UNIVERSIDADE SÃO JUDAS TADEU

ESTRUTURA DE DADOS - PROF. H. Senger

IMPLEMENTAÇÃO DE LISTAS COM VETORES

A implementação de listas utilizando vetores é simples. Existe apenas uma pequena questão, com relação ao tamanho da lista. Como sabemos, um vetor possui algumas características, como :

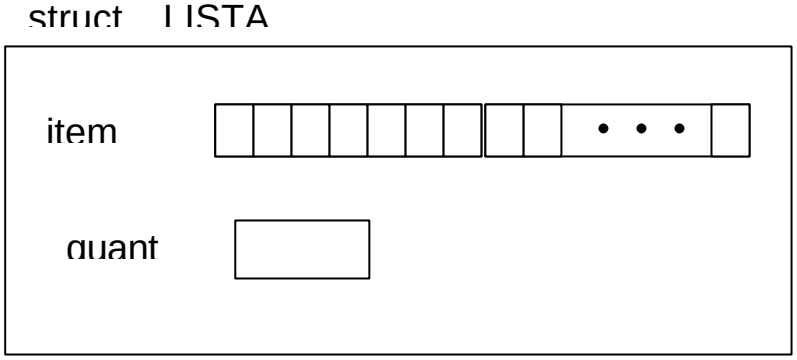
- seu tamanho deve ser declarado na fase de compilação do programa
- o tamanho fixo, e não pode ser alterado durante a execução do programa

Apesar disso, a quantidade de elementos de uma lista pode variar, pois podemos acrescentar novos elementos a uma lista ou mesmo retirar elementos que já existam.

Solução : Para resolver essa questão, pode-se usar as medidas a seguir :

- declarar um vetor grande o bastante, de modo que (provavelmente) nunca venha a faltar posições para o armazenamento de novos elementos; e
- ter uma variável que controla a quantidade "efetiva" de casas ocupadas do vetor

Em termos práticos, teremos a seguinte estrutura de dados :



pode-se declarar o tamanho máximo do vetor :

```
#define     TAM   10
```

declarar o tipo de cada item da lista

```
typedef     char TIPO_ITEM;
```

e finalmente, a estrutura de dados completa

```
typedef struct {
            TIPO_ITEM item[TAM];
            int quant;
        }TIPO_LISTA;
```

Para declarar uma variável lista, pode-se proceder da seguinte forma :

```
TIPO_LISTA      L;
```

isto é possível porque **TIPO_LISTA** foi declarado por meio de um **typedef** , o que faz com que este seja reconhecido como um identificador de tipo, assim como **int, float** ou **char.**

IMPLEMENTAÇÃO :

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

#define     TAM   10     // Quantidade m'axima de itens
#define     FALHA       0
#define     SUCESSO   1

typedef     char TIPO_ITEM;   // define o tipo dos itens da lista

typedef struct {     // cria um identificador de tipo para definir uma lista
            TIPO_ITEM item[TAM];
            int quant;
      }TIPO_LISTA;
/*---------------------------------------------------------------------
            VARIAVEIS GLOBAIS DO PROGRAMA
---------------------------------------------------------------------*/
TIPO_LISTA L;

/*---------------------------------------------------------------------
            PROTOTIPOS DE FUNCOES
---------------------------------------------------------------------*/
void   Inicializa ( TIPO_LISTA *p_L);
int    Insere       ( TIPO_LISTA *p_L, TIPO_ITEM dado, int posicao );
int    Remove       ( TIPO_LISTA *p_L, TIPO_ITEM * p_dado, int posicao );
void   Exibe        ( TIPO_LISTA L );

main ()
{
      TIPO_ITEM x;

      clrscr();
      Inicializa (&L);

      /* a funcao main contem apenas uma sequencia de operacoes realizadas
      sobre a lista L, com a finalidade de testar a correteza das
      implementacoes das funcoes de manipulacao de listas */
```

```
        if (Insere (&L,'A',1))          Exibe (L);
        else printf("Falha\n");
        if (Insere (&L,'B',2))          Exibe (L);
        else printf("Falha\n");
        if (Insere (&L,'C',3))          Exibe (L);
        else printf("Falha\n");
        if (Insere (&L,'X',1))          Exibe (L);
        else printf("Falha\n");
        if (Insere (&L,'Y',3))          Exibe (L);
        else printf("Falha\n");
        if (Insere (&L,'Z',6))          Exibe (L);
        else printf("Falha\n");
        if (Remove (&L,&x,2)) {
                printf("Retirou   dado = %c \n",x);
                Exibe (L);
        }
        else printf("Falha\n");
        if (Remove (&L,&x,5)) {
                printf("Retirou  dado = %c \n",x);
                Exibe (L);
        }
        else printf("Falha\n");
        if (Remove (&L,&x,5)) {
                printf("Retirou  dado = %c \n",x);
                Exibe (L);
        }
        else printf("Falha\n");
        if (Remove (&L,&x,1)) {
                printf("Retirou  dado = %c \n",x);
                Exibe (L);
        }
        else printf("Falha\n");
        return 0;
}

/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Funcao              : Inicializa
Finalidade          : esvaziar uma lista
Entrada             : *p_L -> endereco da lista
Retorno             : nenhum
Observacoes         : zera o valor do campo 'quant' da lista
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
void   Inicializa ( TIPO_LISTA *p_L)
{
        (*p_L).quant=0;
}
```

```
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Funcao              : Insere
Finalidade          : incluir um novo valor em uma determinada posicao da lista
Entrada             :*p_L  -> endereco da lista a ser alterada
                      dado         -> dado a ser acrescentado
                      posicao      -> indica a posicao que o novo dado ira ocupar na lista
Retorno             : 'SUCESSO' ou 'FALHA'
Observacoes         : altera os campos 'item' e 'quant' da lista
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
int     Insere      ( TIPO_LISTA *p_L, TIPO_ITEM dado, int posicao )
{
        int i;
        if ((*p_L).quant == TAM ) return FALHA;       // se a lista ja estiver cheia ...
        else
                if ((posicao < 1) || (posicao > (*p_L).quant + 1)) return FALHA;
                else {
                        for (i=(*p_L).quant; i >= posicao; i--)
                                (*p_L).item[i] = (*p_L).item[i-1];
                        (*p_L).item[posicao-1] = dado;
                        (*p_L).quant ++;
                        return SUCESSO;
                }
}

/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Funcao              : Remove
Finalidade          : retirar o valor de uma determinada posicao dentro da lista
Entrada             :*p_L  -> endereco da lista
                     *p_dado      -> endereco da variavel que ira receber o valor
                                     que foi retirado da lista
                     posicao      -> indica a posicao de onde deve ser retirado o dado
Retorno             : 'SUCESSO' ou 'FALHA'
Observacoes         : altera os campos 'item' e 'quant' da lista
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
int     Remove      ( TIPO_LISTA *p_L, TIPO_ITEM *p_dado, int posicao )
{
        int i;
        if ((*p_L).quant == 0 ) return FALHA;
        else
                if ((posicao < 1) || (posicao > (*p_L).quant)) return FALHA;
                else {
                        *p_dado = (*p_L).item[posicao-1];
                        for (i=posicao - 1; i < (*p_L).quant - 1; i++)
                                (*p_L).item[i] = (*p_L).item[i+1];
                        (*p_L).quant --;
                        return SUCESSO;
```

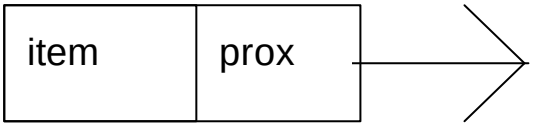
```
                }
}

/* +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Funcao          : Exibe
Finalidade      : exibir o conteudo de uma lista
Entrada         :L      -> lista  a ser exibida
Retorno         : nenhum
Observacoes     :
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
void   Exibe         ( TIPO_LISTA L )
{
        int i;
        printf("Quant : %d    Conteudo : ",L.quant);
        for(i=0; i<L.quant; i++)
                printf("  %c",L.item[i]);
        putchar('\n');
}
```

-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-

Veja a seguir, a saída gerada por esse programa :

LISTAS LIGADAS

É possível criar uma estrutura tal, capaz de armazenar um item de dado, e ainda, conter um endereço que indica o próximo elemento na seqüência.

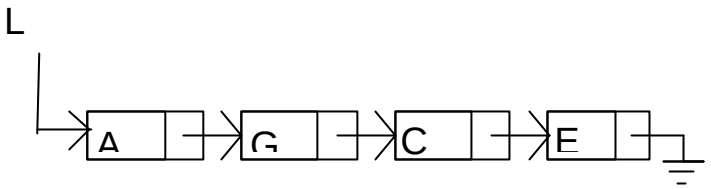



```
typedef       char TIPO_ITEM;
```

Declaração de um nó da lista :

```
struct NO {
            TIPO_ITEM item;
            NO *prox;
      };
```

Assim, pode-se criar um encadeamento de nós com tal estrutura, onde cada nó contém um item da lista :



A variável L é um ponterio cuja função é determinar o início da lista, podendo ser declarada da seguinte forma :

```
typedef struct NO  *PONT;

PONT L;
```

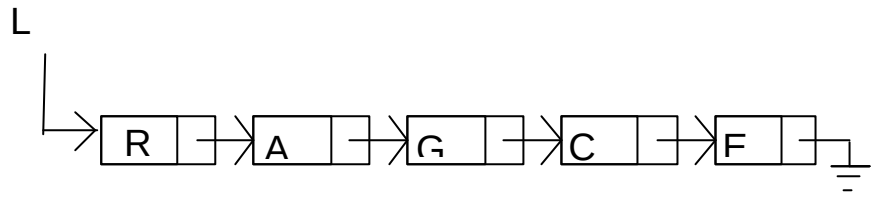
Essa forma de declaração torna-se bastante útil na hora de implementar as funções de manipulação da lista.

A criação de nós para o armazenamento de dados é feita da seguinte forma :

```
p = (PONT) malloc(sizeof(struct NO));
p->item = 'R';
p->prox = L;
```

```
L = p;
```

Assim, ter-se-ia o seguinte resultado :



Vejamos um programa exemplo:

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

#define     FALHA     0
#define     SUCESSO   1

typedef     char TIPO_ITEM;   // define o tipo dos itens da lista

struct NO {
            TIPO_ITEM item;
            NO *prox;
      };

typedef struct NO  *PONT; // cria o tipo ponteiro de NO

/*--------------------------------------------------------------------------
            VARIAVEIS GLOBAIS DO PROGRAMA
--------------------------------------------------------------------------*/
PONT L;     // ponteiro para o primeiro no de uma lista
            // Obs: vale NULL, se a lista nao tiver nenhum no

/*--------------------------------------------------------------------------
            PROTOTIPOS DE FUNCOES
--------------------------------------------------------------------------*/
void  Inicializa ( PONT *p_L);
int   Insere ( PONT *p_L, TIPO_ITEM dado );
int   Remove    ( PONT *p_L, TIPO_ITEM * p_dado );
void  Exibe     ( PONT L );

main ()
{
      TIPO_ITEM x;
```

```
        clrscr();
        Inicializa (&L);

        /* a funcao main contem apenas uma sequencia de operacoes realizadas
        sobre a lista L, com a finalidade de testar a correteza das
        implementacoes das funcoes de manipulacao de listas */

        if (Insere (&L,'A'))   Exibe (L);
        else printf("Falha\n");
        if (Insere (&L,'B'))   Exibe (L);
        else printf("Falha\n");
        if (Insere (&L,'C'))   Exibe (L);
        else printf("Falha\n");
        if (Insere (&L,'X'))   Exibe (L);
        else printf("Falha\n");
        if (Insere (&L,'Y'))   Exibe (L);
        else printf("Falha\n");
        if (Insere (&L,'Z'))   Exibe (L);
        else printf("Falha\n");
        if (Remove (&L,&x)) {
                printf("Retirou   dado = %c \n",x);
                Exibe (L);
        }
        else printf("Falha\n");
        if (Remove (&L,&x)) {
                printf("Retirou  dado = %c \n",x);
                Exibe (L);
        }
        else printf("Falha\n");
        if (Remove (&L,&x)) {
                printf("Retirou  dado = %c \n",x);
                Exibe (L);
        }
        else printf("Falha\n");
        if (Remove (&L,&x)) {
                printf("Retirou  dado = %c \n",x);
                Exibe (L);
        }
        else printf("Falha\n");
        return 0;
}
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Funcao              : Inicializa
Finalidade   : esvaziar uma lista
Entrada             : *p_L -> endereco da lista
Retorno             : nenhum
```

```
Observacoes        : zera o valor do campo 'quant' da lista
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

void   Inicializa ( PONT *p_L)
{
        *p_L = NULL;
}

/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Funcao             : Insere
Finalidade   : incluir um novo valor no inicio da lista
Entrada      :       *p_L   -> endereco do primeiro no da lista
                     dado  -> dado a ser acrescentado
Retorno      :       'SUCESSO' ou 'FALHA'
Observacoes        :
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
int     Insere         ( PONT *p_L, TIPO_ITEM dado )
{
        PONT aux;
        if (((aux = (PONT) malloc(sizeof(struct NO)))==NULL)) return FALHA; // se a lista ja
estiver cheia ...
        else {
                aux->item = dado;
                aux->prox = *p_L;
                *p_L = aux;
                return SUCESSO;
        }
}
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Funcao             : Remove
Finalidade   : retirar o primeiro dado da lista
Entrada      :       *p_L   -> endereco da lista
                     *p_dado       -> endereco da variavel que ira receber o valor
                              que foi retirado da lista
Retorno            : 'SUCESSO' ou 'FALHA'
Observacoes        :
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
int     Remove      ( PONT *p_L, TIPO_ITEM *p_dado )
{
        PONT aux;
        if ((*p_L)==NULL) return FALHA;
        else {
                aux = *p_L;
                *p_L = aux->prox;
                *p_dado = aux->item;
                free (aux);
                return SUCESSO;
```

```
        }
}

/* +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Funcao              : Exibe
Finalidade   : exibir o conteudo de uma lista
Entrada      :      L       -> lista  a ser exibida
Retorno             : nenhum
Observacoes         :
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
void    Exibe       ( PONT L )
{
        int i;
        printf("Conteudo : ");
        while ( L != NULL ){
                printf("  %c",L->item);
                L = L->prox;
        }
        putchar('\n');
}


-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x
```

Tal programa gera a seguinte saída em tela :