Tipos Abstratos de Dados (TADs) / Modularização

1. Você deve alterar o TAD `ListaVetorInteiros` implementado durante a aula (slides sobre TADs disponíveis no PVANet Moodle) para que ele suporte duas novas operações relacionadas à remoção de elementos da lista de inteiros: `remover_primeiro` e `remover_ultimo`, e uma terceira operação `inverte` que modifica a ordem dos elementos da lista (ou seja [1,2,3] para [3,2,1]).

   Utilize a função `main` abaixo para testar suas funções:

```
int main() {
  ListaVetorInteiros l1;
  l1.inserir_elemento(7);
  l1.inserir_elemento(10);
  l1.inserir_elemento(5);
  l1.inserir_elemento(2);
  l1.inserir_elemento(1);

  l1.imprimir();
  // 7, 10, 5, 2, 1

  l1.remover_primeiro();
  l1.imprimir();
  // 10, 5, 2, 1

  l1.remover_ultimo();
  l1.imprimir();
  //10, 5, 2

  l1.inverte();
  l1.imprimir();
  //2, 5, 10

  return 0;
}
```

2. Implemente um TAD `Cubo` para representação do cubo apresentado na Figura abaixo.

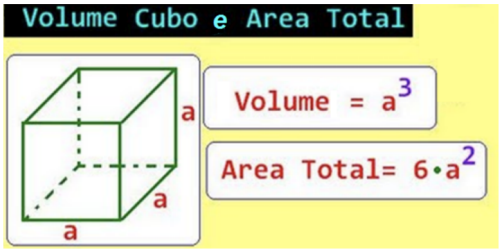


Figura 1: Representação de um cubo.

Você deve incluir um construtor, um destrutor e as operações que retornem o tamanhos de cada lado, a sua área e o seu volume. *Importante!* A alocação do cubo deve ser feita dinamicamente. E a função que retorna o tamanho de cada lado do cubo deve ser acionada a partir das funções para cálculo da área e volume do cubo.

Utilize a função `main` abaixo para testar suas funções:

```
int main() {
    float area, volume;
    Cubo *cubo = new Cubo(3.0);
    area = calculaAreaCubo(cubo);
    std::cout << area << std::endl;
    volume = calculaVolumeCubo(cubo);
    std::cout << volume << std::endl;
    delete cubo;
    return 0;
}
```

**Considerações Gerais!**

- Exercício individual.
- Entrega: conforme agendado no PVANET Moodle;
- Conforme estrutura abaixo apresentada crie um projeto para resolução de cada exercício (ex.: pratica4_exercicio1.zip, pratica4_exercicio2.zip). Cada projeto deve conter os arquivos `.h`, `.cpp`, e `main.cpp` criados para resolução do exercício. Envie, através do PVANet Moodle, uma pasta compactada (.rar ou .zip) contendo todos os projetos (também compactados). A pasta compactada deve conter informações do aluno (ex.: julio_reis-pratica4.zip).

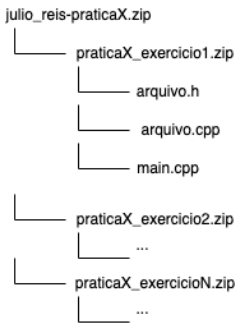


Figura 2: Estrutura de diretórios.

- O seu `main.cpp` deve conter, minimamente, instruções para criação (instanciação de objetos) e chamadas das funções implementadas (TODAS!!!). Para teste, você pode usar os exemplos fornecidos.