MC102 - Algoritmos e Progração de Computador

Prof. Alexandre Xavier Falcão

22° Aula: Listas

1 Listas

Uma **lista** é uma seqüência dinâmica de elementos, denominados **nós**. Por exemplo, podemos usar uma lista para armazenar elementos de um conjunto, que podem ser inseridos e removidos do conjunto durante a execução do programa. Neste sentido, listas são dinâmicas; i.e., possuem tamanho variável durante a execução do programa. Portanto, os nós de uma lista são normalmente armazenados em regiões alocadas dinamicamente em memória e interligadas por apontadores. Para cada nó da lista, podemos ter um apontador para o nó seguinte, e a lista é dita **ligada simples**. Podemos também manter um apontador para o nó anterior, além do apontador para o próximo nó. Neste caso, a lista é dita **ligada dupla**. Neste curso veremos apenas os casos de listas ligadas simples.

A lista será representada por um apontador para o primeiro nó. Quando a lista estiver vazia, o apontador deve ser NULL. Podemos inserir novos nós no início, antes/após um dado elemento, ou no final da lista, **sempre cuidando para que o apontador da lista continue apontando para o primeiro nó**. A mesma observação vale para remoções no início, antes/após um dado elemento, e no final. O programa abaixo ilustra algumas operações com uma lista ligada simples.

```
#include <malloc.h>

typedef enum {false,true} bool;

typedef struct _no {
  int valor;
  struct _no *prox;
} No, Lista;

No    *CriaNo(int val); /* Cria e inicializa nó com valor em val */
void   DestroiLista(Lista **p); /* Destroi lista na memória */
bool   ListaVazia(Lista *p); /* Verifica se a lista está vazia */
void   InsereInicioLista(Lista **p, int val); /* Insere elemento no
                                                 início da lista */
void   ImprimeLista(Lista *p); /* Imprime elementos da lista */
bool   RemoveInicioLista(Lista **p, int *val); /* Remove elemento do
                                                  início da lista
                                                  retornando em val o
                                                  seu valor */
```

```
No  *CriaNo(int val)
{
  No *p;
  p        = (No *)calloc(1,sizeof(No)); /* a memória alocada em uma
                                            função, permanece alocada
                                            no programa até que seja
                                            liberada pelo comando free
                                            ou que o programa seja
                                            concluído. */
  p->valor = val;
  p->prox  = NULL;
  return(p);
}

bool ListaVazia(Lista *p)
{
  if (p == NULL)
    return(true);
  else
    return(false);
}

void DestroiLista(Lista **p)
{
  No *q;

  q = *p;
  while (!ListaVazia(q)){
    q = (*p)->prox;
    free(*p);
    *p = q;
  }
}

void InsereInicioLista(Lista **p, int val)
{
  No *q=NULL;

  q = CriaNo(val);
  if (ListaVazia(*p)){ /* lista vazia */
    *p = q;
  }else {
    q->prox = *p;
    *p      = q;
  }
}
```

```
bool RemoveInicioLista(Lista **p, int *val)
{
  Lista *q;

  if (!ListaVazia(*p)) { /* a lista não está vazia */
    *val = (*p)->valor;
    q    = (*p)->prox;
    free(*p);
    *p  = q;
    return(true);
  }else
    return(false);
}

void ImprimeLista(Lista *p)
{
  while (p != NULL){
    printf("%d ",p->valor);
    p = p->prox; /* estamos manipulando com cópia local de p, portanto
                    não estamos alterando seu conteúdo fora da
                    função. */
  }
  printf("\n");
}

int main()
{
  Lista *p=NULL; /* inicia vazia */
  int val;

  InsereInicioLista(&p,10);
  ImprimeLista(p);
  InsereInicioLista(&p,20);
  ImprimeLista(p);
  if (RemoveInicioLista(&p,&val))
      printf("%d foi removido\n",val);
  ImprimeLista(p);
  InsereInicioLista(&p,30);
  ImprimeLista(p);
  DestroiLista(&p);

  return 0;
}
```

Uma lista com inserção e remoção sempre em uma mesma extremidade, seja no início ou no final, é denominada **pilha** e é muito usada em vários algoritmos. Outra estrutura bastante útil é a **fila**. Neste caso, a inserção pode ser no início com remoção no final (ou a inserção pode ser no final e a remoção ser no início). Isto é, na fila, o primeiro elemento a ser inserido é o primeiro a ser removido (**FIFO**-*first-in-first-out*) e na pilha, o último a ser inserido é o primeiro a ser removido (**LIFO**-*last-in-first-out*).

2 Exercício

Complete o programa acima criando funções para inserir e remover no final da lista; para consultar valores na lista; inserir e remover antes/após um dado elemento da lista; e para concatenar duas listas.