

Tipos Abstratos de Dados (TADs)

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:

```
1 int main() {
2     ListaVetorInteiros l1;
3     l1.inserir_elemento(7);
4     l1.inserir_elemento(10);
5     l1.inserir_elemento(5);
6     l1.inserir_elemento(2);
7     l1.inserir_elemento(2);
8
9     l1.imprimir();
10    // 7, 10, 5, 2, 1
11
12    l1.remover_primeiro();
13    l1.imprimir();
14    // 10, 5, 2, 1
15
16    l1.remover_ultimo();
17    l1.imprimir();
18    //10, 5, 2
19
20    l1.inverte();
21    l1.imprimir();
22    //2, 5, 10
23
24    return 0;
25 }
```

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

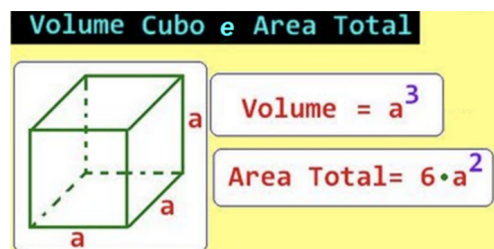


Figura 1: Representação de um cubo.

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:

```
1 int main() {
2     float area, volume;
3     Cubo *cubo = new Cubo(3.0);
4     area = calculaAreaCubo(cubo);
5     cout << area << endl;
6     volume = calculaVolumeCubo(cubo);
7     cout << volume << endl;
8     delete cubo;
9
10    return 0;
11 }
```

3. Implemente um TAD `Data` com três atributos inteiros: `dia`, `mês` e `ano`. Faça um construtor que inicializa as três variáveis e suponha que os valores passados serão sempre corretos. O TAD deve possuir um método para exibir a data em formato de números separados por barra: `dia/mes/ano` e outro método para exibir a data por extenso (ex: 28 de fevereiro de 2021). Por fim, você deve implementar um método `getDataEmSegundos` que recebe uma data como argumento e retorna sua representação em segundos (em caso de dúvidas, buscar por *unix timestamp* - vale lembrar que a data inicial considerada será 01/01/1970 e que para fins de simplificação não vamos considerar o horário no cálculo, o padrão será sempre 00:00:00).

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

```
1 int main() {
2     Data d1(13, 12, 2021);
3     d1.exibirDataFormatoNumero();
4     d1.exibirDataPorExtenso();
5     d1.getDataEmSegundos();
6
7     return 0;
8 }
```

Considerações Gerais!

- Exercício individual.
- Entrega: conforme agendado no PVANET Moodle;
- Conforme estrutura abaixo apresentada crie um projeto para resolução de cada exercício (ex.: `pratica3_exercicio1.zip`, `pratica3_exercicio2.zip`, etc). 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-pratica3.zip`).

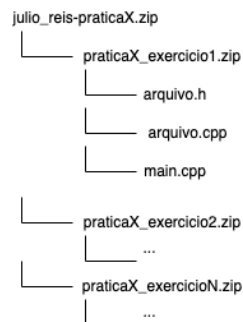


Figura 2: Estrutura de diretórios.

- 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.