

# Algoritmos e Programação Estruturada

## Estrutura de Dados

Ma. Vanessa Matias Leite

1

- Unidade de Ensino: 04
- Competência da Unidade: Conhecer e compreender as listas ligadas, sua construção e uso adequados, e sua aplicação em programas de
- Resumo: Estudo de desenvolvimento de um Programa de Computador para Cálculo de Fatoriais com números
- Palavras-chave: lista; pilha; fila; estrutura de dados;
- Título da Teleaula: Estrutura de dados
- Teleaula nº: 04

2

## Listas

3

### Struct

Variável que armazena valores de tipos diferentes;

```
#include<stdio.h>

struct automovel{
    char modelo[20];
    int ano;
    float valor;
};

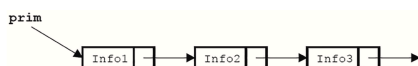
main() {
    struct automovel dadosAutomovel1;
}
```

Fonte: Scheffer (2018)

4

### Lista Ligada

- Estrutura de dados linear e dinâmica;
- Cada elemento da sequência é armazenado em uma célula da lista;



Fonte: Ricard (2018)

5

### Lista Ligada

```
struct lista {
    int info;
    struct lista* prox;
};
typedef struct lista Lista;

struct alunos {
    char nome[25];
    struct alunos* prox;
};
typedef struct alunos Classe;
```

6

### Lista Ligada

- Criação ou definição da estrutura de uma lista.
- Inicialização da lista.
- Inserção com base em um endereço como referência.
- Alocação de um endereço de nó para inserção na lista.
- Remoção do nó com base em um endereço como referência.
- Deslocamento do nó removido da lista.

7

### Lista Ligada

```

Lista* inicializa (void)
{
    return NULL;
}

int main() {
    Lista* listaFinal;
    listaFinal = inicializar();
    listaFinal = inserir(listaFinal, 13);
    listaFinal = inserir(listaFinal, 56);
}

```

8

## Operações com Listas Ligadas - Inserção

9

### Adicionar elementos na lista

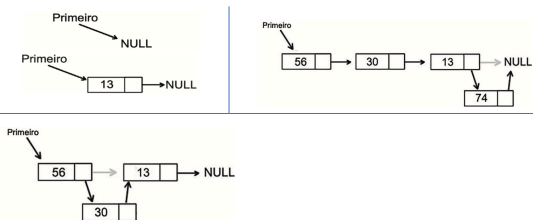
- Para inserirmos um elemento na lista ligada, é necessário alocarmos o espaço na memória;
- Atualizar o valor do ponteiro;

#### Posição do inserção

- Final da lista;
- Primeira posição;
- No meio da lista;

10

### Adicionar elementos na lista



11

### Adicionar elementos na lista

```

Lista* inserir (Lista* l, int i) {
    Lista* novo = (Lista*) malloc(sizeof(Lista));
    novo -> info = i;
    novo -> prox = l;
    return novo;
}

```

12

### Adicionar elementos na lista

```

Lista* inserirPosicao(Lista* l, int pos, int v){
    int cont = 1;
    Lista *p = l;
    Lista* novo = (Lista*)malloc(sizeof(Lista));
    while (cont != pos){
        p = p->prox;
        cont++;
    }
    novo->info = v;
    novo->prox = p->prox;
    p->prox = novo;
    return l;
}

```

13

```

Lista* inserirFim(Lista* l, int v){
    Lista *p = l;
    Lista* novo = (Lista*)malloc(sizeof(Lista));
    while (p->prox != NULL){
        p = p->prox;
        cont++;
    }
    novo->info = v;
    novo->prox = p->prox;
    p->prox = novo;
    return l;
}

```

14

## Operações com Listas Ligadas

15

### Remover Elementos da Lista

Primeiro elemento da lista:



Elemento no meio da lista:



Fonte: Ricardo (2018)

16

### Adicionar elementos na lista

```

Lista* remove (Lista* l, int v) {
    Lista* anterior = NULL;
    Lista* p = l;
    while (p != NULL && p->info != v) {
        anterior = p;
        p = p->prox;
    }
    if (p == NULL)
        return l;
    if (anterior == NULL) {
        l = p->prox;
    } else {
        anterior->prox = p->prox;
    }
    return l;
}

```

17

### Outras operações na lista Ligada

- Percorrer a lista ligada;
  - Saber quais elementos fazem parte da estrutura de dados;
- Verificar se um elemento se encontra na lista ligada

18

### Outras operações na lista Ligada

```
void imprimir (Lista* l) {
    Lista* p;
    printf("Elementos:\n");
    for (p = l; p != NULL; p = p->prox) {
        printf("%d -> ", p->info);
    }
}
```

19

```
Lista* buscar(Lista* l, int v){
    Lista* p;
    for (p = l; p != NULL; p = p->prox) {
        if (p->info == v)
            return p;
    }
    return NULL;
}
```

20

## Relatório

21

## Exercício

22

- ( ) Só é possível retirar um elemento da lista do seu final;
- ( ) Para inserir e retirar um elemento da lista deve-se atualizar o valor do ponteiro;
- ( ) A struct é uma variável que armazena valores de tipos diferentes

23

## Pilha

24

## Pilha

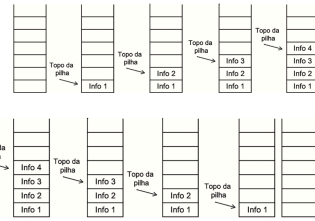
São estruturas de dados do tipo LIFO (*last-in first-out*), onde o último elemento a ser inserido, será o primeiro a ser retirado.



Fonte: Ricard (2018)

25

## Pilha



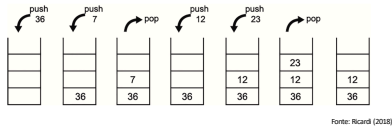
Fonte: Ricard (2018)

26

## Pilha

Duas operações básicas:

- Empilhar um elemento (push())
- Desempilhar um elemento (pop())



Fonte: Ricard (2018)

27

## Pilha

```
struct Pilha {
    int topo;
    int capacidade;
    float * proxElem;
};

void cria_pilha(struct Pilha *p, int c){
    p -> proxElem = (float*) malloc (c * sizeof(float));
    p -> topo = -1;
    p -> capacidade = c;
}

struct Pilha minhaPilha;
```

28

## Pilha

```
void push_pilha(struct Pilha *p, float v){
    p -> topo++;
    p -> proxElem [p -> topo] = v;
}
```

29

## Pilha

```
float pop_pilha (struct Pilha *p){
    float aux = p -> proxElem [p -> topo];
    p -> topo--;
    return aux;
}
```

30

## Fila

31

## Fila

São estruturas de dados do tipo FIFO (*first-in first-out*), onde o primeiro elemento a ser inserido, será o primeiro a ser retirado, ou seja, adiciona-se itens no fim e remove-se do início.

32

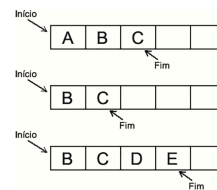
## Fila

Passos para a criação de uma Fila:

- criar uma fila vazia;
- inserir elemento no final;
- retirar um elemento do início;
- verificar se a fila está vazia;

33

## Fila



Fonte: Ricard (2018)

34

## Fila

```
#define N 100

struct fila {
    int n;
    int ini;
    char vet[N];
};

typedef struct fila Fila;

Fila* inicia_fila (void){
    Fila* f = (Fila*) malloc(sizeof(Fila));
    f->n = 0;
    f->ini = 0;
    return f;
}
```

35

## Fila

```
void insere_fila (Fila* f, char elem){
    int fim;
    if (f->n == N){
        printf("A fila está cheia.\n");
        exit(1);
    }

    fim = (f->ini + f->n) % N;
    f->vet[fim] = elem;
    f->n++;
}
```

36

**Fila**

```
float remove_fila (Fila* f){
    char elem;
    if (fila_vazia(f)){
        printf("A Fila esta vazia\n");
        exit(1);
    }
    elem = f->vet[f->ini];
    f->ini = (f->ini + 1) % N;
    f->n--;
    return elem;
}
```

37

**Lista**

38

**Implementação da  
Fila**

39

**Exercício**

40

- ( ) Uma pilha é uma estrutura de dados do tipo LIFO (, onde o primeiro elemento a ser inserido, será o primeiro a ser retirado.
- ( ) As filas são estruturas de dados do tipo FIFO (*first-in first-out*), adiciona-se itens no fim e remove-se do início.
- ( ) As principais funções da fila são push e pop;

41

**Recapitulando**

42

### Recapitulando

- Lista;
- Pilha;
- Fila;

43



44