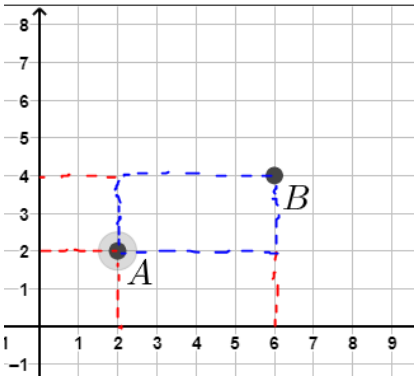
Lista de Exercícios 2

1 – Crie uma estrutura representando um aluno de uma disciplina. Essa estrutura deve conter o número de matricula do aluno, seu nome e as notas de três provas. Agora, escreva um programa que leia os dados de cinco alunos e os armazene-os em um bloco de memória alocado dinamicamente. Em seguida, exiba um relatório com o nome e as notas do aluno que possua a maior média geral dentre os cinco.

2 – Utilizando o conceito de Tipos Abstratos de Dados, crie uma estrutura chamada **Retângulo**, que deverá ficar encapsulada em um módulo aparte do módulo main(), sendo somente acessível através de seu endereço em memória. Essa estrutura deverá conter o **ponto** superior esquerdo e o **ponto** inferior direito do retângulo (2 estruturas **Ponto**). Cada coordenada é definida por uma estrutura **Ponto**, a qual conterá as posições X e Y. Faça um programa leia as coordenadas do retângulo informadas pelo usuário, e armazene-as em uma estrutura Retângulo. Em exiba a área, o comprimento da diagonal e o perímetro desse retângulo.



Ponto A(2, 2)

Ponto B(6, 4)

3 – Crie uma função recursiva que receba um número inteiro “n” e imprima todos os números naturais entre 0 e “n”, em ordem crescente.

4 – Crie uma função recursiva que receba por parâmetro dois valores inteiros *x* e *y*, calcule e retorne o resultado de x^y para o programa principal

5 – Crie uma estrutura chamada Cadastro. Essa estrutura deve conter o nome, a idade e o endereço de uma pessoa. Agora, escreva uma **função** que receba um inteiro “n” e retorne o ponteiro para um bloco de memória de tamanho “n” bytes, alocado dinamicamente, para ser utilizado como vetor desse tipo de estrutura. Solicite também

que o usuário digite os dados desse vetor dentro de uma outra função especializada em coletar dados.

6 – Utilizando o programa do exercício anterior, crie uma rotina (função), para salvar o conteúdo do vetor em um arquivo de forma binária, que é mais fácil de utilizar quando trabalhamos com vetores de estruturas. Este seu novo programa deve ser capaz de salvar todo o conteúdo gerado em uma seção de utilização, para que o mesmo conteúdo seja recuperado na próxima seção de utilização, ou seja, seu novo programa deve ser capaz de recuperar todos os dados gravados em seção anterior para continuar o processamento.

Dica 1: Em uma primeira compilação, seu arquivo não existirá, então comece supondo que o arquivo existe pois essa será a situação recorrente. Tente abri-lo em modo leitura, caso haja erro de abertura pois o arquivo não existe (1ª compilação), tente abri-lo em modo gravação e então execute o processamento normal do seu programa. Quando terminar, você terá o arquivo gerado e salvo, para a próxima execução do programa.

Dica 2: Quando você estiver lendo um arquivo que já existe, simplesmente seu programa não terá a informação de quantas estruturas estão armazenadas no arquivo. Para sanar esse problema aloque o vetor que receberá as estruturas lidas do arquivo, com apenas 1 elemento. Efetue a leitura de apenas 1 estrutura no arquivo e armazene no vetor. Em seguida tente ler novamente outra estrutura, caso consiga, realoque o vetor aumentando em mais uma posição apenas no vetor, e armazene-a no vetor. Lembrando que não sabemos quantos elementos temos no arquivo, então a lógica é se consegui ler mais uma, então aumento em mais um espaço no vetor e armazeno lá, depois tento ler mais uma novamente, e então o processo se repete.

Dica 3: Como controlar a leitura do arquivo, de forma automática até que chegue ao seu final e eu saiba quando não é mais necessário realocar o vetor?

Coloque a função de leitura do arquivo dentro da condição de avaliação de um laço while. Enquanto a função conseguir ler o arquivo, ela retornará a quantidade lida, que nesse caso será 1 unidade. Quando ela não conseguir ler, ela retornará 0 unidades lidas, e então a avaliação do laço será “false” e ele não se repetirá. Dentro do laço coloque todas as rotinas de realocação e atribuição ao vetor, conseguindo assim a leitura e o preenchimento automático do vetor.