

”

**E-fólio B** | Folha de resolução para E-fólio



## LABORATÓRIO DE PROGRAMAÇÃO | 21178

**Nome:** Carlos Miguel Faria Martins da Costa

**N.º de Estudante:** 2203524

**Curso:** Licenciatura em Engenharia Informática

**Docentes:** Nelson Russo/ Pedro Viegas Júnior

**Data:** 01.06.2024

**Ano Letivo:** 2023/24

## E1.Expliação da organizaão modular do seu código;

Este programa encontra-se organizado de maneira modular, o que significa que ele está dividido em várias partes, cada uma com uma responsabilidade específica. Isso ajuda a manter o código organizado e facilita a sua manutenção.

Cada um desses módulos tem um arquivo de cabeçalho correspondente (.h), que declara as funções e estruturas que o módulo fornece, permitindo que outros módulos usem essas funcionalidades.

Passo agora a explicar sucintamente os três módulos principais:

1. menu.c: Este módulo é responsável por interagir com o usuário. Ele exibe um menu e processa os comandos que o usuário digita. Quando o utilizador digita um comando como 'create' ou 'moveleft', este módulo verifica se o comando é válido e chama as funções apropriadas para realizar a ação. Este modulo é chamado pela função main.c;
2. rectangle.c: Neste módulo estão todas as funções que manipulam retângulos. Ele tem funções para criar, mover, fundir retângulos e aplicar gravidade, além de verificar se áreas específicas da tabela estão ocupadas.
3. list.c: Aqui temos as estruturas de dados e funções necessárias para gerenciar uma lista duplamente encadeada. Cada nó desta lista representa um retângulo, e este módulo permite-nos adicionar, remover e verificar nós na lista.

A utilização de uma lista duplamente encadeada neste programa é vantajosa principalmente devido à necessidade de frequentes inserções de retângulos, bem como à necessidade de navegação bidirecional para operações de fusão e movimentação. A eficiência e simplicidade proporcionadas por uma lista duplamente encadeada facilitam a implementação dessas operações, tornando o programa mais eficiente e o código mais limpo e fácil de manter.

## E2. Estruturas de Dados Usadas na Implementação

Para implementar a manipulação dos retângulos, utilizo duas estruturas de dados:

1. **Rectangle**: Esta estrutura representa um retângulo com quatro propriedades: a posição x e y do canto superior esquerdo, e a largura (l) e altura (h) do retângulo.

```
typedef struct {  
    int x;  
    int y;  
    int l;  
    int h;  
} Rectangle;
```

2. **Node:** Esta é uma estrutura de nó para uma lista duplamente encadeada, onde cada nó contém um retângulo. Além do retângulo, ele tem ponteiros para o próximo e o nó anterior na lista.

```
typedef struct Node {  
    Rectangle rect;  
    struct Node* next;  
    struct Node* prev;  
} Node;
```

Estas estruturas permitem gerir vários retângulos de forma eficiente, armazenando-os numa lista duplamente encadeada.

### E3. Descrição da Funcionalidade Global do Programa

O programa permite que se crie e manipule retângulos dentro duma área de trabalho.

Vamos ver o que ele pode fazer:

- Criação de Retângulos: Permite criar e inserir novos retângulos na lista, definidos por coordenadas e dimensões. (createRectangle(&rects, x, y, l, h));
- Movimentação: Possibilita mover retângulos para novas posições. (moveSide(rects, x, y, p));
- Aplicação de Gravidade: Aplica a gravidade após cada operação. (applyGravity(rects));
- Verificação de Sobreposição: Checa se dois retângulos se sobrepõem. (alertaSobreposicao(rects));
- Fusão: Combina dois ou mais retângulos em um único retângulo maior. (mergeCommand(&rects, x1, y1, x2, y2));
- Imprime os retângulos: printRectangle(rects): Imprime os retângulos atuais na lista;
- Navegação e Iteração: Facilita a navegação eficiente pela lista, tanto para frente quanto para trás.

### E4: Descrição de como modificou ou adaptou o código da Atividade Formativa 3 para acomodar as novas funcionalidades.

Quando comecei a Atividade Formativa 3 reconheço que comecei por utilizar um vetor fixo de estruturas (Rectangle), sem recorrer ao uso de memória dinâmica pois em muitos casos, operar com um vetor fixo de estruturas pode ser mais rápido do que operar com uma estrutura de dados dinâmica, especialmente em situações onde o tamanho máximo do vetor é conhecido e fixo.

Na Sessão Síncrona, o Professor explicou-nos a vantagem do uso de memória dinâmica neste exercício. Tenho noção que depender exclusivamente de vetores estáticos limita a

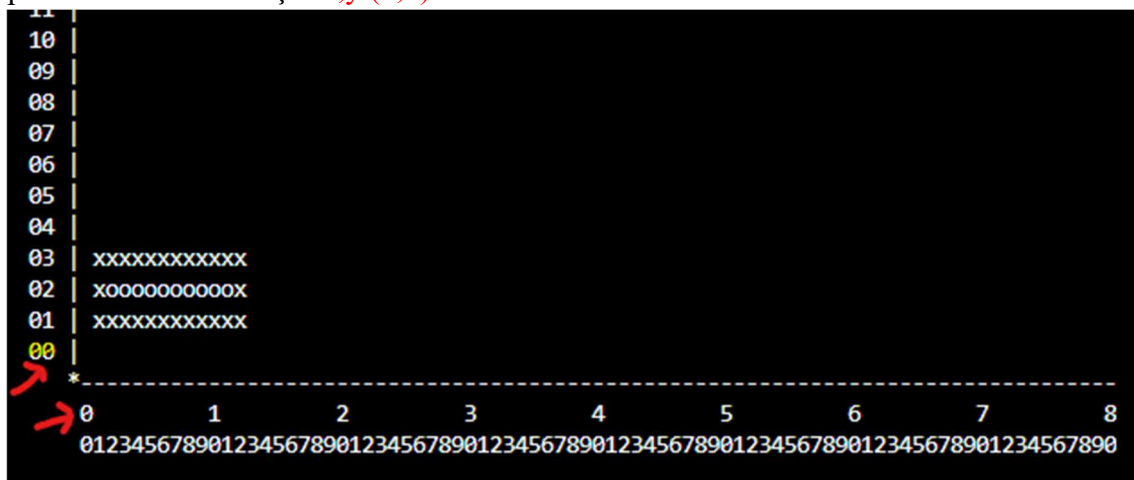
flexibilidade em termos de número de retângulos que podem ser manipulados, mas simplifica a gestão de memória, evitando problemas como vazamentos de memória. Recorrer a uma lista duplamente encadeada torna-se mais flexível e escalável, permitindo uma manipulação dinâmica de retângulos sem um limite fixo.

Por isso tive de refazer todo o código de forma a responder ao enunciado do efolio B.

Se posso fazer um breve resumo diria que este programa tem como objetivo a manipulação de retângulos num espaço 2D, oferecendo uma série de funcionalidades essenciais como criar, mover e fundir retângulos de forma eficiente e flexível.

### Notas:

1.O professor nos seus gráficos tinha colocado **x,y (1,1)**, por uma questão de estética pessoal resolvi começar **x,y (0,0)**.



2.Para aparecer o executável de todo o programa modular (com exceção aos testes) - **./main** – deve utilizar o comando : **gcc main.c list.c menu.c rectangle.c -o main**

Uma vez criado o executável será somente necessário colocar - **./main** – para executar todos os testes.

```
PS C:\Users\cmfmc\OneDrive\Bureau\progrma-modular-feito> gcc main.c list.c menu.c rectangle.c -o main
PS C:\Users\cmfmc\OneDrive\Bureau\progrma-modular-feito> ./main
```

3.Nos testes de Integridade e Unidade para obter o executável - **./a.exe** – utilizamos o comando: **gcc test.c list.c menu.c rectangle.c**

Uma vez criado o executável será somente necessário colocar - **./a.exe** – para executar todos os testes.

```
PS C:\Users\cmfmc\OneDrive\Bureau\progrma-modular-feito> gcc test.c list.c menu.c rectangle.c
PS C:\Users\cmfmc\OneDrive\Bureau\progrma-modular-feito> ./a.exe
```