Algoritmos e Estruturas de Dados I

Profª Priscilla Abreu

priscilla.abreu@ime.uerj.br

2022.1

Algoritmos e Estruturas de Dados I

Roteiro da aula

- Revisão – P1

Revisando...

Algoritmos e Estruturas de Dados I

Listas

O QUE É UMA LISTA?

Consideraremos como listas conjuntos sem repetições!

Uma lista é um conjunto de dados relacionados, e de número variável de elementos.

- Exemplo:
  - Lista de alunos de uma turma;
  - Lista de aprovados em um concurso;
  - Lista de produtos de uma loja;
  - ...

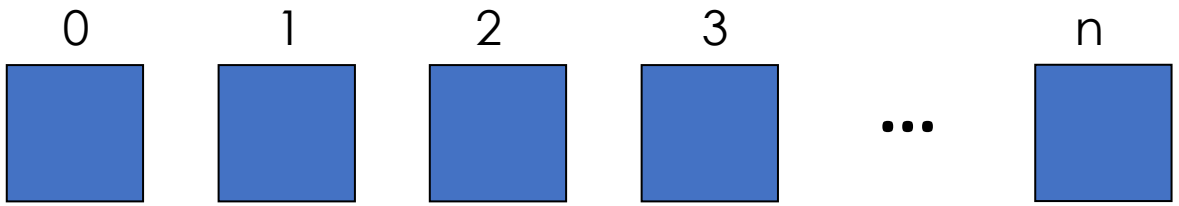
Cada elemento da lista terá uma chave – identificador único.

Algoritmos e Estruturas de Dados I

Listas Linear

Estrutura que permite representar um conjunto de dados de forma a preservar a relação de ordem existente entre eles.

Temos um primeiro elemento, segundo elemento, ..., n-ésimo elemento.



Algoritmos e Estruturas de Dados I

Listas Linear

Listas lineares

Listas lineares gerais
SEM restrição de inserção e remoção de elementos

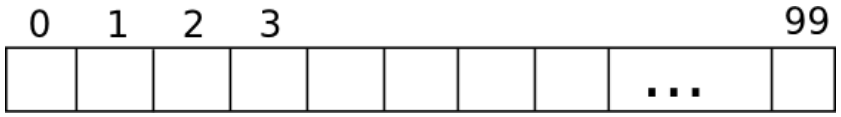
Listas particulares
COM restrição de inserção e remoção de elementos

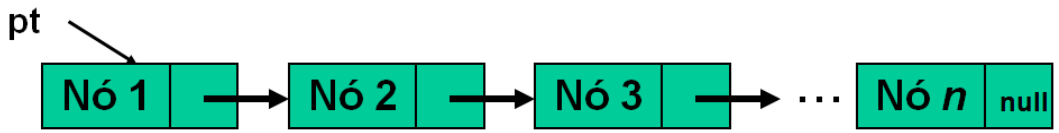
Algoritmos e Estruturas de Dados I

O tipo de armazenamento de uma lista linear pode ser classificado de acordo com a posição relativa na memória (contínua ou não) de cada dois nós consecutivos na lista.

Existem dois tipos de alocação:

- Alocação sequencial
- Alocação encadeada

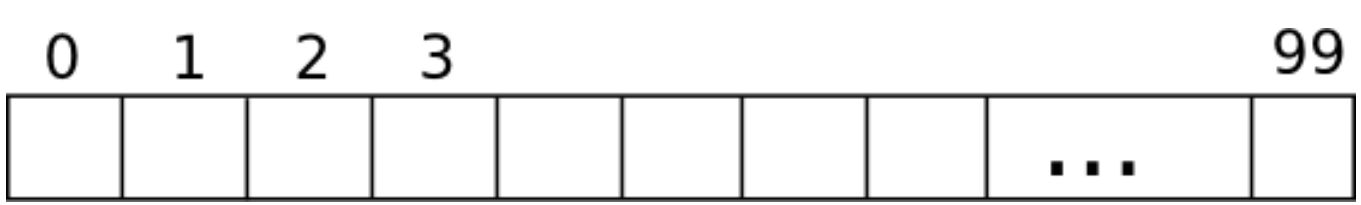




Algoritmos e Estruturas de Dados I

Lista Linear Sequencial

- Uso de vetores.



  - MAX é a quantidade máxima de elementos que a lista poderá armazenar.
  - **n representa o número de elementos.**

Algoritmos e Estruturas de Dados I

Lista Linear Sequencial

estrutura no:
    chave: <inteiro>

estrutura listaSeq:
    valores [1..MAX]: no
    n: inteiro

```
const int MAX = 10
typedef struct no{
    int chave;
}no;

typedef struct listaSeq{
    no valores[MAX];
    int n;
} listaSeq;
```

Algoritmos e Estruturas de Dados I

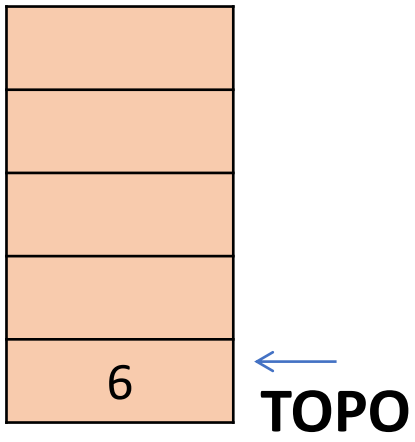
Pilha

- Estruturas de dados do tipo LIFO (last-in first-out): o último elemento a ser inserido, será o primeiro a ser removido.
- Manipulação dos elementos em apenas uma das extremidades: **topo**.
- Exemplos: pilha de pratos, pilha de livros, pilha de cartas de um baralho, etc.
    - Inserção: Empilha() ou Push()
    - Remoção: Desempilha() ou Pop()

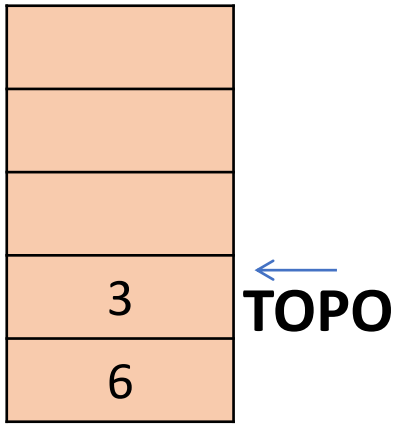
Algoritmos e Estruturas de Dados I

Pilha

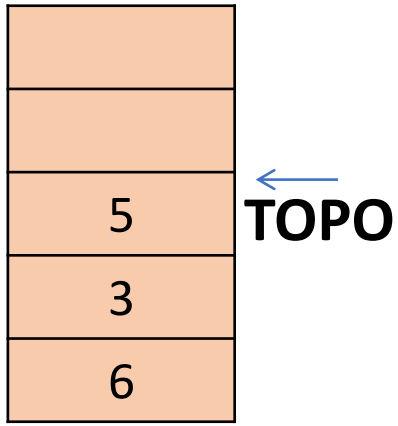
EMPILHA (6)



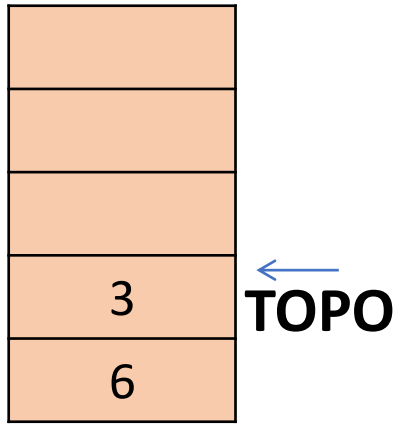
EMPILHA (3)



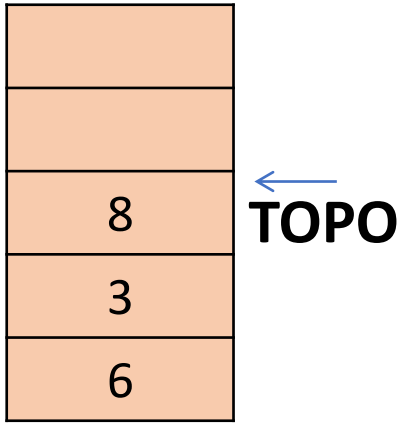
EMPILHA (5)



DESEMPILHA ()



EMPILHA (8)



Algoritmos e Estruturas de Dados I

Pilha – implementação

estrutura pilha:
    valores [1..MAX]: inteiro
    topo: inteiro

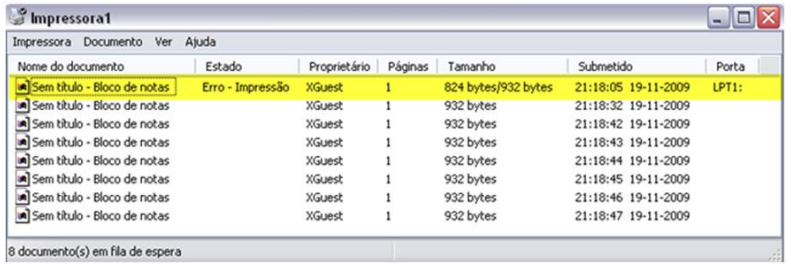
```
const int MAX = 10;

typedef struct pilhaSeq{
    int valores[MAX];
    int topo;
} pilhaSeq;
```

Algoritmos e Estruturas de Dados I

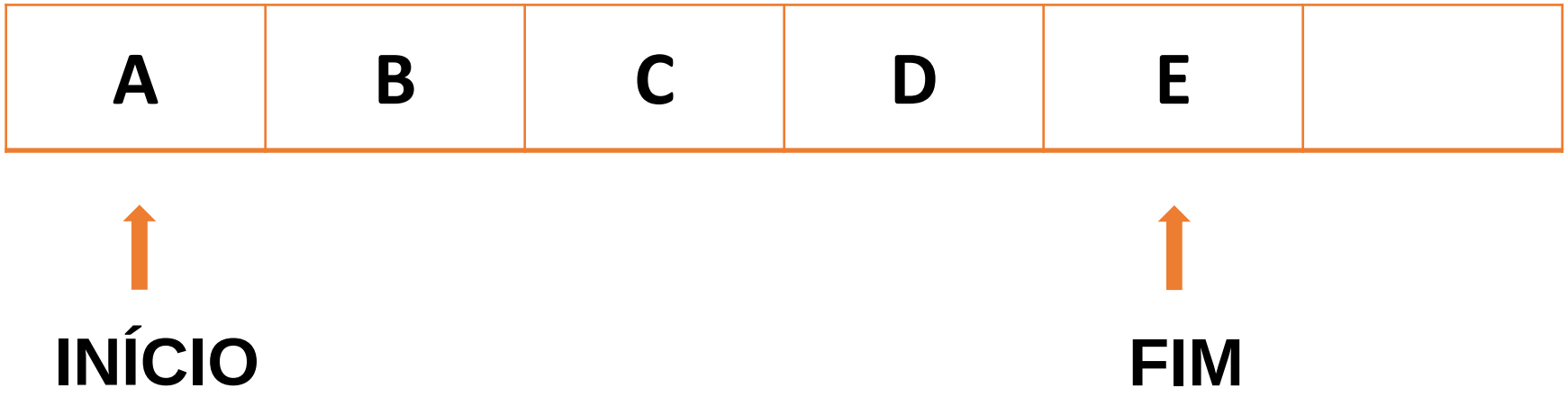
Fila

- São listas em que todas as inserções ocorrem em uma extremidade e as remoções em outra extremidade;
- Estruturas de dados do tipo FIFO (first-in first-out): o primeiro elemento a ser inserido, será o primeiro a ser removido.
- Exemplos: filas de banco, supermercado, fila de impressão de arquivos , etc.



Algoritmos e Estruturas de Dados I

Fila

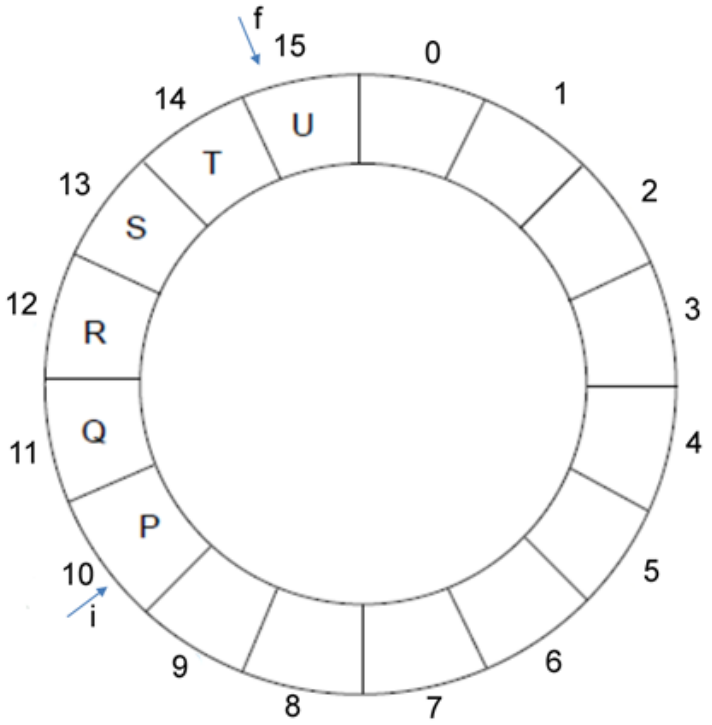


Início da fila: extremidade onde ocorrem as remoções.

Final da fila: extremidade onde ocorrem as inserções.

Algoritmos e Estruturas de Dados I

Fila – Uso da fila circular!



Algoritmos e Estruturas de Dados I

Fila – implementação

estrutura fila:
    valores [1..M]: inteiro
    i, f, n: inteiro

```
const int M = 10;

typedef struct filaSeq{
    int valores[MAX];
    int i, f, n;
} filaSeq;
```

Algoritmos e Estruturas de Dados I

Exercício

Faça um programa que implemente uma fila sequencial. O usuário deverá informar números inteiros, que serão inseridos na fila. Ao final da leitura de dados, o programa deverá inverter a ordem dos elementos na fila, isto é, os últimos serão os primeiros!
Você pode utilizar estruturas auxiliares, dentre as que já foram vistas. Decida que estrutura poderá auxiliar nessa tarefa!

Algoritmos e Estruturas de Dados I

Exercício – correção

```
#include <stdio.h>
#define MAX 10

typedef struct pilhaSeq{
        int valores[MAX];
        int topo;
}pilhaSeq;

typedef struct filaSeq{
        int valores[MAX];
        int i,f,n;
}filaSeq;
```

Algoritmos e Estruturas de Dados I

```
//pilha
void inicializa_pilha(int *topo){
		*topo = -1;
}
int pilha_cheia(int topo){
		return(topo==MAX-1);
}
int pilha_vazia(int topo){
		return(topo==-1);
}
int mostra_valor_topo(pilhaSeq *pilha){
		if(!pilha_vazia(pilha->topo)){
				printf("\nTopo da pilha: %d\n",pilha->valores[pilha->topo]);
		}
		else{
				printf("\nPilha vazia.\n");
		}
}
```

Algoritmos e Estruturas de Dados I


```
void empilha(pilhaSeq *pilha, int valor){
        if(!pilha_cheia(pilha->topo)){
                (pilha->topo)++;
                pilha->valores[pilha->topo]=valor;
                //printf("\nValor empilhado!\n");
        }
        else{
                printf("\nPilha cheia!\n");
        }

}
```

Algoritmos e Estruturas de Dados I


```
void desempilha(pilhaSeq *pilha, int *valor){
        if(!pilha_vazia(pilha->topo)){
                //printf("\nValor %d dempilhado!\n",pilha->valores[pilha->topo].chave);
                *valor = pilha->valores[pilha->topo];
                (pilha->topo)--;
        }
        else{
                printf("\nPilha vazia!\n");
        }

}
```

Algoritmos e Estruturas de Dados I


```
//fila
void inicializa_fila(int *i, int *f, int *n){
        *i = -1;
        *f = -1;
        *n = 0;
}

int fila_cheia(int n){
        if(n==MAX)
                return 1;
        return 0;
}
```

Algoritmos e Estruturas de Dados I

```
int fila_vazia(int n){
        if(n==0)
                return 1;
        return 0;
}

void mostra_proximo(filaSeq *fila){
        if(!fila_vazia(fila->n)){
                printf("\nPróximo da fila: %d\n",fila->valores[fila->i]);
        }
        else{
                printf("\nFila vazia.\n");
        }
}
```

Algoritmos e Estruturas de Dados I


```
void mostra_ultimo(filaSeq *fila){
        if(!fila_vazia(fila->n)){
                printf("\nÚltimo da fila: %d\n",fila->valores[fila->f]);
        }
        else{
                printf("\nFila vazia.\n");
        }
}
```

Algoritmos e Estruturas de Dados I


```
void enfileira(filaSeq *fila, int valor){
        if(!fila_cheia(fila->n)){
                if(fila->f==MAX-1){
                        fila->f=0;
                }
                else{
                        (fila->f)++;
                }
                if(fila->i==-1)
                        fila->i=0;
                fila->valores[fila->f]=valor;
                (fila->n)++;
                //printf("\nValor enfileirado!\n");
        }
        else{
                printf("\nFila cheia!\n");
        }

}
```

Algoritmos e Estruturas de Dados I


```
void desenfileira(filaSeq *fila, int *valor){
        if(!fila_vazia(fila->n)){
                //printf("\nValor %d denfileirado!\n",fila->valores[fila->i]);
                *valor = fila->valores[fila->i];
                if(fila->i==MAX-1){
                        fila->i=0;
                }
                else{
                        (fila->i)++;
                }
                (fila->n)--;
                if(fila->n==0)
                {
                        fila->i=-1;
                        fila->f=-1;
                }
        }
        else{
                printf("\nFila vazia!\n");
        }
}
```

Algoritmos e Estruturas de Dados I

```
int main(){
        filaSeq fila;
        pilhaSeq pilha;
        int valor,resp;
        inicializa_fila(&fila.i,&fila.f,&fila.n);
        inicializa_pilha(&pilha.topo);
        do{
                printf("Informe um valor: ");
                scanf("%d",&valor);
                enfileira(&fila,valor);
                printf("\nDeseja enfileirar um novo valor (1-sim ou 2-não)?");
                scanf("%d",&resp);
        }while(resp==1 && !fila_cheia(fila.n));
```

Algoritmos e Estruturas de Dados I

```
while(!fila_vazia(fila.n)){
        //desenfileirar
        desenfileira(&fila,&valor);
        //empilhar
        empilha(&pilha,valor);
}

while(!pilha_vazia(pilha.topo)){
        //desempilhar
        desempilha(&pilha,&valor);
        //enfileirar
        enfileira(&fila,valor);
}
```

Algoritmos e Estruturas de Dados I


```
        mostra_proximo(&fila);
        mostra_ultimo(&fila);

}
```

FIM