Introdução à Computação II

Listas estáticas

Profa.: ***Adriane Beatriz de Souza Serapião***

adriane@rc.unesp.br

Listas

- **Uma lista é uma estrutura que armazena elementos de forma alinhada, ou seja, com elementos dispostos um após o outro, como em uma lista de nomes, peças, valores, pessoas, compras, etc.**
- **Uma lista, como um array, pode ser implementada como uma sequência de 'records' com elementos disponíveis de forma consecutiva – Lista Estática Sequencial – ou não consecutiva – Lista Estática Encadeada.**
- **Pascal permite construir estruturas de dados avançadas – Listas Dinâmicas –, mais versáteis, utilizando *ponteiros* e *variáveis dinâmicas*.**

Listas estáticas sequenciais

- **Uma lista sequencial pode ser ordenada ou não-ordenada.**
- **As operações referentes a elas dependem da organização, pois o tipo de manipulação de uma lista ordenada não é o mesmo de uma lista desordenada.**
- **Um exemplo é a inserção de um novo elemento. Numa lista ordenada sob algum critério, a inserção só pode ocorrer num determinado lugar, enquanto que numa lista desordenada, ela pode ocorrer em qualquer lugar.**

Listas estáticas sequenciais

- **Uma lista estática sequencial (ou linear) é um array de registros onde estão estabelecidos regras de precedência entre seus elementos ou é uma coleção ordenada de componentes do mesmo tipo.**
- **O sucessor de um elemento ocupa posição física subsequente.**
  - **Ex: lista telefônica, lista de alunos**

Listas estáticas sequenciais

- **Características de Lista Estática Sequencial:**
  - **armazenados fisicamente em posições consecutivas;**
  - **inserção de um elemento na posição a(i) causa o deslocamento a direita do elemento de a(i) ao último;**
  - **eliminação do elemento a(i) requer o deslocamento à esquerda do a(i+1) ao último;**
- **Mas, absolutamente, uma lista estática sequencial ou é vazia ou pode ser escrita como (a(1), a(2), a(3), ... a(n)) onde a(i) são átomos de um mesmo conjunto S.**
- **Além disso, a(1) é o primeiro elemento, a(i) precede a(i+1), e a(n) é o último elemento.**

Listas estáticas sequenciais

- **Uma das formas mais simples de interligar os elementos de um conjunto.**
- **Estrutura em que as operações inserir, retirar e localizar são definidas.**
- **Podem crescer ou diminuir de tamanho durante a execução de um programa, de acordo com a demanda.**
- **Itens podem ser acessados, inseridos ou retirados de uma lista.**

Listas estáticas sequenciais

- **Assim as propriedades estruturadas da lista permitem responder a questões como:**
  - **qual é o primeiro elemento da lista;**
  - **qual é o último elemento da lista;**
  - **quais elementos sucedem um determinado elemento;**
  - **quantos elementos existem na lista;**
  - **inserir um elemento na lista;**
  - **eliminar um elemento da lista;**

- **Consequência: As quatros primeiras operações são feitas em tempo constante. Mas, as operações de inserção e remoção requerem mais cuidados.**

Listas estáticas sequenciais

- **O conjunto de operações a ser definido depende de cada aplicação.**
- **Um conjunto de operações necessário a uma maioria de aplicações é:**
  - **Criar uma lista linear vazia.**
  - **Inserir um novo item imediatamente após o *i*-ésimo item.**
  - **Retirar o *i*-ésimo item.**
  - **Localizar o *i*-ésimo item para examinar e/ou alterar o conteúdo de seus componentes.**
  - **Combinar duas ou mais listas lineares em uma lista única.**
  - **Particionar uma lista linear em duas ou mais listas.**
  - **Copiar lista linear.**
  - **Ordenar os itens da lista em ordem ascendente ou descendente, de acordo com alguns de seus componentes.**
  - **Pesquisar a ocorrência de um item com um valor particular em algum componente.**

Operações sobre listas dinâmicas encadeadas

- **Inserção de itens:**
  - **No começo de uma lista**
  - **No final de uma lista**
  - **No meio de uma lista**

Implementações de listas estáticas sequenciais

- **Exemplo de Conjunto de Operações:**

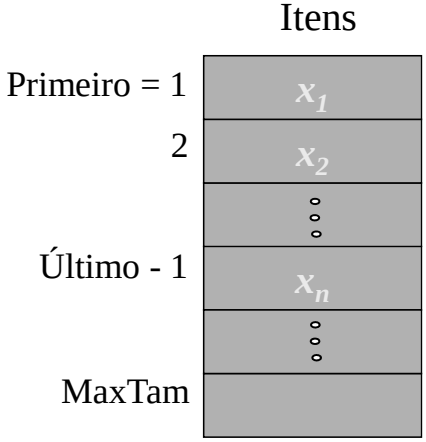
1. **FLVazia(Lista) = faz a lista ficar vazia.**
2. **InsereFim(x, Lista) = insere x após o último item da lista.**
3. **Retira(p, Lista, x) = retorna o item x que está na posição p da lista, retirando-o da lista e deslocando os itens a partir da posição p+1 para as posições anteriores.**
4. **Vazia(Lista) = esta função retorna *true* se lista vazia; senão retorna *false*.**
5. **Imprime(Lista) = imprime os itens da lista na ordem de ocorrência.**

Implementação de listas estáticas sequenciais

- **Os itens da lista são armazenados em posições contíguas de memória.**
- **A lista pode ser percorrida em qualquer direção.**
- **A inserção de um novo item pode ser realizada após o último item com custo constante.**
- **A inserção de um novo item no meio da lista requer um deslocamento de todos os itens localizados após o ponto de inserção.**

Implementações de listas estáticas sequenciais

- **Retirar um item do início da lista requer um deslocamento de itens para preencher o espaço deixado vazio.**



Estrutura da lista estática sequencial

- **Os itens são armazenados em um** array **de tamanho suficiente para armazenar a lista.**
- **O campo Último aponta para a posição seguinte a do último elemento da lista.**
- **O *i*-ésimo item da lista está armazenado na *i*-ésima posição do** array**, 1<= i <Último.**
- **A constante MaxTam define o tamanho máximo permitido para a lista.**

Estrutura da lista estática sequencial

```
#define InicioArranjo = 0;
#define MaxTam = 1000;
typedef int TipoChave;
typedef int Apontador;

typedef struct  {
          TipoChave Chave;
          /* outros componentes */
} TipoItem;

Typedef struct {
     TipoItem Item[MaxTam];
     Apontador Ultimo;
} TipoLista ;
```

Operações sobre listas estáticas sequenciais

```
void FLVazia (TipoLista *Lista);
{
   Lista->Ultimo = InicioArranjo;
};
```

```
int Vazia (TipoLista Lista)
{
   return (Lista.Ultimo == InicioArranjo);
};
```


Operações sobre listas estáticas sequenciais

```
InsereFim (TipoItem x, TipoLista *Lista)
{
   if (Lista->Ultimo > MaxTam)
      printf( “Lista esta cheia\n“ );
   else
   {
       Lista->Item[Lista->Ultimo].Chave = x;
       Lista->Ultimo++;
   };
};
```

Operações sobre listas estáticas sequenciais

```
void Retira (Apontador p, TipoLista *Lista,
             TipoItem *Item)
{
   int Aux;
   if Vazia (*Lista) || (p >= Lista->Ultimo)
   {
      printf (“Erro : Posicao nao existe\n“);
      return;
   }
   *Item = Lista->Item[p-1];
   Lista->Ultimo--;
   for (Aux = p; Aux < Lista->Ultimo; Aux++)
       Lista->Item[Aux-1].Chave      =      Lista-
   >Item[Aux].Chave;
};
```


Operações sobre listas estáticas sequenciais

```
void Imprime (TipoLista Lista)
{
  int Aux;
  for(Aux=InicioArranjo;  Aux  <=  (Lista.Ultimo  –
  1); Aux++)
      printf(“%d12\n”, Lista.Item[Aux].Chave);
};
```

Listas estáticas sequenciais

- **Vantagem:**
  - **acesso direto indexado a qualquer elemento da lista;**
  - **tempo constante para acessar o elemento *i* - dependerá somente do índice;**
  - **economia de memória (os apontadores são implícitos nesta estrutura).**
- **Desvantagem:**
  - **custo de movimentação para inserir ou retirar itens da lista, o que pode causar um deslocamento de todos os itens, no pior caso;**
  - **tamanho máximo tem que ser estabelecido em tempo de compilação, isto é, tamanho limitado.**
- **Quando usar:**
  - **listas pequenas;**
  - **inserção/remoção no fim da lista;**
  - **tamanho máximo bem definido.**

Exercício

- **Dada um lista estática L cuja estrutura é:**

```
typedef struct {
    char nome [80];
    int chave
)  tipo_lista;
```

A lista é declarada como segue:

```
tipo_lista L[50];
```

E assumindo que as chaves podem repetir, forneça as chaves que aparecem o menor (ou maior) número de vezes. A saída deve ser as chaves e o número de vezes que elas ocorrem.

Solução

```
void contaChaves(tipo_lista L[], int n) {
    int qtde[n];
    int chaves[n];
    int m = 0;
    for (int i=0; i<n; i++) {
        int achou = 0;
        for (int j=0; j<m; j++) {
            if (chaves[j] == L[i].chave) {
                qtde[j] = qtde[j] + 1;
                achou = 1;
            }
        }
        if (!achou) {
           chaves[m] = L[i].chave;
           qtde[m] = 1;
           m++;
        }
    }
```


Solução

```
    // determinando menor (ou maior) ocorrência de chaves
    int menor = n;                    // int maior = 0;
    for (int j=0; j<m; j++) {
        if (qtde[j] < menor)          // if (qtde[j] > maior)
             menor = qtde[j]'         // maior = qtde[j];
    }

    // imprimindo as chaves com menor (ou maior) ocorrência
     for (int j=0; j<m; j++) {
        if (qtde[j] == menor) {       // if (qtde[j] == maior)
           printf("Chave: %sn", chaves[j]);
           printf("Quantidade: %d\n", qtde[j]);
        }
     }
}
```

Bibliografia

- **ZIVIANI, N. *Projeto de Algoritmos com Implementações em Pascal e C*, (4a. ed.). Thomson, 2011.**