Lista Circular, cabeça de lista, lista duplamente ligada

Sétima aula de Estrutura de dados

Mac2301 - primeiro semestre de 2002

1 Listas circulares ligadas

Uma lista circular ligada tem a propriedade que o último nó aponta sempre para o primeiro nó. Então de qualquer nó da lista pode-se atingir qualquer outro nó da lista.

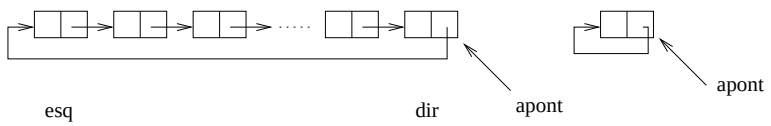

Figura 1: Listas circulares: Caso genérico e lista com apenas um nó.

funções para a manipulação de listas circulares:

- Variáveis:

```
ApontoNo apont;
```

- Inicialização:

```
void InicializaListaCircular(){
  apont = NULL;
}
```

- Inserção à esquerda:

```
void InsereEsquerda(TipoInfo x){
  ApontNo p;
  ExtraiNo(&p);
  p->info = x;
  if (apont==NULL){  /* Lista vazia */
     p->prox = p;
     apont = p;
  }
  else {
     p->prox = apont->prox;
     apont->prox = p;
  }
}
```

- Inserção à direita:

```
void InsereDireita(TipoInfo x){
  InsereEsquerda(x);
  apont = apont->prox;
}
```

- Remoção à esquerda:

```
void RemoveEsquerda(TipoInfo *x){
  ApontNo p;
  if (apont == NULL)
     Underflow();
  else {
     p = apont->prox;
     *x = p->info;
     if (apont==p) /* Lista com apenas um nó */
        apont = NULL;
     else
        apont->prox = p->prox;
     DevolveNo(p);
  }
}
```

Observações:

1. Remoções à direita não são eficientes;
2. Uma lista circular pode ser usada como algumas estruturas já vistas, considerando:
   (a) Inserir à esquerda;
   (b) Inserir à direita;
   (c) Remover à esquerda.

- pilha: combinando (a) e (c);
- fila: combinando (b) e (c);
- fila dupla com saída restrita: combinando (a), (b) e (c).

2 Operações mais eficientes com lista circular

Devolver uma lista inteira para a lista livre

```
ApontNo p;
if (apont!=NULL){
   p = apont->prox;
   apont->prox = livre;
   livre = p;
   apont = NULL;
}
```

Concatenar duas listas circulares ligadas

Como concatenar as listas apont1 e apont2 na lista apont?

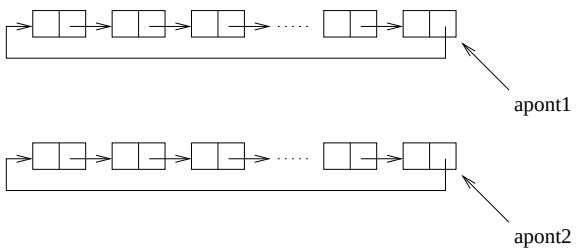

Figura 2: Caso genérico de duas listas circulares.

```
ApontNo p;
if (apont2 == NULL)
   apont = apont1;
else {
   if (apont1 != NULL){
      p = apont1->prox;
      apont1->prox = apont2->prox;
      apont2->prox = p;
   }
   apont = apont2;
}
```

Nó cabeça de lista (head of the list)

Nó cabeça de lista é um nó utilizado para indicar o início e o fim de uma lista circular, e armazena no campo info um valor especial, que não é um "valor válido" para os outros nós da lista.

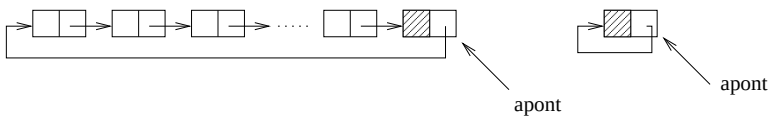

Figura 3: Listas circulares com cabeça de lista: Caso genérico e lista vazia.

Aplicações de listas circulares com "cabeça" de lista

Representação de um polinômio. Cada nó tem o formato Cada nó tem o formato:

coeficiente	expoente	prox

e os nós estão ligados em ordem crescente de expoente.

Exemplo: $p(x) = 7x^{10} + 5x^2 + 4$

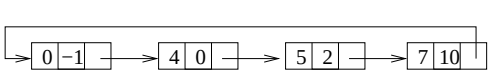

Figura 4: Polinômio representado por lista circular com cabeça de lista.

Representação de inteiros positivos arbitrariamente grandes.

Uma idéia é armazenar os dígitos da direita para a esquerda numa lista circular ligada, de modo que o primeiro nó contenha o dígito menos significativo e o último nó contenha o dígito mais significativo. Ex: O número 12345678901234 é representado por:

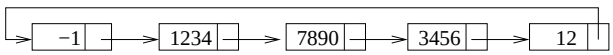

Figura 5: Número arbitrariamente grande representado por lista circular com cabeça de lista.

3 Listas duplamente ligadas

Cada nó tem o formato: | eprox | info | dprox | onde:

- info: contém a informação;
- eprox: armazena o endereço do nó predecessor;
- dprox: armazena o endereço do nó sucessor.

3.1 Representação gráfica

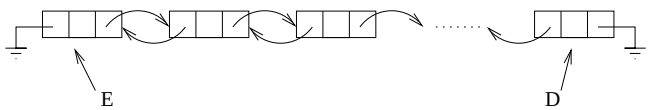

Figura 6: Listas duplamente ligada: Caso genérico.

3.2 Um pouco de código

- preliminares:

```
typedef struct no1 {
          TipoInfo info;
          struct no1  *eprox, *dprox;
        } No, *ApontNo;
ApontNo E, D;
```

- Inicialização:

```
void InicializaListaDupla(){
  E = D = NULL;
}
```

- Inserção à direita:

```
void InsereDireita(TipoInfo x){
  ApontNo p;
  ExtraiNo(&p);
  p->info = x;
  p->dprox = NULL;
  p->eprox = D;
  if (D == NULL) /* lista vazia */
    E = D = p;
  else {
    D->prox = p;
    D = p;
  }
}
```

- Remoção à esquerda:

```
void RemoveEsquerda(TipoInfo *x){
  ApontNo p;
  if (E == NULL)
     Underflow();
  else {
    p = E;
    *x = p->info;
    E = E->dprox;
    if (E == NULL) /* lista ficou vazia */
      D = NULL;
    else
      E ->eprox = NULL;
    DevolveNo(p);
   }
}
```

3.3 Observações

1. Podemos ter variações como a lista circular duplamente ligada;

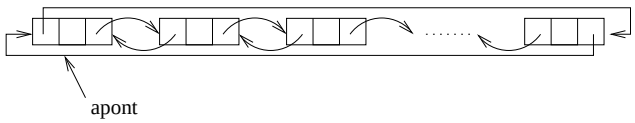

Figura 7: Lista circular duplamente ligada.

2. Lista circular duplamente ligada com cabeça de lista;

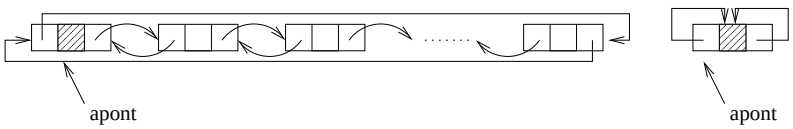

Figura 8: Lista circular duplamente ligada com cabeça de lista: situação genérica e lista vazia.

3. Numa lista circular duplamente ligada, todo o nó `p`, inclusive o cabeça de lista satisfaz: `dprox(eprox(p)) = eprox(dprox(p)) = p`;

4. Listas duplamente ligadas requerem um espaço adicional para as ligações, mas alguma operações tornam-se mais eficientes. Por exemplo, remover um nó apontado por p.

```
/* esta funcao supoe uma lista circular duplamente ligada com cabeca
   de lista, e p diferente do no cabeca */
void RemoveNo (ApontNo p, TipoInfo *x){
   *x = p->info;
   (p->eprox)->dprox = p->dprox;
   (p->dprox)->eprox = p->eprox;
   DevolveNo(p);
}
```