

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

SUL DE MINAS GERAIS
Campus Muzambinho

Algoritmo e Estruturas de Dados II

*Lista Simplesmente Encadeada por
Alocação Dinâmica de Memória*

IFSULDEMINAS, *campus* Muzambinho

Curso de Bacharelado em Ciência da Computação

AED II – Algoritmo e Estruturas de Dados II



Estrutura de Dados Lista Ligada (*linked list*)

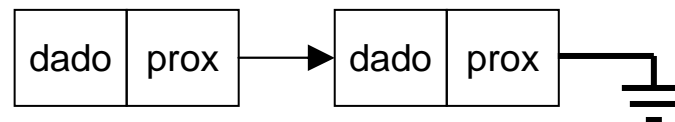
Uma Lista Encadeada (ou *lista linear*) é um conjunto linear de estruturas auto-referenciadas, chamadas nós, concatenadas por *links* (*ligações* ou *encadeamento*) de ponteiros.

Vantagens sobre Arrays

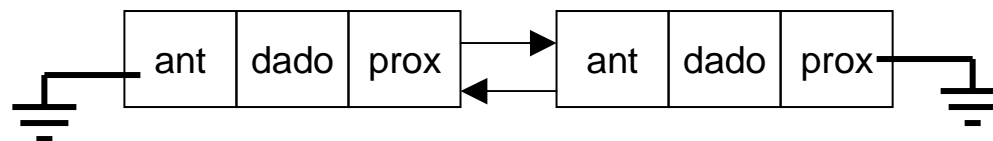
- número de elementos não previstos (alocação dinâmica)
- inserção e remoção de elementos entre outros elementos

Listas

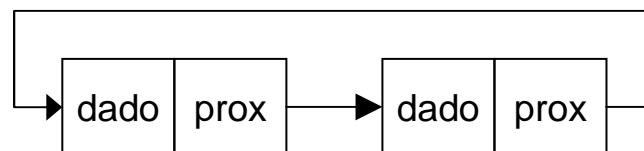
Lineares Simplesmente Encadeadas



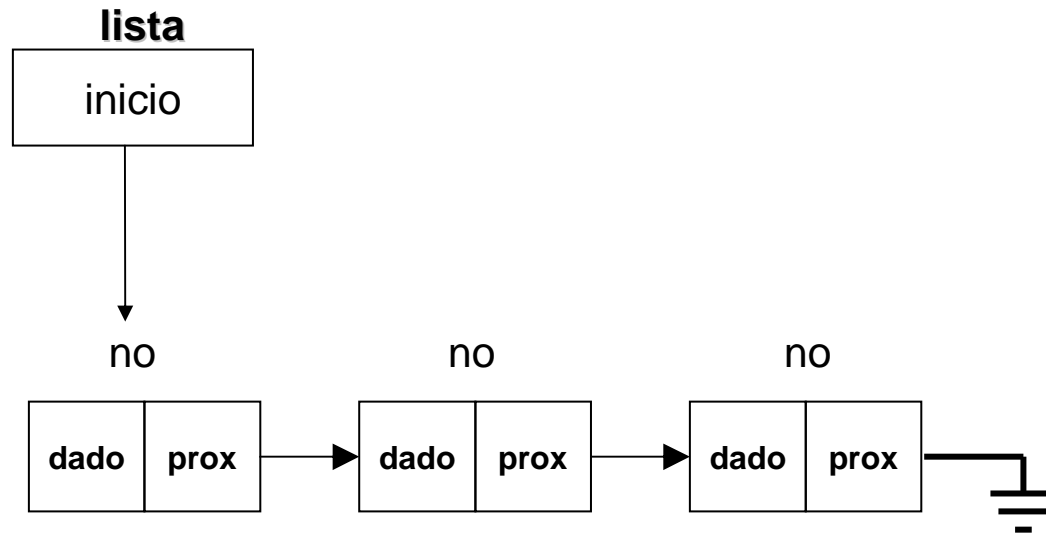
Lineares Duplamente Encadeadas



Circulares



TAD - Lista Simplesmente Encadeada



```
#include <stdio.h>


#define TRUE 1
#define FALSE 0

struct no
{
    int dado;
    struct no *prox;
};

typedef struct
{
    struct no *inicio;
} lista;
```

```
void create(lista *q);
int isEmpty(lista q);
int insert(lista *q, int d);
int remover(lista *q, int d);
void imprime(lista q);
```

TAD - Lista Simplesmente Encadeada




```
#include <stdio.h>
main( )
{
    lista L;
    create(&L);
    insert(&L,12);
    insert(&L,320);
    insert(&L,197);
    insert(&L,26);
    imprime(L);
    remover(&L,320);
    imprime(L);
}
```

```
void create(lista *q)
{
    q->inicio=NULL;
}

int isEmpty(lista q)
{
    if (q.inicio==NULL)
        return TRUE;
    else
        return FALSE;
}
```


TAD - Lista Simplesmente Encadeada




```
main( )
{
    lista L;
    create(&L);
    insert(&L,12);
    insert(&L,320);
    insert(&L,197);
    insert(&L,26);
    imprime(L);
    remover(&L,320);
    imprime(L);
}
```

```
int insert(lista *q, int d)
{
    struct no *aux, *atual, *anterior;
    aux = (struct no *) malloc(sizeof(struct no));
    if (aux!=NULL)
    {
        aux->dado=d;    aux->prox=NULL;
        anterior = NULL;  atual = q->inicio;
        while (atual != NULL && d > atual->dado)
        {
            anterior = atual;
            atual = atual->prox;
        }

        if (anterior == NULL)
        {
            aux->prox = q->inicio;
            q->inicio = aux;
        }
        else
        {
            anterior->prox=aux;
            aux->prox = atual;
        }
    }
}
```

TAD - Lista Simplesmente Encadeada




```
main( )
{
    lista L;
    create(&L);
    insert(&L,12);
    insert(&L,320);
    insert(&L,197);
    insert(&L,26);
    imprime(L);
    remover(&L,320);
    imprime(L);
}
```

```
void imprime(lista q)
{
    struct no *aux;

    aux = q.inicio;
    if (!isEmpty(q))
    {
        while (aux != NULL)
        {
            printf("%d ", aux->dado);
            aux = aux->prox;
        }
    }
}
```


TAD - Lista Simplesmente Encadeada



```
main( )
{
    lista L;
    create(&L);
    insert(&L,12);
    insert(&L,320);
    insert(&L,197);
    insert(&L,26);
    imprime(L);
    remover(&L,320);
    imprime(L);
}
```

```
int remover(lista *q, int d)
{
    struct no *aux, *atual, *anterior;
    if (d == (q->inicio)->dado)
    {
        aux = q->inicio;
        q->inicio = (q->inicio)->prox;
        free(aux); return(TRUE);
    }
    else
    {
        anterior = q->inicio;
        atual = (q->inicio)->prox;
        while (atual != NULL && atual->dado != d)
        {
            anterior = atual;
            atual = atual->prox;
        }
        if (atual != NULL)
        {
            aux = atual;
            anterior->prox = atual->prox;
            free(aux);
            return(TRUE);
        }
    }
}
```


TAD - Lista Simplesmente Encadeada



```
main( )
{
    lista L;
    create(&L);
    insert(&L,12);
    insert(&L,320);
    insert(&L,197);
    insert(&L,26);
    imprime(L);
    remover(&L,320);
    imprime(L);
}
```

```
void imprime(lista q)
{
    struct no *aux;

    aux = q.inicio;
    if (!isEmpty(q))
    {
        while (aux != NULL)
        {
            printf("%d ", aux->dado);
            aux = aux->prox;
        }
    }
}
```