Vitor Amadeu Souza

vitor@cerne-tec.com.br

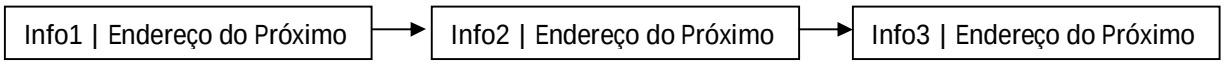
www.cerne-tec.com.br

Lista Encadeada e Duplamente Encadeada

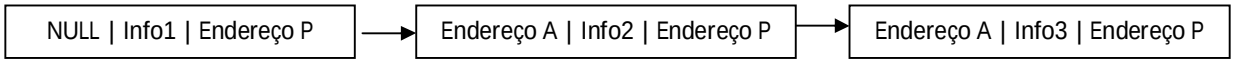
Para armazenarmos diversos dados na linguagem C por exemplo, podemos criar um vetor como o apresentado a seguir:

```
int dado[100];
```

O problema deste tipo de solução é que ele ocupa um espaço físico na memória, que não pode ser menor caso a quantidade de informações a serem guardadas seja menor nem pode ser maior, pois caso isso ocorra irá invadir outras áreas de memórias o que poderá ocasionar sérios problemas em um programa. Para resolver isso, existem as listas encadeadas, onde cada informação é alocada na memória em tempo de execução e caso seja necessário guardar um novo dado, uma nova região é utilizada para isso e caso algum dado seja retirado, uma região da memória pode ser liberada. Podemos representar as listas como na figura a seguir, note que em cada posição temos o endereço do próximo registro e a última informação fica com o endereço NULL que é 0.



A grande vantagem do uso deste método é o fato dele não ocupar uma região fixa de memória, o que permite com que sejam alocadas ou não novas regiões de memória de acordo com a necessidade. Note que de acordo com a figura acima, sempre que precisamos acessar a próxima informação, devemos saber o endereço da mesma através da informação anterior. Isto pode atrasar o processamento do processador em determinadas aplicações pela forma explicada. Existe neste caso as listas duplamente encadeadas, onde em cada registro há o endereço posterior e anterior a ele. Vejamos como ficará a organização da memória neste caso:



Desta forma ocupamos mais memória porém ganhamos tempo na inserção e remoção de algum dado da memória além de poder neste caso percorrer todos os dados tanto no sentido crescente como decrescente.

Exemplo

```
#include <stdio.h>                  //Include do sistema
#include <string.h>                 //Include do sistema
#include <stdlib.h>

struct tipo_aluno
{
       int codigo;

};



struct tipo_lista_encadeada
{
       struct tipo_aluno aluno;
       struct tipo_lista_encadeada *prox;

};



typedef struct tipo_lista_encadeada LISTA;
typedef struct tipo_aluno ACADEMICO;

LISTA *primeiro=NULL;
LISTA *ultimo;

main()
{

  LISTA *novo;

  novo = (LISTA *)malloc(1*sizeof(LISTA));

  if (!novo)
  {

     printf("\nNao existe espaco na memoria!");
     return;
  }

  novo->aluno.codigo=10;
  novo->prox = NULL;

}
```