Aula 2 – Listas Encadeadas

Alocação de Espaço de Memória Dinamicamente

Em programação é possível implementar dois tipos de alocação: estática ou dinâmica, cada uma com suas particularidades, propriedades e comandos.

- Alocação Estática ocorre em tempo de compilação, ou seja, no momento em que se define uma variável ou estrutura é necessário que se definam seu tipo e seu tamanho, os vetores (Figura 1) e matrizes (Figura 2), por exemplo.

Figura 1 - Lista Estática Homogênea

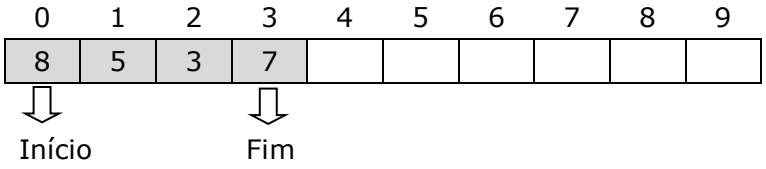
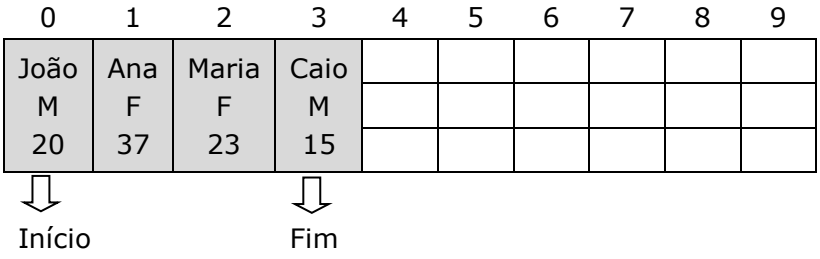


Figura 2 - Lista Estática Heterogênea



- Alocação Dinâmica ocorre em tempo de execução, ou seja, as variáveis e estruturas são declaradas sem a necessidade de se definir seu tamanho, pois nenhuma memória será reservada ao colocar o programa em execução.

Durante a execução do programa, no momento em que uma variável ou parte de uma estrutura precisa ser utilizada, sua memória será reservada e, no momento em que não foi mais necessária, deverá ser liberada. Isso é feito com o auxílio de comandos ou funções que permitem, por meio do programa, reservar e liberar memória.

A função básica para alocar memória no C é `malloc`. Ele recebe com parâmetro o número de bytes que se deseja e retorna o endereço inicial da área de memória alocada.

Podemos implementar dois tipos de Listas Dinâmicas Encadeadas: homogênea (Figura 3) e heterogênea (Figura 4).

Figura 3 - Lista Dinâmica Homogênea

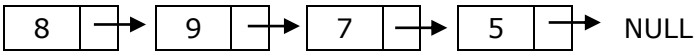
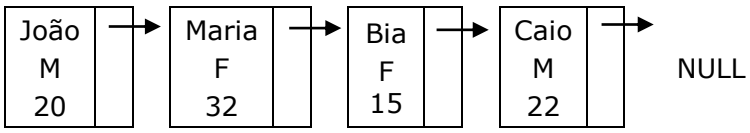


Figura 4 - Lista Dinâmica Heterogênea



Listas

Uma lista é um conjunto de dados ordenados e de número variável de elementos. Existem dois tipos:

- **Lista Sequencial**: cada elemento da lista ocupa posição sucessiva ao elemento anterior, e é percorrida através do seu índice ou indexador (vetores), vide Figura 1;
- **Lista Encadeada**: a ordem dos elementos da lista é dada pelos apontadores (ponteiros, elos ou links), vide Figura 3.

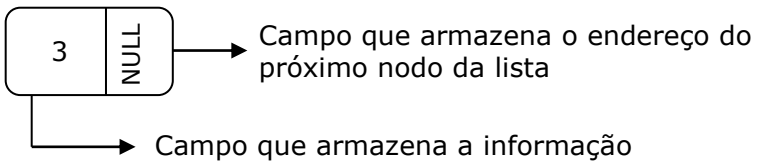
Listas Encadeadas

O uso de apontadores é uma forma diferenciada de alocar espaço em memória denominada alocação dinâmica de memória. Nesse caso, a alocação é realizada durante a execução do programa, sempre que necessário. A liberação de memória também é dinâmica, ou seja, espaços que não são mais necessários no programa podem ser liberados antes de seu término.

O elemento básico de uma lista encadeada é o nodo. Cada nodo é composto por uma parte que armazena dados e outra que armazena campos de ligação.

O campo de ligação (links, ponteiros ou elos) de uma Lista Encadeada possui o endereço de memória onde o próximo nodo está armazenado, permitindo assim o encadeamento dos dados e possibilitando a implementação de Listas Encadeadas (Figura 5).

Figura 5 – Nodo de uma Lista Encadeada



Uma vez que a memória necessária para armazenar cada elemento será reservada durante a execução do programa (alocação dinâmica), cada elemento pode ocupar posições aleatórias de memória, não apresentando uma sequência física entre os elementos.

Declaração de uma Lista Simplesmente Encadeada

```
struct nodo{
 int dados; //dado inteiro
 struct nodo *prox; //ponteiro elo para o próximo nodo
 };
```

Alocação de Memória

A alocação de memória ocorre da seguinte forma:

- No momento em que o programa necessita de um espaço para armazenar determinado valor, este é solicitado por meio de um comando especial (em C `malloc`). A alocação desse espaço ocorre neste momento, e ele fica disponível para ser utilizado no programa;

- Quando determinado espaço alocado dinamicamente não for mais necessário, este pode ser liberado mediante outro comando especial (`free` no C). Este comando libera o endereço de memória indicado para a LED (Lista de Espaços Disponíveis);

- O mesmo espaço de memória pode ser alocado novamente, porém é como se outra variável fosse alocada. A escolha da área a ser utilizada é feita pelo gerenciador de memória, não pelo programa de aplicação;

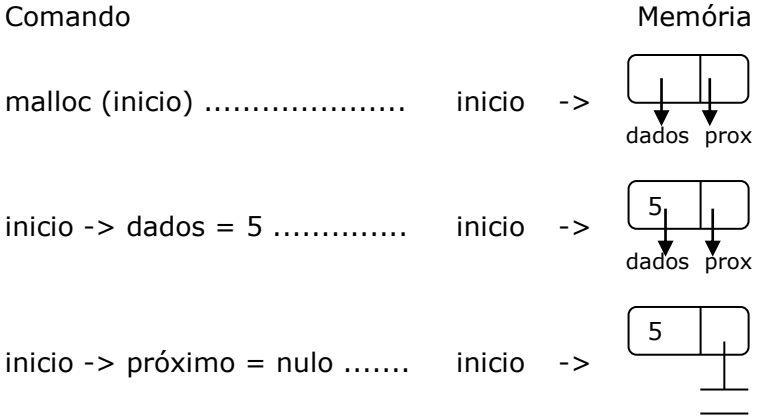
- Ao fazer a solicitação de espaço, o programa deve especificar seu tamanho e o seu tipo. Isso é feito por meio de variáveis do tipo apontador.

Manipulação da Estrutura

A Figura 6 mostra a situação na memória depois da execução de cada comando.

- Inicialmente a variável inicio não contém um endereço válido de memória (pode-se dizer que ela contém lixo);
- Depois da execução do comando `malloc` (inicio) um novo nodo é criado e a variável inicio recebe o endereço desse nodo, ou seja, aponta para o nodo;
- O comando `inicio->dados=5` atribui o valor 5 ao campo dados do nodo apontado por inicio;
- E o comando `inicio->próximo=nulo` atribui o valor nulo (endereço nulo de memória) ao campo próximo do nodo criado.

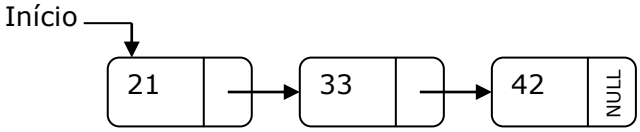
Figura 6 – Execução dos comandos na memória



Listas Simplesmente Encadeadas

As listas simplesmente encadeadas utilizam nodos com apenas um campo de ligação (Figura 7). Os nodos são formados por registros que possui, pelo menos, dois campos: a informação e o endereço de memória onde está armazenado o próximo elemento da lista.

Figura 7 – Lista Simplesmente Encadeada



Criação de Lista Simplesmente Encadeada

A criação da LSE pode ser implementada por duas funções padrões: InsereDireita e InsereEsquerda.

Função Insere a Direita em uma LSE

```
int InsereDireita(struct nodo **inicio,int v){
  struct nodo *novo,*aux;
  novo=(struct nodo *)malloc(sizeof(struct nodo));
  if(novo!=NULL){
    novo->dados=v;
    novo->prox=NULL;
    if(*inicio==NULL) *inicio=novo;
     else{
       aux=*inicio;
       while(aux->prox!=NULL)aux=aux->prox;
       aux->prox=novo;
       }
    }
 }

//Esqueleto da Função Principal
  int main(){
  struct nodo *ptri=NULL;
  int valor,i;
   for(i=0;i<TAM;i++){
     valor=rand()%100;
     InsereEsquerda(&ptri,valor);
     //InsereDireita(&ptri,valor);
     }

   //imprime lista
   printf("\n\n--- Lista Simplesmente Encadeada ---\n\n");
   ImprimeLSE(&ptri);
  }
```