

Lista Estática

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados I (AE22CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

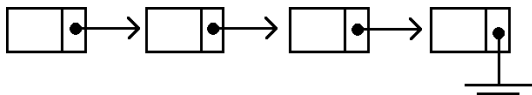
- Listas
- Listas Estáticas
- TAD Listas Estáticas

- Programas operam sobre dados
- Estrutura de dados
- Abstração de informações
- Conjunto de dados
- Conjuntos são tão fundamentais para a Computação quanto para a Matemática

- Abstração de conjunto de dados
 - Alocação sequencial

Endereço na memória	3000	3001	3003	3004	
Conteúdo na memória

- Alocação por encadeamento



Listas

- Listas são coleções de elementos (variáveis, registros, etc)
- Agrupam informações referentes a um conjunto de elementos que se relacionam entre si de alguma forma
- As listas estão presentes em várias situações e aplicações do nosso cotidiano
 - Compras
 - Lista de presença
 - Contatos no smartfone
 - Livros

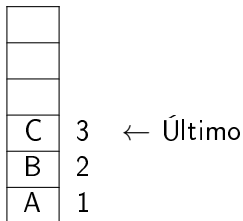
- Aplicações
 - Adequadas para aplicações em que não é possível prever a demanda por memória
 - Manipulação simbólica
 - Gerenciamento de memória
 - Simulações
 - Processamento de imagens
 - Compiladores
- Listas podem ser do tipo linear ou não-linear (generalizada)
- Tipos especiais de listas
 - Fila
 - Pilha

- Uma lista linear é uma sequência de 0 ou mais itens x_1, x_2, \dots, x_n
 - n é o tamanho da lista
 - x_i é o i -ésimo elemento da lista
 - x_1 é o primeiro item da lista e x_n o último
 - x_i sucede x_{i-1} e precede x_{i+1}
 - Se $n = 0$, então a lista é vazia

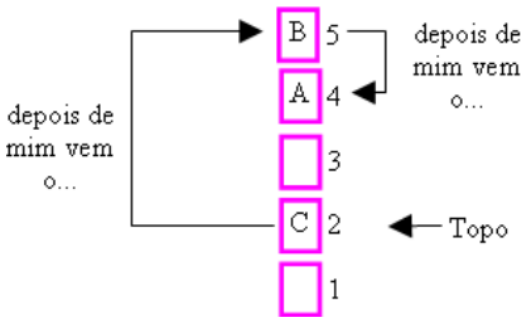
- Operações básicas em uma lista:
 - Criar uma lista vazia
 - Inserir um elemento
 - Remover um elemento
 - Verificar se a lista está vazia
 - Procurar um elemento
 - Concatenar duas ou mais listas

- Operações básicas em uma lista:
 - Dividir uma lista em duas ou mais listas
 - Ordenar a lista
 - Imprimir todos os elementos
 - Retornar referência (e.g. primeiro ou próximo item da lista)
 - Liberar lista
 - Etc

- Alocação sequencial: elementos são alocados em sequência (sequência "física")
 - Geralmente são utilizadas estruturas estáticas (vetores)



- Alocação encadeada: elementos não estão necessariamente em posições adjacentes de memória (sequência "lógica" ou "virtual")
 - Geralmente são utilizadas estruturas dinâmicas (apontadores)

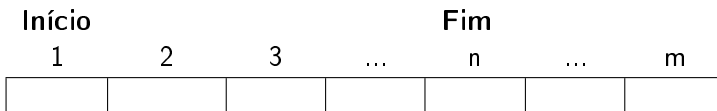


Listas Estáticas

- Os itens dessa estrutura são armazenados em posições contíguas da memória
- A lista estática pode ser percorrida, linearmente, em qualquer direção
- Armazenamento em arranjos (*arrays*/vetores)

Endereço na memória	3000	3001	3003	3004	
Conteúdo na memória

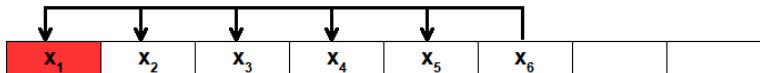
- Uma forma de implementação de listas estáticas, além de considerar um vetor, também podem ser definidas
 - Uma variável para indicar a posição do primeiro elemento da lista (útil em filas estáticas)
 - Uma variável para indicar a posição do último elemento



- Um novo item pode ser inserido no final da lista com custo constante



- No entanto, a remoção de item requer o deslocamento de itens para preencher o espaço vazio



- Vantagens
 - Simples implementação
 - Economia de memória
- Desvantagens
 - Custo para retirar itens
 - Caso a lista atinja o limite de armazenamento, não é possível realocar memória

TAD Listas Estáticas

- Geralmente, em uma lista é necessária as seguintes operações
 - Criar uma lista vazia
 - Inserir um item
 - Remover um item
 - Acessar um item
 - Verificar se a lista está vazia
 - Verificar se a lista está cheia
 - Imprimir a lista
 - Liberar lista

- Primeiro passo: definir arquivo .h

```
// List.h
#define MAX_SIZE 100 // tamanho máximo da lista

typedef struct List List;
```

- Primeiro passo: definir arquivo .h

```
// List.h
List* create();

int vazia(List *l);

int cheia(List *l);

int buscar(List *l, int key);

int inserir(List *l, int key);

int remover(List *l, int key);

void imprimir(List *l);

void liberar(List *l);
```

- Segundo passo: definir arquivo .c

```
// List.c
#include "List.h"

struct List{
    int item[MAX_SIZE];
    int pos_last;
};

List* criar(){
    List *l = (List*) malloc(sizeof(List));
    l->pos_last = -1;

    return l;
}
```

```
int vazio(List *l){
    return l->pos_last < 0;
}

int cheio(List *l){
    return l->pos_last >= (MAX_SIZE - 1);
}

int buscar(List l, int key){
    int i;

    for (i = 0; i <= l->pos_last; i++)
        if (key == l->item[i])
            return i;

    return -1;
}
```

TAD Listas Estáticas

```
int inserir(List *l, int key){  
    if (!cheio(l)){  
        l->pos_last++;  
        l->item[l->pos_last] = key;  
        return 1;  
    }  
  
    return 0;  
}
```



```
int remover(List *l, int key){
    int i;
    int p = search(*l, key);

    if (p >= 0){
        for (i = p; i < l->pos_last; i++)
            l->item[i] = l->item[i + 1];

        l->pos_last--;
        return 1;
    }

    return 0;
}
```





TAD Listas Estáticas

```
void imprimir(List *l){
    int i;

    for (i = 0; i <= l->pos_last; i++)
        printf("%d\n", l->item[i]);
}

void liberar(List *l){
    free(l);
}
```

- Exercício: aproveitando o TAD anterior, faça:
 - Implemente uma função que concatena duas listas
 - Altere o TAD de forma que os itens devam estar ordenados
 - Implemente uma função que intercale duas listas em uma terceira de forma ordenada

-  Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Clifford, S.
Algoritmos: teoria e prática.
Elsevier, 2012.
-  Pereira, S. L.
Estrutura de Dados e em C: uma abordagem didática.
Saraiva, 2016.
-  Szwarcfiter, J.; Markenzon, L.
Estruturas de Dados e Seus Algoritmos.
LTC, 2010.
-  Tenenbaum, A.; Langsam, Y.
Estruturas de Dados usando C.
Pearson, 1995.



Ziviani, M.

Projetos de Algoritmos: com implementações em Pascal e C.
Thomson, 2004.