ESTRUTURA DE DADOS II

**22/08/2023**

**Alocação Dinâmica de Memória**

Durante a declaração das variáveis é feita a reserva de memória para cada uma das variáveis declaradas, sendo em quantidade de bytes, de acordo com o tipo de cada variável.

LINGUAGEM C:

* int = 2 bytes p/ número inteiro
* float = 4 bytes p/ número real
* char = 1 byte p/ cada palavra alocada

Essa alocação de memória é **ESTÁTICA**, ou seja, não se altera em tempo de execução.

Algumas aplicações precisam alocar memória em tempo de execução. A Linguagem C oferece a função “*Malloc*” para que se faça alocação de memória durante a execução do programa, ou seja, **ALOCAÇÃO DINÂMICA DE MEMÓRIA.**

Da mesma forma que é necessário alocar memória para armazenar informação é necessário **LIBERÁ-LA** em tempo de execução, quando alocada dinamicamente.

A função *“Free”* libera a memória apontada pelo ponteiro. O tamanho de memória liberado é o mesmo que foi alocado pela função *“Malloc”.*

As funções Malloc() e Free() pertencem a biblioteca stdlib.h.

**Exemplo utilizando as duas funções:**

*int \*p, \*q;*

*p = (int \*) malloc (sizeof(int));*

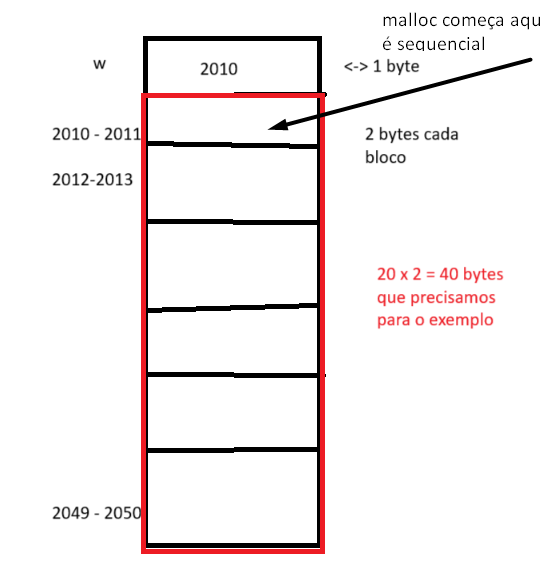
*q = (int \*) malloc (sizeof(int));*

*\*p = 5;*

*\*q = 10;*

*Free(p);*

*P=q;*



*“função malloc explicada com desenho de memória”*

*Malloc () = significa que a função irá alocar "size" bytes de memória e retornará um ponteiro para o início da memória alocada.*

**EXEMPLO:**

*Int \* w;*

*W = (int \*) malloc (20 \* sizeof(int));*

*“Durante a execução do programa você vai alocar 20x o valor de um inteiro”*

*Ponteiro = é uma variável tipada que armazena o endereço de memória de outra variável. O “\*” indica a declaração do ponteiro.*

*Free() = função que libera a memória apontada pelo ponteiro deixando o programa mais leve.*

**EXEMPLO:**

*int \*p;*

*float \*q;*

*p = (int \*) malloc(25\*sizeof(int));*

*q = (float \*) malloc(sizeof(float));*

*free(p); => Liberou os 50 bytes que ‘p’ aponta.*

*free(q); => Liberou os 4 bytes que ‘q’ aponta.*

**29/08/2023**