Módulo 4

Listas Encadeadas

Algoritmos e Estruturas de Dados II

C++

(Rone Ilídio)

Ponteiro para struct

- Ponteiro pode apontar para qualquer tipo de dados
- A partir de um ponteiro para uma variável cujo tipo é um struct e do operador “->” pode-se acessar o conteúdo.

```
#include<iostream>
using namespace std;
struct Pessoa{
      char nome[50];
      int idade;
};
int main(){
   Pessoa *p;
   p = new Pessoa;
   cout<<"Digite o nome:";
   cin>> p->nome;
   cout<<"Digite a idade:";
   cin>>p->idade;
   cout << p->nome << " " << p->idade << endl;
   system("pause");
}
```

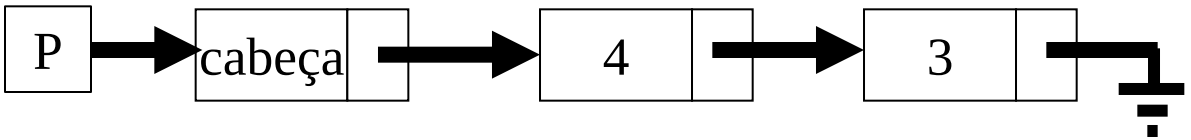
Listas Encadeadas

- Sequência de objetos na memória
- Cada elemento é uma célula com:
  - Conteúdo
  - Apontador para a próxima célula
- Toda lista tem um ponteiro para o início da lista
- O último aponta para NULL

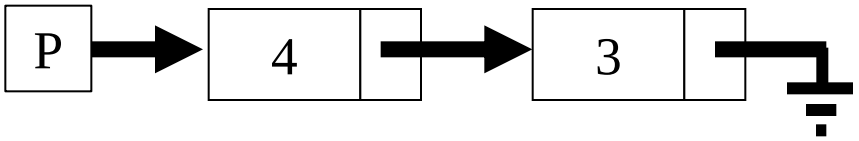
```
struct cel{
   int conteudo;
   cel *seg;
};
typedef cel celula;
```

Listas Encadeadas

- Dois tipos de listas:
  - Com cabeça: primeiro elementos só para facilitar os algoritmos



  - Sem cabeça: o ponteiro do início aponta diretamente para o primeiro elemento



Lista encadeada-Introdução

Com cabeça

```
#include <iostream>
struct cel{
   int conteudo;
   cel *seg;
};
typedef cel celula;

int main(){
    celula ex;
    celula *p;
    p = &ex;
    //Código para manipulação da lista
    system("pause");
}
```

Sem cabeça

```
#include <iostream>
struct cel{
   int conteudo;
   cel *seg;
};
typedef cel celula;

int main(){
    celula *p;
    //Código para manipulação da lista
    system("pause");
}
```

Lista Encadeada-Funções

- Criação
- Funções
  - Insere
  - Imprime
  - Busca
  - Remoção
  - Busca e remoção
- Exemplo básico
  - Usuário insere números da lista
  - Ao final o conteúdo é impresso na tela

Lista Encadeada - Criação

```
int main(){
    celula c;             //Cria o no cabeça
    celula *lst;          //Cria um ponteiro para o cabeca
    c.seg = NULL;  //Lista vazinha
    lst = &c;             //Ponteiro aponta para o cabeca, sempre
    while(1){
        cout<<"\nDigite um número para inserir na lista (sair - 0):";
        int num;
        cin >> num;
        if(num == 0) break;
        insere(num,lst);
    }
    cout << "\n\nConteudo da Lista:\n";
    Imprima(lst);
    getch();
}
```

Lista Encadeada - Insere

→Insere um novo elemento no início da lista

```
void Insere(int y, celula *lst){
    celula *nova;
    nova = new celula;
    nova->conteudo = y;
    nova->seg = lst ->seg;
    lst ->seg = nova;
}
```

Lista Encadeada – Imprima

→Recebe o ponteiro para a cabeça e imprime o conteúdo da lista, do início para o fim

```
void Imprima(celula *lst){
    celula *p;
    for(p=lst->seg; p!=NULL; p=p->seg){
        cout << "\n" << p->conteudo;
    }
}
```

Lista Encadeada

- Exemplo mais complexo
  - Insere
  - Busca
  - Busca e Remove

- Função main ...

```
int main(){
   celula c;      //Cria o no cabeça
   celula *lst;   //Cria um ponteiro para o cabeca
   c.seg = NULL;  //Lista vazinha
   lst = &c;      //Ponteiro aponta para o cabeca
   int num;
   while(1){
      cout<<"\nDigite um numero para inserir na lista (sair - 0):";
      cin >> num;
      if(num == 0) break;
      Insere(num,lst);
   }
   while(1){
     cout << "\n\nDigite um numero (0 - sair):\n";
     cin >> num;
     if (num == 0) break;
     celula *elemento = Busca(num,lst);
     if(elemento == NULL) cout<<"\nElemento nao encontrado.";
     else {
         cout << "\nEncontrado, conteudo: "<< elemento->conteudo << "  Deseja apaga-lo(S/N):";
         char op;
         op = getche();
         if(op == 's') BuscaERemove(num, lst);
     }
   }
   Imprima(lst);
   getch();
}
```

Lista Encadeada - Busca

→Retorna o ponteiro para o elemento com conteúdo igual a x na lista lst passada como parâmetro

```
célula *Busca (int x, celula *lst) {
        celula *p;
        p = lst->seg;
        while (p != NULL && p->conteudo != x)
                p = p->seg;
        return p;
}
```

Lista Encadeada – Busca e Remove

Recebe uma lista lst com cabeça e remove da lista a primeira célula que contiver x,se tal célula existir.

```
void BuscaERemove (int x, celula *lst) {
    celula *p, *q;
    p = lst;
    q = lst->seg;
    while (q != NULL && q->conteudo != x) {
        p = q;
        q = q->seg;
    }
    if (q != NULL) {
        p->seg = q->seg;
        free (q);
    }
}
```

Lista Encadeada - Busca

→Busca um elemento com conteúdo igual a x na lista (recursivo)

```
célula *BuscaR (int x, celula *lst) {
        if (lst->seg == NULL)
                return NULL;
        if (lst->seg->conteudo == x)
                return lst->seg;
        return BuscaR (x, lst->seg);
}
```

Lista Encadeada Ordenada

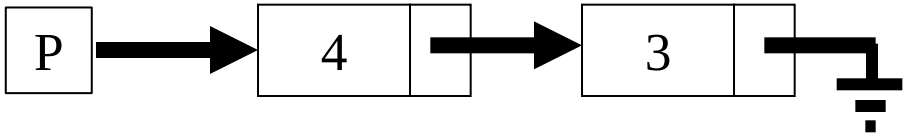
- Os elementos da lista estão em ordem crescente.
- Altera somente função de inserção
- Exemplo:
  - Utilize a função main do slide 7 e a função Imprima do slide 9
  - Foram criadas duas funções para inserção, ambas produzem o mesmo resultado. Entretanto, uma utiliza 2 ponteiro e a outra somente um ponteiro. Apenas uma precisa deve ser utilizada para o exemplo funcionar.

```
void Insere(int y, celula *lst){
    celula *p,*q;
    p = lst;
    q = lst->seg;
    while(q != NULL && q->conteudo < y){
        p = q;
        q = q->seg;
    }
    celula *nova = new celula;
    nova->conteudo = y;
    nova->seg = q;
    p->seg = nova;
}
```

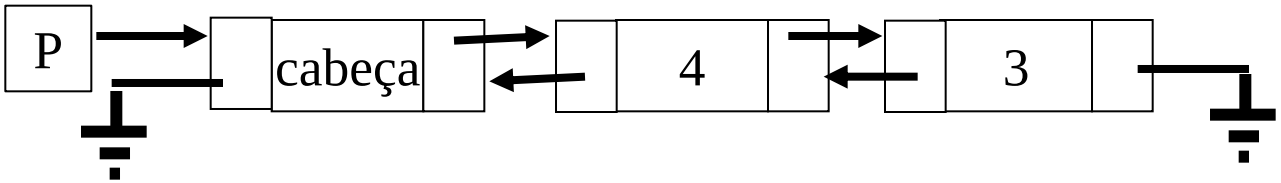
```
void Insere(int y, celula *lst){
    celula *p = lst;
    while(p->seg != NULL && p->seg->conteudo < y){
        p = p->seg;
    }
    celula *nova = new celula;
    nova->conteudo = y;
    nova->seg = p->seg;
    p->seg = nova;
}
```

Tipos de Lista Encadeada

- Simplesmente encadeada sem cabeça



- Duplamente Encadeada (com cabeça ou não)



- Circular (com cabeça ou não)