Listas Lineares com Alocação Sequencial

Estrutura de Dados — QXD0010

Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará
Campus Quixadá

1º semestre/2021

Introdução

Introdução

- Uma estrutura de dados armazena dados na memória do computador a fim de permitir o acesso eficiente dos mesmos.

- A maioria das estruturas de dados usam como recurso principal a memória primária (a chamada RAM) como pilhas, filas, árvores binárias de busca, árvores AVL e árvores rubro-negras.

- Outras são especialmente projetadas e adequadas para serem armazenadas em memórias secundárias como o disco rígido, como as árvores B.

- Uma estrutura de dados bem projetada permite a manipulação eficiente, em tempo e em espaço, dos dados armazenados através de operações específicas.

Lista linear — Definição

- Uma lista linear L é um conjunto de $n \geq 0$ nós (ou células) $L[0], L[1], \ldots, L[n-1]$ tais que suas propriedades estruturais decorrem, unicamente, da posição relativa dos nós dentro da sequência linear:
  - Se $n > 0$, $L[0]$ é o primeiro nó,
  - Para $0 < k \leq n-1$, o nó $L[k]$ é precedido por $L[k-1]$.

Lista linear — Definição

- Uma lista linear L é um conjunto de $n \geq 0$ nós (ou células) $L[0], L[1], \ldots, L[n-1]$ tais que suas propriedades estruturais decorrem, unicamente, da posição relativa dos nós dentro da sequência linear:
  - Se $n > 0$, $L[0]$ é o primeiro nó,
  - Para $0 < k \leq n-1$, o nó $L[k]$ é precedido por $L[k-1]$.
- Os nós de uma lista linear armazenam informações referentes a um conjunto de elementos que se relacionam entre si.
  - Informações sobre os funcionários de uma empresa.
  - Notas de alunos
  - Itens de estoque, etc.

Lista linear – Operações

Algumas operações que podemos querer realizar sobre listas lineares:

- Ter acesso a $L[k]$, $0 \leq k \leq n-1$ qualquer, a fim de examinar ou alterar o conteúdo de seus campos.

Lista linear – Operações

Algumas operações que podemos querer realizar sobre listas lineares:

- Ter acesso a $L[k]$, $0 \leq k \leq n-1$ qualquer, a fim de examinar ou alterar o conteúdo de seus campos.
  - Se fixamos $k = n-1$, temos uma ED chamada Pilha.

Lista linear – Operações

Algumas operações que podemos querer realizar sobre listas lineares:

- Ter acesso a $L[k]$, $0 \leq k \leq n-1$ qualquer, a fim de examinar ou alterar o conteúdo de seus campos.
  - Se fixamos $k = n-1$, temos uma ED chamada Pilha.
  - Se fixamos $k = 0$, temos uma ED chamada Fila.

Lista linear – Operações

Algumas operações que podemos querer realizar sobre listas lineares:

- Ter acesso a $L[k]$, $0 \leq k \leq n-1$ qualquer, a fim de examinar ou alterar o conteúdo de seus campos.
  - Se fixamos $k = n-1$, temos uma ED chamada Pilha.
  - Se fixamos $k = 0$, temos uma ED chamada Fila.
- Inserir um elemento novo antes ou depois de $L[k]$.

Lista linear – Operações

Algumas operações que podemos querer realizar sobre listas lineares:

- Ter acesso a $L[k]$, $0 \leq k \leq n-1$ qualquer, a fim de examinar ou alterar o conteúdo de seus campos.
  - Se fixamos $k = n-1$, temos uma ED chamada Pilha.
  - Se fixamos $k = 0$, temos uma ED chamada Fila.
- Inserir um elemento novo antes ou depois de $L[k]$.
- Remover $L[k]$.

Lista linear – Operações

Algumas operações que podemos querer realizar sobre listas lineares:

- Ter acesso a $L[k]$, $0 \leq k \leq n-1$ qualquer, a fim de examinar ou alterar o conteúdo de seus campos.
  - Se fixamos $k = n-1$, temos uma ED chamada Pilha.
  - Se fixamos $k = 0$, temos uma ED chamada Fila.
- Inserir um elemento novo antes ou depois de $L[k]$.
- Remover $L[k]$.
- Colocar todos os elementos da lista em ordem.

Lista linear – Operações

Algumas operações que podemos querer realizar sobre listas lineares:

- Ter acesso a $L[k]$, $0 \leq k \leq n-1$ qualquer, a fim de examinar ou alterar o conteúdo de seus campos.
  - Se fixamos $k = n-1$, temos uma ED chamada Pilha.
  - Se fixamos $k = 0$, temos uma ED chamada Fila.
- Inserir um elemento novo antes ou depois de $L[k]$.
- Remover $L[k]$.
- Colocar todos os elementos da lista em ordem.
  - Estudaremos algoritmos de ordenação no final do curso.

Lista linear – Operações

Algumas operações que podemos querer realizar sobre listas lineares:

- Ter acesso a $L[k]$, $0 \leq k \leq n-1$ qualquer, a fim de examinar ou alterar o conteúdo de seus campos.
  - Se fixamos $k = n-1$, temos uma ED chamada Pilha.
  - Se fixamos $k = 0$, temos uma ED chamada Fila.
- Inserir um elemento novo antes ou depois de $L[k]$.
- Remover $L[k]$.
- Colocar todos os elementos da lista em ordem.
  - Estudaremos algoritmos de ordenação no final do curso.
- Combinar duas ou mais listas lineares em uma só.

Lista linear – Operações

Algumas operações que podemos querer realizar sobre listas lineares:

- Ter acesso a $L[k]$, $0 \leq k \leq n-1$ qualquer, a fim de examinar ou alterar o conteúdo de seus campos.
  - Se fixamos $k = n-1$, temos uma ED chamada Pilha.
  - Se fixamos $k = 0$, temos uma ED chamada Fila.
- Inserir um elemento novo antes ou depois de $L[k]$.
- Remover $L[k]$.
- Colocar todos os elementos da lista em ordem.
  - Estudaremos algoritmos de ordenação no final do curso.
- Combinar duas ou mais listas lineares em uma só.
- Quebrar uma lista linear em duas ou mais.

Lista linear – Operações

Algumas operações que podemos querer realizar sobre listas lineares:

- Ter acesso a $L[k]$, $0 \leq k \leq n-1$ qualquer, a fim de examinar ou alterar o conteúdo de seus campos.
  - Se fixamos $k = n-1$, temos uma ED chamada Pilha.
  - Se fixamos $k = 0$, temos uma ED chamada Fila.
- Inserir um elemento novo antes ou depois de $L[k]$.
- Remover $L[k]$.
- Colocar todos os elementos da lista em ordem.
  - Estudaremos algoritmos de ordenação no final do curso.
- Combinar duas ou mais listas lineares em uma só.
- Quebrar uma lista linear em duas ou mais.
- Copiar uma lista linear em um outro espaço.

Implementação de uma lista linear

- O modo de implementar listas lineares depende da classe de operações mais frequentes. Não existe, em geral, uma única implementação para a qual todas as operações são eficientes.

Implementação de uma lista linear

- O modo de implementar listas lineares depende da classe de operações mais frequentes. Não existe, em geral, uma única implementação para a qual todas as operações são eficientes.

- Por exemplo, não existe uma implementação para atender às seguintes duas operações de maneira eficiente:

(1) ter acesso fácil ao elemento $L[k]$, para k qualquer.
(2) inserir ou remover elementos em qualquer posição da lista linear.

Implementação de uma lista linear

- O modo de implementar listas lineares depende da classe de operações mais frequentes. Não existe, em geral, uma única implementação para a qual todas as operações são eficientes.

- Por exemplo, não existe uma implementação para atender às seguintes duas operações de maneira eficiente:

(1) ter acesso fácil ao elemento $L[k]$, para k qualquer.
(2) inserir ou remover elementos em qualquer posição da lista linear.

A operação (1) fica eficiente se a lista é implementada em um vetor (array) em alocação sequencial na memória.

Implementação de uma lista linear

- O modo de implementar listas lineares depende da classe de operações mais frequentes. Não existe, em geral, uma única implementação para a qual todas as operações são eficientes.

- Por exemplo, não existe uma implementação para atender às seguintes duas operações de maneira eficiente:

(1) ter acesso fácil ao elemento $L[k]$, para k qualquer.
(2) inserir ou remover elementos em qualquer posição da lista linear.

A operação (1) fica eficiente se a lista é implementada em um vetor (array) em alocação sequencial na memória.

Para a operação (2) é mais adequada a alocação encadeada, com o uso de ponteiros.

Tipos de alocação

O tipo de armazenamento de uma lista linear pode ser classificado de acordo com a posição relativa na memória de dois nós consecutivos na lista.

- **Alocação sequencial:** dois nós consecutivos na lista estão em posições contíguas de memória.

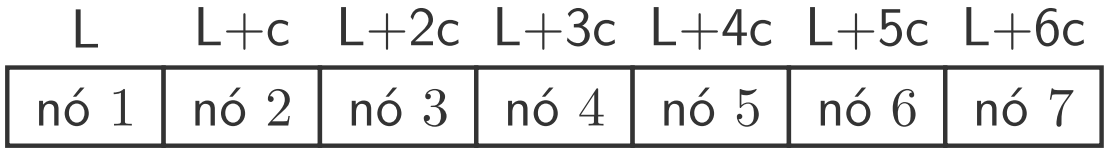
- **Alocação encadeada:** dois nós consecutivos na lista podem estar em posições não contíguas da memória.

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Listas Sequenciais

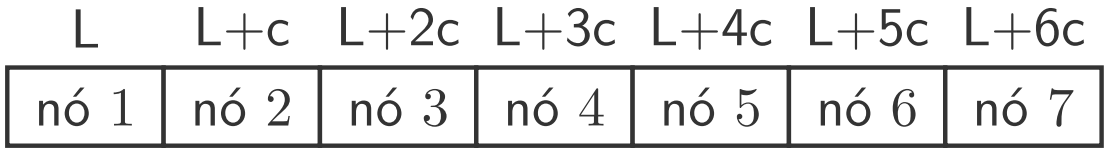
Listas sequenciais (Vetores)

- Nós em posições contíguas da memória.



Listas sequenciais (Vetores)

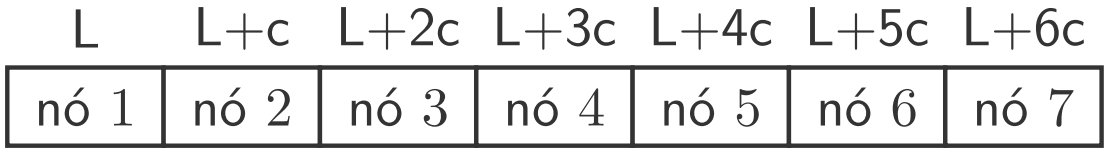
- Nós em posições contíguas da memória.



- Neste caso, o endereço real do $(j+1)$-ésimo nó da lista se encontra c unidades adiante daquele correspondente ao j-ésimo. A constante c é o número de bytes que cada nó ocupa.

Listas sequenciais (Vetores)

- Nós em posições contíguas da memória.



- Neste caso, o endereço real do $(j+1)$-ésimo nó da lista se encontra c unidades adiante daquele correspondente ao j-ésimo. A constante c é o número de bytes que cada nó ocupa.

- A correspondência entre o índice da lista e o endereço real é feita automaticamente pela linguagem de programação quando da compilação do programa.

TAD Lista Sequencial

- Lista sequencial pode ser modelada como um Tipo Abstrato de Dados.

TAD Lista Sequencial

- Lista sequencial pode ser modelada como um Tipo Abstrato de Dados.

- O TAD Lista Sequencial tem os seguintes atributos:
  - um vetor de inteiros.
  - a capacidade total do vetor.
  - a quantidade de elementos no vetor.
  - a posição atual do índice.

TAD Lista Sequencial

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Lista sequencial pode ser modelada como um Tipo Abstrato de Dados.

- O TAD Lista Sequencial tem os seguintes atributos:
  - um vetor de inteiros.
  - a capacidade total do vetor.
  - a quantidade de elementos no vetor.
  - a posição atual do índice.

- Operações possíveis são:
  - Criar lista.
  - Liberar lista.
  - Consultar o tamanho atual da lista.
  - Saber se lista está cheia.
  - Adicionar um elemento ao final da lista.
  - Remover um elemento da lista.

Arquivo AList.h

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  #ifndef ALIST_H
2  #define ALIST_H
3
4  class AList {
5  private:
6      int maxSize;      // capacidade da lista
7      int listSize;     // numero de elementos na lista
8      int pos;          // posicao atual
9      int* listArray;   // ponteiro para array de inteiros
10
11 public:
12     AList(int size);  // Construtor
13     ~AList();  // Destrutor
14
15     // Limpa a lista deixando-a vazia, com 0 elementos.
16     void clear();
17
18     // Insere um elemento na posicao atual
19     // item: o elemento a ser inserido
20     void insert(const int& item);
```

Arquivo AList.h (continuação)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1       // adiciona um elemento ao final da lista
2       // item: o elemento a ser inserido
3       void append(const int& item);
4
5       // Remove e retorna o elemento atual
6       int remove();
7
8       // Configura a posicao atual para o inicio da lista
9       void moveToStart();
10
11      // Configura a posicao atual para o final da lista
12      void moveToEnd();
13
14      // Move a posicao atual uma poicao para tras.
15      // Nada acontece se a posicao ja estiver no inicio da
        lista
16      void prev();
17
18      // Move a posicao atual uma poicao a frente.
19      // Nada acontece se a posicao ja estiver no final da lista
20      void next();
```

Arquivo AList.h (final)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1       // Devolve o numero de elementos da lista
2       int lenght() const;
3
4       // Devolve a posicao do elemento atual
5       int posAtual() const;
6
7       // Configura a posicao atual
8       void moveToPos(int pos);
9
10      // Devolve o elemento atual
11      const int& getValue() const;
12
13      // Devolve true se cheia e false caso contrario
14      bool full() const;
15  };
16
17  #endif
```

Arquivo `AList.cpp`

Exercício: Implementar as funções-membro da classe AList.

Programa cliente main.cpp


```
1  #include <iostream>
2  #include "AList.h"
3  using namespace std;
4
5  int main()
6  {
7    AList list(30);
8
9    for(int i = 0; i < 15; ++i)
10     list.append(i);
11
12   list.moveToStart();
13
14   while(list.posAtual() < list.lenght()) {
15     int it = list.getValue();
16     cout << it << " ";
17     list.next();
18   }
19   cout << endl;
20
21   return 0;
22 }
```

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

FIM