> option;
        if (option == 1)
        {
            // reorganiza a fila sobrescrevendo os pedidos do início
            for (int i = 0; i <= orders.end; i++)
                orders.ordersList[i] = orders.ordersList[i + 1];
            orders.end--;
            cout << " Pedido da mesa " << table << " marcado como concluído!\n";
        }
    }
}
```

9. Mostra os pedidos pendentes (que não foram concluídos)

```cpp
void showOrders()
{
    cout << "\n»———————————— PEDIDOS ————————————«\n\n";
    for (int i = 0; i < orders.end; i++)
    {
        cout << "Nome do cliente: " << orders.ordersList[i].customerName << "\n";
        cout << "Descrição do pedido: " << orders.ordersList[i].description << "\n";
        cout << "Número da mesa: " << orders.ordersList[i].tableNumb << "\n\n";
    }
    cout << "\n»————————————————————————————————————————«\n\n";

    system("pause");
}
```

10. Mostra o próximo pedido (o segundo pedido da fila)

```cpp
void showNextOrder()
{
    cout << "\n»———————————— PRÓXIMO PEDIDO ————————————«\n\n";
    cout << "Nome do cliente: " << orders.ordersList[1].customerName << "\n";
    cout << "Descrição do pedido: " << orders.ordersList[1].description << "\n";
    cout << "Número da mesa: " << orders.ordersList[1].tableNumb << "\n\n";
    cout << "\n»————————————————————————————————————————«\n\n";

    system("pause");
}
```

11. Importa os pedidos de um arquivo

```
212  void importOrders()
213  {
214      ifstream read;
215
216      read.open("pedidos.txt");
217
218      if (read.fail())
219      {
220          cerr << "\aError: Não foi possível abrir o arquivo\n";
221          system("pause");
222          read.clear();
223      }
224      else
225      {
226          read >> orders.end;
227          for (int i = 0; i < orders.end; i++)
228          {
229              getline(read >> ::ws, orders.ordersList[i].customerName);
230              getline(read >> ::ws, orders.ordersList[i].description);
231              read >> orders.ordersList[i].tableNumb;
232              cout << "Pedido da mesa " << orders.ordersList[i].tableNumb << " foi anotado com sucesso!\n";
233          }
234          cout << "Importação realizada!\n";
235          system("pause");
236          read.close();
237      }
238  }
```

12. Exporta os pedidos para um arquivo

```
240  void exportOrders()
241  {
242      ofstream write;
243
244      write.open("pedidos.txt");
245
246      if (!write.is_open())
247          cerr << "\aError: Não foi possível abrir o arquivo\n";
248      else
249      {
250          write << orders.end + 1 << endl;
251          for (int i = 0; i <= orders.end; i++)
252          {
253              write << orders.ordersList[i].customerName << endl;
254              write << orders.ordersList[i].description << endl;
255              write << orders.ordersList[i].tableNumb << endl;
256              cout << "Pedido da mesa " << orders.ordersList[i].tableNumb << " foi exportado com sucesso!\n";
257          }
258          cout << "Exportação realizada!\n";
259          system("pause");
260          write.close();
261      }
262  }
```