



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: 02/06/24

TRABALHO / RESOLUÇÃO:

E1- Explicação da organização modular do meu código:

- 1) comando.h / comando.c:
 - a) Funções Disponibilizadas: runComando, alertaSobreposicao, verificaSobreposicao, etc... (funções para todas as operações relacionadas com os comandos inseridos).
- 2) plano.h / plano.c:
 - a) Funções Disponibilizadas: startPlano, printPlano. (funções para todas operações relacionadas com a impressão em consola do mundo dos retângulos.).
- 3) retangle.h / retangle.c:
 - a) Funções Disponibilizadas: createRectangle, freeRectangle, gravity. (funções para realizar sobre os retângulos).

E2: Estruturas de dados usadas na implementação:

- 1) Rectangle (Retângulo): Estrutura para representar as informações de um retângulo no plano, com os campos x (coordenada x), y (coordenada y) ,l (largura), h (altura).
- 2) Plano: Matriz de caracteres representando o plano, utilizada para impressão do mundo dos retângulos.

E3: Descrição da funcionalidade global do programa:

A função main é o ponto de entrada do programa onde a funcionalidade global do mesmo é a manipulação de um ambiente bidimensional composto por retângulos, representados num plano virtual. Ele oferece funcionalidades como Inserir comandos, ver mundo, reset ao mundo através de um Menu, para atender aos critérios esse menu está segmentado com a inserção de comandos em uma única entrada (podendo ser repetida n vezes), preferi esta abordagem, pois caso no futuro seja para essa mesma inserção de comandos separados será de fácil adaptação. Acrescentei a opção reset mundo (embora não pedido), para assim ser de fácil restart ao programa, sem ter de sair.

Para atender aos critérios da AF3 e do Efolio B optei por uma estrutura de arquivos modular conforme já utilizado anteriormente em Efolio A. 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.

E4: Descrição de como modifiquei ou adaptei o código da Atividade Formativa 3 para acomodar as novas funcionalidades:

a) Módulos/Funções que foram reutilizados:

- freeRectangle - Função para liberar a memória alocada para um retângulo.
- createRectangle - Função para criar um retângulo com as coordenadas e dimensões especificadas.

- gravity - Função para aplicar a gravidade a um retângulo.
- startPlano - Função para inicializar o plano com espaços vazios.
- moverRetangulo - Função para movimentar retângulo numa direcção.
- estaDentroDosLimites - Função para verificar se um retângulo com as dimensões especificadas está dentro dos limites do plano.

b) Código que pode ser adaptado:

- printPlano – teve de ser acrescentado o interior do retângulo 'O'.
- Acrescentar em todas as funções de comando e gravidade a mensagem de sucesso.
- runComando – teve de ser acrescentado merge e mergePossivel.

c) Módulos/Funções que precisaram de ser desenvolvidos:

- verificaSobreposicao - Função para verificar a sobreposição entre dois retângulos, utilizada em alertaSobreposicao, e em runComando.
- alertaSobreposicao - Função para alertar a sobreposição entre dois retângulos.
- buscarRetanguloPorCoordenadas - Função para buscar um retângulo com base em suas coordenadas.

Em resumo, enquanto algumas partes do código desenvolvido para a AF3 puderam ser reutilizadas e adaptadas para atender aos requisitos o Efolio B, outras partes exigiram desenvolvimento adicional para lidar com as operações específicas. Por opção em AF3 os comandos estão a ser processados por a função runComando, por opção não criei cada comando (create, moveleft, moveright, merge, mergePossivel) isoladamente, pois só são utilizados apenas na execução dos comandos, e assim o meu código manteve-se quase na sua totalidade para acomodar as exigências do Efolio B.

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

Nota2: Assumi moverigth 1,1+1 (na tabela do Efolio B) como moveright 1,1+1.

Nota3: Testes em Windows em computador de familiar, sendo que nos prints a palavra Retângulo não é reconhecido o acento circunflexo. Visto não ter ambiente Windows disponível com IDE para corrigir o mesmo, e como também é um “erro” de caracter baixo não achei necessário editar novamente todo o código para o efeito.