INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ

CAMPUS TERESINA-CENTRAL

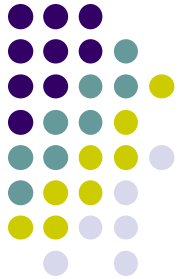
DIRETORIA DE ENSINO

Estrutura de Dados
Aula 4 – Pilhas e Filas

Professora: Elanne Cristina O. dos Santos

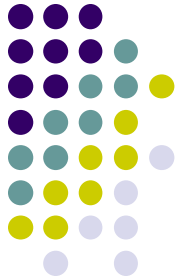
elannecristina.santos@gmail.com
elannecristina.santos@ifpi.edu.br

INTRODUÇÃO



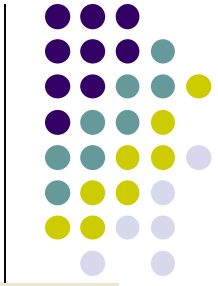
- O objetivo desta aula é o de conceituar as estruturas de dados: Pilhas (stack) e Filas (queue) abordando suas diferentes implementações e seus mecanismos de manipulação.

Pilha (stack) e Fila (queue) – Conceitos



- Uma ***pilha***, assim como uma ***fila***, é simplesmente uma lista linear de informações.
- Tanto a *pilha* como a *fila* podem ser implementadas por meio de uma lista encadeada ou de um vetor.
- O que difere uma ***pilha*** de uma ***fila*** é o mecanismo responsável pelo armazenamento e recuperação dos seus elementos.
- Enquanto a ***fila*** obedece ao princípio FIFO *(First In First Out),* uma ***pilha*** *(ou stack)* é manipulada pelo mecanismo LIFO *(Last In First Out)*.
- A analogia mais próxima que se faz de uma **pilha** é compará-la a uma pilha de pratos e a mais próxima de uma fila é a própria fila única de um banco ou supermercado.

Pilha

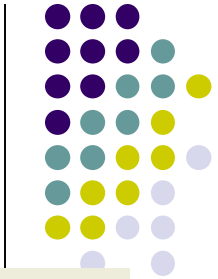


- Tipo Abstrato de dados com a seguinte característica:
- O **último** elemento a ser inserido é o **primeiro** a ser retirado/ removido

(**LIFO – Last in First Out**)

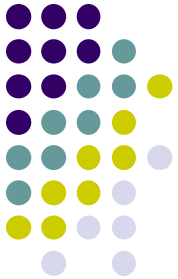
- Analogia: pilha de pratos, livros, etc.
- Usos: Chamada de subprogramas, avaliação de expressões aritméticas, etc.

Pilha



- Operações primárias:
  - PilhaPush(s, x) – Insere item **x** no **topo** da pilha **s**.
  - PilhaPop(s) – Retorna (e remove) o **topo** da pilha **s**.
  - PilhaInicializa(s) – Cria uma pilha vazia em **s**.
  - PilhaCheia(s), PilhaVazia(s) – Testa se a pilha **s** está cheia ou vazia, respectivamente.
- **Existem opções de estruturas de dados que podem ser usadas para representar pilhas.**
- As duas representações mais utilizadas são as implementações por meio de arranjos e de ponteiros

Pilha

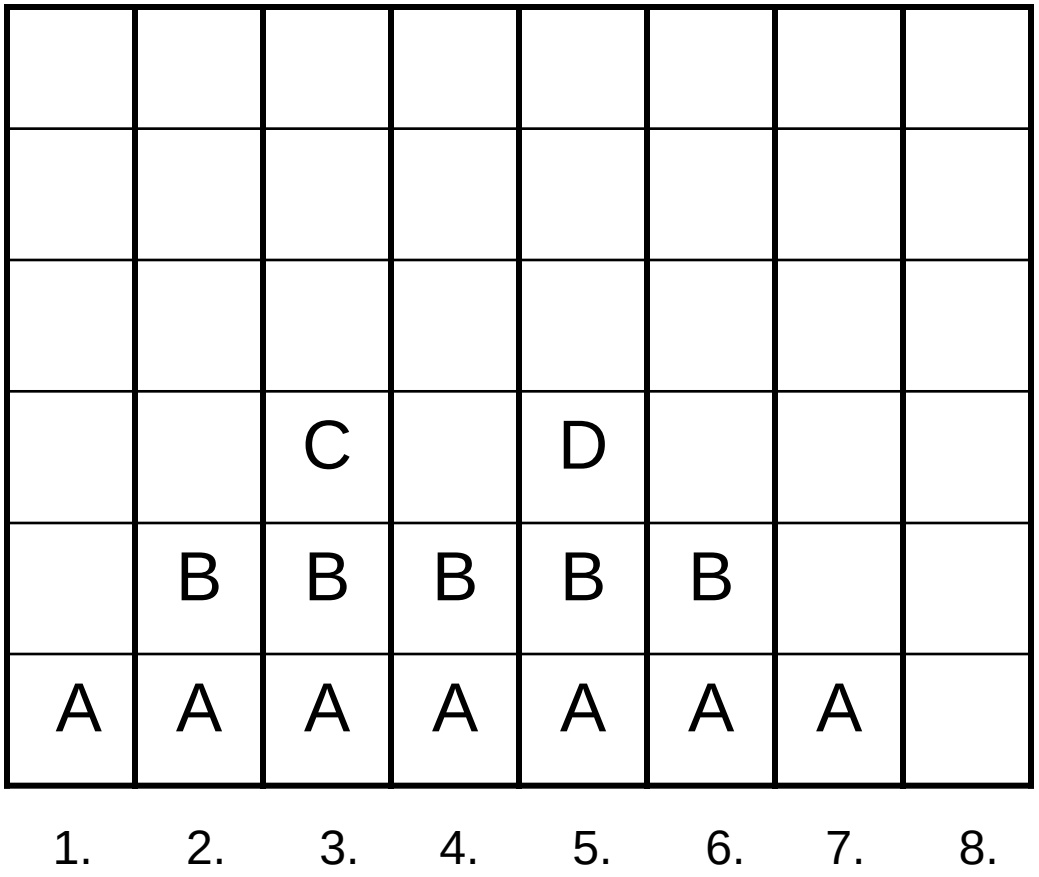


- O conceito de **pilha** é usado em muitos softwares de sistemas incluindo compiladores e interpretadores. (*A maioria dos compiladores C usa pilha quando passa argumentos para funções) .*
- As duas operações básicas – armazenar e recuperar – são implementadas por funções tradicionalmente chamadas de ***push e pop***, respectivamente).
- A função ***push()*** coloca um item na pilha e a função ***pop()*** recupera um item da pilha.
- A região de memória a ser utilizada como pilha pode ser um vetor, ou uma área alocada dinamicamente .

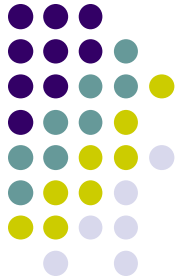
Pilha – Representação Gráfica

Ação:

1. push (A)
2. push (B)
3. push (C)
4. pop ( ) – recupera C
5. push (D)
6. pop ( ) – recupera D
7. pop ( ) – recupera B
8. pop ( ) – recupera A

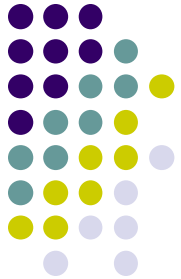


Interface do tipo Pilha



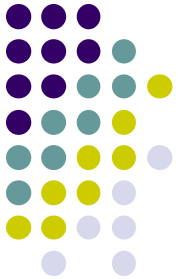
- Operações básicas:
  - criar uma pilha vazia
  - inserir um elemento (push)
  - retirar um elemento (pop)
  - verificar se a pilha está vazia
  - liberar a estrutura pilha
  - mostrar a pilha(*)

Filas - Conceitos



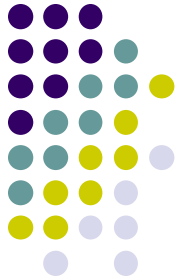
- A estrutura de fila é análoga ao conceito que temos de filas em geral. O primeiro a chegar é sempre o primeiro a sair, e a entrada de novos elementos sempre se dá no fim da fila.
- Em computação vemos este conceito sendo implementado em filas de impressão.
- Assim como as pilhas, uma fila também pode ser implementada por meio de um vetor ou de uma lista encadeada.

Interface do tipo Fila



- Operações básicas:
  - criar uma fila vazia
  - inserir um elemento no fim
  - retirar um elemento do início
  - verificar se a fila está vazia
  - liberar a estrutura fila
  - mostrar a fila (*)

Listas X Pilhas e Filas



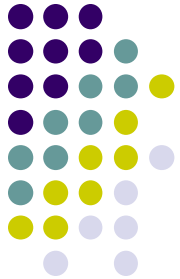
- Filas e pilhas têm regras bastante rigorosas para acessar dados, e as operações de recuperação têm caráter destrutivo .
- Listas normalmente são acessadas sequencialmente, pois cada item de uma lista contem além da informação um elo de ligação ao próximo item da cadeia, o que torna sua estrutura um pouco mais complexa.
- Além disso, a recuperação de um item da lista encadeada não causa a sua destruição. (*É preciso no caso de pilhas e filas uma operação de exclusão específica para esta finalidade) .*

Implementação usando Struct

1. interface pilha usando lista:

```
typedef struct no {
    float valor;
    struct no *prox;
 }No;

typedef struct pilha {
  No *topo;
}Pilha;
```

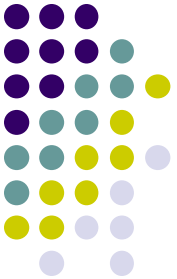
Implementação usando Struct



2. interface fila usando lista:

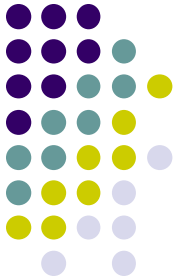
```
typedef struct no {
    float valor;
    struct no *prox;
 }No;

typedef struct fila {
  No *inicio;
  No *fim;
}Fila;
```

Implementação usando classes



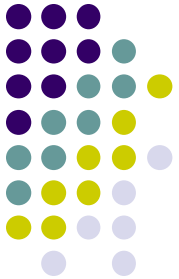
```
class No{
        public:
                char nome;
                No *prox;
                No(char n){
                        nome = n;
                        prox = NULL;}   };
class Fila{
        public:
                No *inicio;
                No *fim;
                Fila(){
                        inicio = NULL;
                        fim = NULL;             }  ........
```

Implementação usando classes



```
class No{
        public:
                char nome;
                No *prox;
                No(char n){
                        nome = n;
                        prox = NULL;                    }   };
class Pilha{
        public:
                No *topo;
                Pilha(){
                        topo = NULL;
                }.............
```

ATIVIDADE –operações básicas filas e pilhas



1. Inserir um elemento na fila.
2. Apagar um elemento da fila.
3. verificar se a fila está vazia.
4. Apagar todos os elementos da fila.
5. inserir em uma pilha.
6. Apagar em uma pilha.
7. verificar se a pilha está vazia.
8. apagar todos os elementos da pilha.

Obrigada pela atenção!!!
Boa semana de estudo pra vcs!!