Universidade de São Paulo – São Carlos
Instituto de Ciências Matemáticas e de Computação

Listas

Parte do Material preparado pela profa
Silvana Maria Affonso de Lara

2º semestre de 2010

Aula Anterior

- Alocação Dinâmica de Memória
- Vetores e alocação dinâmica
- Alocação da memória principal
- Funções para alocar e liberar memória
- Alocação dinâmica de matrizes

Roteiro da Aula

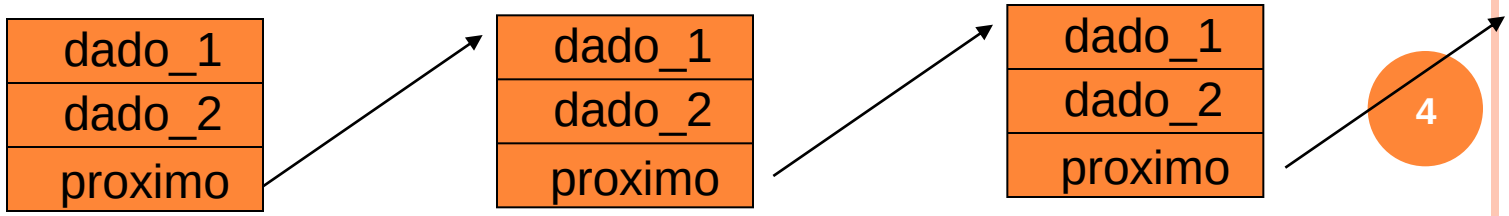
- Estruturas autoreferenciadas
- Listas
- Listas de inteiros
- Listas de alunos

Estruturas Autoreferenciadas

- É possível definir, dentro de uma estrutura, um ponteiro para a própria estrutura:

```
struct Qualquer {
        tipo_dado_1 dado_1;
        tipo_dado_2 dado_2;
            ...
        struct Qualquer *proximo;
}
```

- *próximo* aponta para uma estrutura do mesmo tipo



Listas Dinâmicas

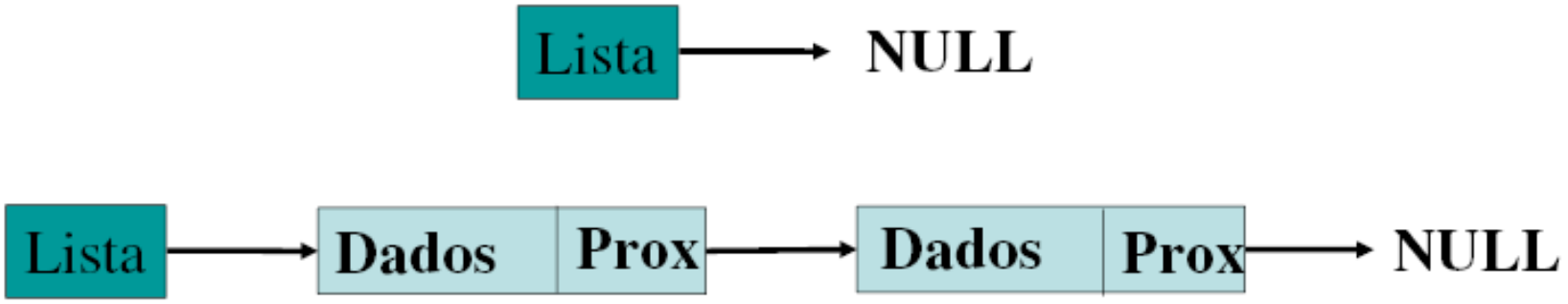
- Comumente chamadas de **Listas Ligadas ou Listas Encadeadas**
- Estrutura de tamanho variável que utiliza apenas a quantidade de memória que precisa.
- São representadas como seqüências de dados definidas pelo encadeamento dos elementos.
- **Cada elemento** é chamado de **nó** da lista e contém os dados e um ponteiro (ou link) para o próximo nó da lista

Listas dinâmicas

- Vantagens
  - A memória é alocada e liberada quando necessário.
  - A alocação dinâmica nos oferece a necessária flexibilidade para mantermos, sem um grande número de movimentações de nós na lista, a estrutura devidamente ordenada a cada inserção e/ou retirada de elemento
  - São adequadas para aplicações onde não é possível prever a demanda por memória, permitindo a manipulação de quantidades imprevisíveis de dados, de qualquer formato
- **Desvantagem**
  - Maior grau de complexidade de implementação

Listas dinâmicas

- Início da Lista
- O início da lista é estabelecido por um ponteiro para o 1o nó da lista.
- Caso a lista esteja vazia, inicialmente o ponteiro aponta para NULL ou None



Listas dinâmicas

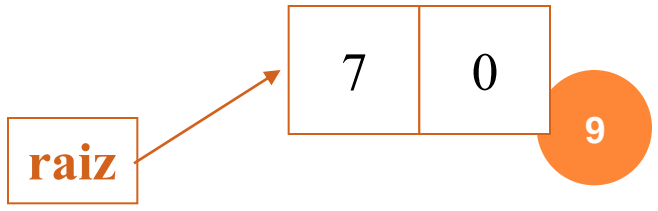
- Operações Básicas
  - Criar a Lista
  - Inserir elemento
  - Remover elemento
  - Consultar elemento
  - Alterar elemento
  - Listar os elementos

Lista de Inteiros

- Exemplo: lista encadeada de inteiros. *Raiz* é um ponteiro para o primeiro nodo da lista.

```
struct IntNode {
  int dado;
  struct IntNode *proximo;
} *raiz;

raiz = (IntNode *)malloc(sizeof(IntNode));
raiz->dado = 7;
raiz->proximo = NULL;
```

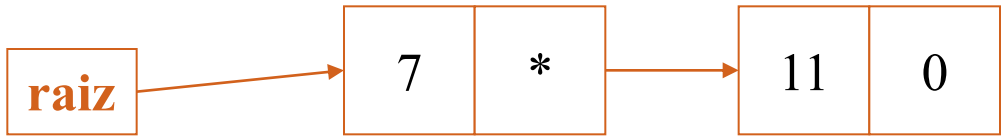


Lista de Inteiros

- Inserindo elementos na lista:

```
IntNode *pnode;
pnode = (IntNode *)malloc(sizeof(IntNode));
pnode->dado = 11;
pnode->proximo = NULL ;        /*terminador de lista*/

raiz->proximo = pnode;
```



Percorrendo a Lista

- Percorre-se uma lista encadeada por meio de um ponteiro:

```
...
IntNode *pnode;
pnode = raiz;
while (pnode != NULL) {
        printf("%d ", pnode->dado);
        pnode = pnode->proximo;
}
```

Insere Nó

```
 struct IntNode *insere_int (int i,  struct IntNode *pinicio)
{
 struct IntNode *pi;

 pi=(struct IntNode *) malloc(sizeof(struct IntNode));
 if (pi) {
    pi->dado = i;
    pi->proximo = pinicio;
     pinicio = pi;
 }
    return pi;
}
```

Print Lista

```
void print_list (struct IntNode *pi) {

 printf("\nLista = ");
 while (pi)      {
   printf (" %d ", pi->dado);
   pi = pi->proximo;
 }
}
```

Lista de Alunos

```
struct aluno {

   char *nome;
   short idade;
   char matricula[8];
   struct aluno *prox;
};

struct aluno *ListaAlunos;
/* variavel global */
```

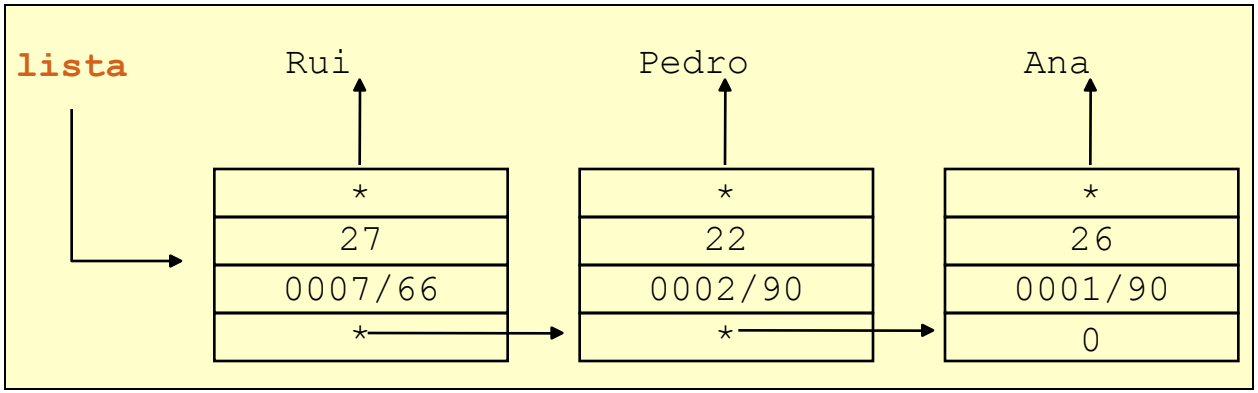
Insere Aluno

```
int insere_aluno (char *n, short id, char *m)
{
   struct aluno *pa;

 pa = (struct aluno *)malloc(sizeof(struct aluno));
 pa->nome = n;
 pa->idade = id;
 strcpy(pa->matricula, m);
 pa->prox = ListaAlunos;
 ListaAlunos = pa;
}
```

Criação Lista Alunos

```
void main () {
 insere_aluno("Ana", 26, "0001/90");
 insere_aluno("Pedro", 22, "0002/90");
 insere_aluno("Rui", 27, "0007/66");
}
```



Esvaziando a Lista

```
/* esvazia a lista */
void esvazia_lista () {
struct aluno *tmp;

   while (ListaAlunos != NULL) {
         tmp = ListaAlunos ;
         ListaAlunos = lista->prox;
         free(tmp);
   }
}
```

Exemplo de Lista Encadeada

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

struct aluno{
        char nome[20];
        int idade;
        struct aluno *prox;
};
```

Exemplo de Lista Encadeada

```
main() {
    struct aluno *lista, *aux, *p;
   char num[4];
   char alt[4];
   int i, tipo;

   lista=NULL;
```

Exemplo de Lista Encadeada

```
for(i=0;i<5;i++) {
      aux = (struct aluno *)malloc(sizeof(struct aluno));

      if (aux){
         printf("\nDigite o nome do aluno %d:",i);
         gets((*aux).nome);
         fflush(stdin);
         printf("\nDigite a idade do aluno %d:",i);
         scanf("%d",&(*aux).idade);
         fflush(stdin);
         printf("\nDigite o tipo de encadeamento 1-cabeca,
    2-final:");
         scanf("%d", &tipo);
         fflush(stdin);
```

Exemplo de Lista Encadeada

```
//encadeamento inserindo o novo elemento sempre
   na primeira posicao (cabeça da lista)

                if (tipo == 1){
                   (*aux).prox=lista;
                   lista=aux;
                }
```

Exemplo de Lista Encadeada

```
  else{
//encadeamento inserindo o novo elemento sempre na ultima posicao da lista

                    aux->prox=NULL;
                    if (lista){
                       p=lista;
                       while (p->prox != NULL)
                             p=p->prox;
                       p->prox=aux;
                    }
                    else{
                       lista=aux;
                    }
          }
      }
}
```

Exemplo de Lista Encadeada

```
//impressão do conteúdo da lista
aux=lista;
printf("\n\n\nLista = ");
while (aux) {
   printf (" %s ", aux->nome);
   aux = aux->prox;
}
getch();
}
```

Universidade de São Paulo – São Carlos
Instituto de Ciências Matemáticas e de Computação

Listas

Parte do material preparado pela profa Silvana Maria Affonso de Lara

2º semestre de 2009