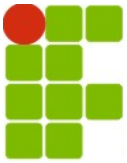


INSTITUTO FED. DE EDUCAÇÃO, CIÊNC. E TEC. DE PERNAMBUCO
CURSO: TEC. EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: ALGORITMOS E ESTRUTURAS DE DADOS
PROFESSOR: RAMIDE DANTAS
ASSUNTO: PILHAS, FILAS E LISTAS COM PONTEIROS

Prática 05

Parte 0: Preparação

Passo 1: Use o mesmo projeto da prática 4. **Faça um backup antes de continuar.**

Esta prática é uma continuação da prática 4 e faz uso dos mesmos códigos que acompanham aquela prática.

Parte 1: Implementando Pilhas com Ponteiros

Passo 1: Mude o nome do arquivo **pilha.h** feito na prática 4 para **pilha_array.h**.

Faça os ajustes necessários em outros arquivos.

Passo 2: Crie um arquivo chamado **pilha_ligada.h** e implemente nele a classe `Pilha` usando ponteiros e estruturas encadeadas, como descrito no material de aula.

OBS.: Leia o quadro abaixo antes de começar a implementação.

Passo 3: Modifique os arquivos **pilha_teste.cpp** e **polonesa.cpp** para incluir **pilha_ligada.h**.

Passo 4: Compile e teste seu código, verificando se os resultados são os esperados.

ATENÇÃO

- Por questão de compatibilidade, mantenha a mesma assinatura dos construtores como estavam na implementação com *arrays*. Isso vai forçar a `Pilha`, por exemplo, a ter uma capacidade artificial, mas isso é uma opção de implementação válida.
- Declare a classe Nó (*Node*) vista no material como uma classe (ou `struct`) interna e privada na classe da Pilha (isto é, uma classe aninhada). Isso evita que Nó seja usada fora da Pilha. Faça o mesmo nas outras estruturas (`Fila` e `Lista`).
- Objetos do tipo Nó são alocados uma vez (**`new`**), no momento que um novo elemento é inserido na estrutura, e devem ser desalocados (**`delete`**) quando o elemento é retirado.
- Ponteiros temporários usados para navegar a estrutura não precisam ser alocados (**`new`**).
- Tome o cuidado de nunca acessar o conteúdo de um ponteiro não inicializado (**`NULL`**) ou depois que ele já foi desalocado (**`delete`**).
- Lembre-se de modificar os destrutores para desalocar os elementos restantes na estrutura. A forma mais fácil é reutilizar o método **`desempilha()`** (ou equivalente na `Fila` e `Lista`) para remover os elementos até que a pilha (ou fila, lista) esteja vazia.

Parte 2: Implementando Filas com Ponteiros

Passo 1: Mude o nome do arquivo **fila.h** para **fila_array.h**.

Passo 2: Crie o arquivo **fila_ligada.h** e implemente a classe `Fila` utilizando ponteiros conforme descrito no material de aula.

Passo 3: Modifique **fila_teste.cpp** e **impressora.cpp** para incluir esse arquivo. Faça os ajustes necessários.

Passo 4: Compile e teste a aplicação, verificando se o resultado é o esperado.

Verifique se o comportamento está coerente com o observado na prática 4.

Parte 3: Implementando Listas com Ponteiros (Listas Ligadas ou Encadeadas)

Passo 1: Mude o nome do arquivo **lista.h** para **lista_array.h.**

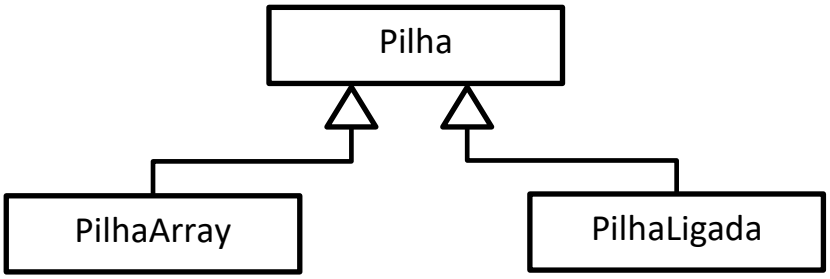
Passo 2: Crie o arquivo **lista_ligada.h** e implemente a classe `Lista` usando ponteiros conforme descrito no material de aula.

Passo 3: Modifique **lista_teste.cpp** e **lista_char.cpp** de forma a incluir esse arquivo. Faça os ajustes que forem necessários

Passo 4: Compile e teste a aplicação, verificando se o resultado é o esperado.

Parte 4: Refatore e Reorganize o Código

Passo 1: Refatore o código de forma que as duas implementações (com Array e Ponteiros) da classe Pilha sejam subclasses de uma classe abstrata Pilha, que define a interface dessas estruturas, conforme diagrama abaixo:



A classe abstrata deve ficar no arquivo **pilha.h,** que deve ser incluído em **pilha_array.h** e **pilha_ligada.h**. Renomeie as classes para `PilhaArray` e `PilhaLigada` e faça com que elas derivem de Pilha. Ao herdar usando templates (veja chamada do construtor):

```
template <class T> class Subclasse : public Superclasse<T> {
   ...
public:
       Subclasse (int param) : Superclasse<T> (param) { ... }
}
```

Passo 2: Repita os passos acima para as estruturas `Fila` e `Lista` e seus respectivos **.h** e **.cpp**.

Passo 3: Compile e rode os testes alternando entre estruturas ligadas e com *arrays*.