

”

E-fólio A | Folha de resolução para E-fólio



UNIDADE CURRICULAR: Laboratório de Programação

CÓDIGO: 21178

DOCENTE: Vitor Rocio

A preencher pelo estudante

NOME: Luís Carlos Crispim Pereira

N.º DE ESTUDANTE: 2300163

CURSO: LEI – Licenciatura de Engenharia Informática

DATA DE ENTREGA: 02/05/24

TRABALHO / RESOLUÇÃO:

1 – Para a atividade formativa 2 escolhi enunciado A – Biblioteca

2.a) Módulos e Interface:

1. livro.h / livro.c:

1.1. Variáveis: Livro (estrutura de dados para representar um livro).

1.2. Funções Disponibilizadas: adicionarLivro, removerLivro, etc. (funções para todas as operações relacionadas com os livros).

2. emprestimo.h / emprestimo.c:

2.1. Variáveis: Empréstimo (estrutura de dados para representar um empréstimo).

2.2. Funções Disponibilizadas: registrarEmpréstimo, removerEmpréstimo, etc. (funções para todas as operações relacionadas a empréstimos).

3. pesquisa.h / pesquisa.c:

3.1. Funções Disponibilizadas: pesquisarLivro, pesquisarEmpréstimo, etc. (funções para realizar pesquisas).

4. relatorio.h / relatorio.c:

4.1. Funções Disponibilizadas: relatorioLivrosMaisEmprestados, relatorioLivrosNaoDevolvidos, etc. (funções para gerar relatórios).

5. time.h / time.c:

5.1. Funções Disponibilizadas: getDataAtual (função para obter a data atual).

2.b) Estruturas de Dados:

- Livro: Estrutura para representar informações de um livro, como título, autor, gênero, exemplares.
- Empréstimo: Estrutura para representar informações de um empréstimo, como título do livro, utilizador, dias de empréstimo, data devolução, devolvido.

2.c) Função main e Funcionalidade Global do Programa:

A função main é o ponto de entrada do programa onde a funcionalidade global do programa é gerir uma biblioteca, permitindo ao utilizador realizar várias operações relacionadas a livros e empréstimos. Ele oferece funcionalidades como adicionar, remover e editar livros, registar e devolver empréstimos, pesquisar livros e empréstimos, e gerar relatórios.

Para atender aos critérios de avaliação listados na secção de Efolio A optei por um Estrutura de Arquivos modular conforme explicado em 2.b. O código está dividido em vários módulos, cada um com responsabilidades bem definidas. Cada módulo contém um ficheiro de cabeçalho (.h) e um ficheiro de implementação (.c), seguindo as práticas padrão de programação em C. Dentro de cada módulo, as funções estão agrupadas logicamente de acordo com a sua funcionalidade, promovendo a coesão dentro dos módulos. Os ficheiros de cabeçalho contêm apenas as definições de estruturas de dados, protótipos de funções e macros necessários para a utilização dos módulos correspondentes.

Cada módulo foi projetado para ter alta coesão, ou seja, as funções dentro de cada módulo estão relacionadas e desempenham uma função semelhante. Para manter o acoplamento baixo, as dependências entre módulos são minimizadas. Por exemplo, a comunicação entre os módulos é feita principalmente através de passagem de parâmetros e chamadas de função, em vez de acesso direto a variáveis globais. O código foi escrito com reutilização em mente. Funções genéricas e modulares foram criadas para realizar tarefas comuns, permitindo que sejam facilmente reutilizadas em diferentes partes do programa. Além disso, foram evitadas duplicações de código, concentrando funcionalidades semelhantes em funções compartilhadas entre os módulos, promovendo assim a reutilização e a manutenção do código.

O tipo de codificação adotado no programa, como modularidade, encapsulamento e abstração funcional, promove a reutilização de código. Isso é evidente na estruturação dos módulos e na definição de funções genéricas que podem ser adaptadas e reutilizadas em diferentes contextos.

Testes de utilização em anexo.

Nota: Feito VSC em SO macOS, mas junto ficheiros EXEC de SO ubuntu e SO Windows, onde testei a compilação também do mesmo.

3. Para adaptar o código desenvolvido para o cenário de gestão de um restaurante, teria de repensar os seguintes pontos:

a) Módulos/Funções que podem ser reutilizados:

1. Funções de manipulação de arquivos CSV para leitura e escrita de dados dos pratos e pedidos (utilizando os existentes de Livros e empréstimos).
2. Funções de pesquisa e manipulação de dados, como adicionar, remover, editar e pesquisar pratos (no código biblioteca é livros) e pedidos (no código biblioteca é empréstimos)
3. Funções de gestão de memória dinâmica para alocar e liberar memória conforme necessário.
4. Funções de relatórios para analisar estatísticas sobre pratos mais pedidos e mesas com maior taxa de ocupação.

b) Código que pode ser adaptado:

1. A lógica para adicionar, remover, editar e pesquisar livros pode ser adaptada para lidar com os dados dos pratos do restaurante.
2. A estrutura de dados utilizada para representar os livros e os empréstimos pode ser ajustada conforme necessário para armazenar informações relevantes, como nome, descrição, preço, tipo de prato, estado do pedido, etc.
3. Funções relacionadas à manipulação de datas e cálculos podem ser adaptadas para calcular o total da conta com base nos pratos pedidos e para gerenciar a disponibilidade das mesas.
4. Funções relacionadas com os relatórios podem ser adaptadas para novos relatórios.

c) Módulos/Funções que precisariam ser desenvolvidos:

1. Módulos específicos para lidar com a gestão de mesas, atribuição de pedidos a mesas específicas e controle de disponibilidade de mesas.
2. Funções para registrar pedidos de clientes, controlar o estado dos pedidos e calcular o total da conta com base nos pratos pedidos.

Em resumo, enquanto algumas partes do código desenvolvido para a gestão de empréstimos de livros podem ser reutilizadas e adaptadas para atender aos requisitos da gestão de um restaurante, outras partes exigiriam desenvolvimento adicional para lidar com as operações específicas, como gestão de mesas, controle de pedidos e cálculo de contas.