Listas - Outras

Listas Circulares

Nós Cabeça

Listas Duplamente Ligadas/Encadeadas

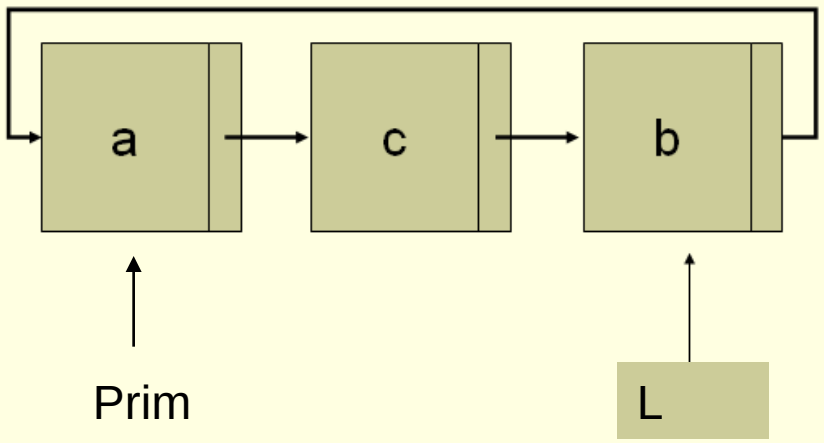
Aplicações

5/10/2010

Listas Circulares Encadeadas Dinâmicas

- Se o nó next do último nó apontar para o primeiro, teremos uma lista circular.
- Em listas circulares não temos primeiro nem último naturais.
- O esquema abaixo é o mais usual: como não há primeiro nem último nó, fazemos o nó L ser o último e o próximo ser o primeiro.

- Lista vazia: NULL



Representação/Implementação

```
struct rec {
      elem info;
      struct rec *lig;
};
typedef struct rec *recptr;
recptr  L;
```

Operações

```
void Criar(recptr *L){
     *L= null;
 }
void Insere_Prim(recptr *L, int valor){
recptr p =(recptr) malloc(sizeof(struct rec));
p->info= valor;
p->lig= p;
*L= p;
}
```

Deixem para relatar os erros de Memória insuficiente quando estiverem fazendo o TAD, assim uniformizam todos os retornos de erros.

Insere depois de um nó k

```
void Insere_Depois(elem v, recptr k) {

recptr j =(recptr) malloc(sizeof(struct rec));
j->info = v;
j->lig = k->lig;
k->lig = j;

}
```

Deixem para relatar os erros de Ponteiro Nulo quando estiverem fazendo o TAD, assim uniformizam todos os retornos de erros.

Se a inserção se dá após o último nó (após L) o nó L deve ser modificado

(Façam)

```
void Insere_Depois(recptr *L, elem v, recptr k){

recptr j =(recptr) malloc(sizeof(struct rec));
j->info = v;
j->lig = k->lig;
k->lig = j;
if (*L == k) *L = j;

}
```

Deleta depois de um nó p

- Se houver 1 único nó nós vamos querer deletar ele?
  - Conceitualmente não!
- Se é vazia não podemos deletar nada.

- Mas podemos querer remover mesmo que haja só 1 nó
  - O nó L deve ser modificado.
- Vamos pensar em erros nesta operação,
  - pois não vou fazer a função deleta.

Primeira opção

```
int deleta_depois(recptr p, elem *px){
recptr q;

if ((p == NULL) || (p == p->lig)){
        return 0;
else {
      q=p->lig;
      *px = q->info;
      p->lig = q->lig;
      free(q);
      return 1;
}
}
```

Não pode ser usado para deletar o último nó (nó L).

Segunda opção – deleta mesmo que tenha 1 único nó

```
int deleta_depois(recptr *L, recptr p, elem *px){
recptr q;

if (p ! = NULL) {

   if (p == p->lig)){
            free(p); *L = NULL; return 1;
   }
   else {
             q=p->lig;
             *px = q->info;
             p->lig = q->lig;
            if (q = *L) *L = p; //remoção de L
             free(q);
            return 1;
       }
else return 0;
}
```

Vantagens de Listas Circulares

- Podemos varrer a lista completamente, a partir de qualquer ponto e voltar ao início.
- Algumas operações como concatenação e divisão são mais eficazes.
- Implementem concatenação em Listas Circulares (L1 e L2), devolvendo o resultado em L1.