Aula 5 - Listas Estáticas

Automação de uma Bilbioteca

- todos os livros devem ser cadastrados;
- sistema deve informar se um determinado livro está ou não disponível nas estantes;
- se livro não disponível, o usuário poderá aguardar pela liberação do livro se cadastrando em uma fila de espera;
- quando o livro for devolvido e liberado, o primeiro da fila deve ser contatado para vir buscá-lo.

Solução 2

- alocar espaço para 1000 elementos
- todas as 120.000 filas compartilham o mesmo espaço

Problema: como 120.000 filas podem compartilhar a memória reservada a elas?

Dados

- 120.000 livros
- 1 fila de espera para cada livro
- máximo de 1000 pessoas ficam a espera por um dos livros da biblioteca
- até 30 pessoas ficam a espera de um mesmo livro

O que precisamos?

- interligar os elementos de um conjunto
- definir operações de inserir, retirar e localizar
- flexível para alterar seu tamanho (de acordo com a demanda)

Solução 1

- reservar espaço para 120.000 filas (uma para cada livro), com capacidade para 30 pessoas
- 120.000 vetores de tamanho 30
- espaço reservado para 3.600.000 pessoas

$\Rightarrow$ **muito espaço reservado não é utilizado**

Listas Lineares

- Sequência de zero ou mais itens $x_1, x_2, \ldots, x_n$, na qual x_i é de um determinado tipo e n representa o tamanho dsta linear.
- Assumindo $n \geq 1$:
  - x_1 é o primeiro elemento
  - x_n é o último elemento
  - x_i precede x_{i+1} para $i = 1, 2, \ldots, n$
  - x_i sucede x_{i-1} para $i = 1, 2, \ldots, n$
  - x_i é dito estar na i-ésima posição

Exemplos - listas

- telefônica
- clientes de uma agência bancária
- setores de disco a serem acessados por um SO
- pacotes a serem transmitidos por uma rede
- variáveis globais e outra para variáveis locais
- funções do programa
- parâmetros formais de uma função
- parâmetros reais de uma chamada de função

Operações com Lista

- cria uma lista vazia
- verifica se a lista está vazia
- determina o tamanho da lista
- localiza/modifica o nó que contém um determinado valor
- insere novo item imediatamente após (ou antes) o i-ésimo item
- retira o i-ésimo item
- localiza/modifica o i-ésimo item
- combina duas listas em uma única
- exibe os elementos da lista

O conjunto de operações a ser definido sobre os objetos do tipo Lista depende de cada aplicação, não existindo um conjutno adequado a todas as aplicações. No slide acima descrevemos algumas operações que normalmente são utilizadas pela maioria de aplicações.

Implementação – representações

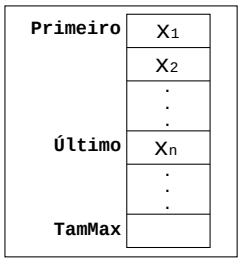
- sequencial (arranjos)
- encadeada (apontadores)

A escolha de uma destas formas dependerá da frequência com que determinadas operações serão executadas sobre a lista.

Existem várias estruturas de dados que podem ser usadas para representar listas lineares, cada uma com vantagens e desvantagens particulares. As representações mais utilizadas são as implementações por meio de arranjos e de apontadores. A implementação por cursores (Aho, Hopcroft e Ullman, 1983 p. 48) pode ser útil em algumas aplicações

Lista Sequencial

- itens são armazenados em posições contíguas de memória:



Explora a sequencialidade da memória do computador, de tal forma que os nós de uma lista sejam armazenados em endereços sequenciais, ou igualmente distanciados um do outro. Pode ser representado por um vetor na memória principal ou um arquivo seqüencial em disco.

Características

- inserção após o último item
- inserção na i-ésima posição requer um deslocamento à direita do i-ésimo elemento ao último
- eliminação i-ésimo elemento requer o deslocamento à esquerda do $i+1$-ésimo ao último

Vantagens

- economia de memória – apontadores implícitos
- acesso direto indexado a qualquer elemento da lista
- tempo constante para acessar o elemento i – dependerá somente do índice

Desvantagens

- custo para inserir/eliminar itens (deslocamento)
- tamanho máximo pré-determinado – ruim em aplicações em que não existe previsão sobre o crescimento da lista

Quando usar?

- listas pequenas
- inserção/eliminação no fim da lista
- tamanho máximo bem definido

Implementação

- `void cria(Lista *p_l);`
- `int vazia(Lista *p_l);`
- `void insere_inicio(Lista *p_l, elem_t e);`
- `void insere_fim(Lista *p_l, elem_t e);`
- `int insere_ordenado(Lista *p_l, elem_t e);`
- `int ordenada(Lista *p_l);`
- `void ordena(Lista *p_l);`
- `int remove_inicio(Lista *p_l, elem_t *p_e);`
- `int remove_fim(Lista *p_l, elem_t *p_e);`

- `int remove_valor(Lista *p_l, elem_t e);`
- `void inverte(Lista *p_l);`
- `void libera(Lista *p_l);`
- `void exibe(Lista *p_l);`

Bibliografia

- Roberto Ferrari, *Curso de estruturas de dados*, São Carlos, 2006. Apostila disponível em: `http://www2.dc.ufscar.br/~ferrari/ed/ed.html`