CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO ORIENTADA A OBJETOS
PROF. JUDSON SANTIAGO

LABORATÓRIO 8

TIPOS ABSTRATOS DE DADOS

EXERCÍCIOS DE REVISÃO

VOCÊ DEVE RESPONDER PARA REVISAR OS CONCEITOS IMPORTANTES

1. Quais são as principais características de um TAD?

2. Como criar constantes para definir o tamanho de membros de uma classe?

3. Como criar constantes que serão usadas apenas dentro de métodos da classe?

EXERCÍCIOS DE FIXAÇÃO

VOCÊ DEVE FAZER OS EXERCÍCIOS PARA FIXAR O CONTEÚDO

1. Modifique a declaração da classe Pilha mostrada abaixo para que ela utilize um vetor dinâmico no lugar do vetor estático. O número de elementos deve ser passado no construtor da Pilha. Ela não deve ter um construtor padrão, ou seja, não deve ser possível criar uma pilha sem indicar a sua capacidade. Crie um destrutor para liberar a memória alocada dinamicamente.

   Crie um programa para testar a nova classe.

```
class Pilha
{
private:
    enum { MAX = 10 };
    Item itens[MAX];
    int topo;

public:
    Pilha();

    bool Vazia() const;
    bool Cheia() const;
    bool Empilhar(const Item & item);
    bool Desempilhar(Item & item);
};
```

   Desafio: modifique a classe pilha de forma que sua capacidade se expanda automaticamente sempre que a operação empilhar for realizada com uma pilha cheia. Faça também as seguintes alterações:

   - Remova o método `Cheia()`
   - Modifique o método `Empilhar()` para retornar void.
   - Adicione um construtor padrão que crie uma pilha vazia

EXERCÍCIOS DE APRENDIZAGEM

VOCÊ DEVE ESCREVER PROGRAMAS PARA REALMENTE APRENDER

1. Construa um programa para descobrir se uma palavra é um palíndromo. Uma forma de fazer isso é **empilhar** todas as letras da palavra e depois **desempilhar** uma a uma enquanto analisa se as letras desempilhadas são iguais as letras da palavra original.

```
Digite uma palavra: pilha
Empilhando e desempilhando fica: ahlip
A palavra não é um palíndromo.
```

2. Construa um programa que leia uma expressão matemática como uma sequência de caracteres (string) e verifique se os parênteses da expressão estão corretos, isto é, todo parêntese aberto deve ser fechado. **Utilize uma pilha** para resolver o problema.

```
Expressão: ((2+3)-5
[Erro] Parêntese não foi fechado

Expressão: (2+3)-5)
[Erro] Parêntese não foi aberto

Expressão: ((2+3)-(5*4))
[Ok] Parênteses corretos
```

3. A notação polonesa reversa é uma notação para expressões matemáticas que é superior a notação tradicional por não precisar de parênteses e por não ser ambígua. Na notação polonesa reversa os operandos veem sempre antes do operador.

   Notação tradicional: 2 * 3 – 4 / 5
   Notação parentizada: (2 * 3) – (4 / 5)
   Polonesa reversa: 23*45/-

   Considerando que os operandos são sempre números com apenas um dígito, construa um programa que receba uma expressão em notação polonesa reversa e calcule o seu resultado. **Use uma pilha** para resolver o problema.

```
Expressão: 23+
Resultado: 5

Expressão: 23+5-
Resultado: 0

Expressão: 54*42-*
Resultado: 40
```

4. Crie uma classe para representar uma lista de Itens.

    - A lista guarda zero ou mais itens de um tipo qualquer
    - A lista é criada vazia
    - Você pode adicionar itens na lista
    - Você pode verificar se a lista está vazia
    - Você pode verificar se a lista está cheia
    - Você pode visitar cada item da lista e realizar alguma ação sobre ele

    Forneça um arquivo Lista.h com a declaração de uma classe Lista e um arquivo Lista.cpp com a sua implementação. Crie também um pequeno programa para testar a classe.

    Os dados da lista podem ser armazenados em um vetor ou, se você conhecer o tipo de dado, através de uma lista ligada. Mas a interface pública não deve depender da sua escolha para o formato de armazenamento, ou seja, a interface pública não deve depender de índices nem de ponteiros.

    Implemente a visita através de um método que recebe por parâmetro um ponteiro para função:

```
void Lista::Visitar(void (*fn)(Item &));
```

    O ponteiro `fn` aponta para uma função externa a classe (não uma função membro). Ele recebe uma referência para um Item como argumento. Item é o tipo de elemento armazenado na lista. O método `Visitar()` aplica essa função a cada item da lista.

    A função `Exibir()` abaixo seria um exemplo de função que pode ser usada com o método Visitar. Crie outras funções externas que possam ser úteis e teste sua execução sobre os elementos da lista.

```
void Exibir(Item & i)
{
    cout << "[" << i << "] ";
}
```