typedef struct no {

int RA;

char nome[20];

} no;

struct no \*A, \*B;

A declaração acima reconhece a existência do Ponteiro na memória, mas não aloca endereço de memória ao mesmo.

Observe que na declaração do tipo Ponteiro é especificado o tipo de estrutura que pode ser apontada por uma variável Ponteiro. No exemplo acima, uma variável do tipo Ponteiro pode ser referenciada ou ser associada para a estrutura tipo Nó.

A declaração das variáveis A e B não é inicializada, isto é, nenhum valor é armazenado em A e B e na memória existe ainda nenhuma estrutura do tipo Nó.

Para estabelecer uma variável dinâmica na memória, é preciso uma certa quantidade de espaço nela. No exemplo acima é necessário o número de Bytes que acomode o tipo Nó. A alocação desse espaço e a sua localização na memória são feitas pela execução de um procedimento chamado Malloc.

O procedimento Malloc tem como parâmetro uma variável do tipo Ponteiro e, quando ativado, aloca espaço na memória de acordo com a declaração do tipo Ponteiro dessa variável.

A = (no\*)malloc(sizeof(no))

Esta semântica permite alocar um espaço de memória para uma estrutura do tipo Nó e faz com que o Ponteiro aponte para um endereço de memória permitindo o carregamento de dados.

A estrutura dinâmica não é inicializada, pois nada é armazenado quando em Nó. A variável A contém o endereço de memória e pode ser visualizada via ‘printf’.

Para acessar a estrutura dinâmica escrevemos o nome da variável Ponteiro, seguido do símbolo ‘\*’.

No exemplo, \*A e \*B identifica a estrutura estabelecida que é chamada de variável dinâmica.

Como a estrutura do tipo Nó é um Record, o acesso a cada campo terá que ser feito através do símbolo ‘->’ seguido do identificador do campo. Exemplo:

A -> RA;

A -> nome;

As variáveis Ponteiro podem ser operacionalizadas da mesma forma que as variáveis estáticas, como veremos a seguir.

scanf(“%s”, &A -> nome);

scanf(“%d”, &A -> RA):

if(A -> RA > 0) {

printf(“O nome da pessoa é %s”, A -> nome)

}

As variáveis Ponteiro permitem o comando de atribuição, entre elas, sendo do mesmo tipo, sinais como operações relacionadas ‘=’, ‘<’, ‘>’ também permitem o seu uso.

Há uma constante predefinida para o tipo Ponteiro que é identificado como ‘NULL’. O valor ‘NULL’ não é nenhum endereço válido e é compatível com qualquer variável Ponteiro.