Alocação Dinâmica de Memória

- realloc

A função **realloc()** serve para realocar memória e tem a seguinte forma:

```
void *realloc (void *ptr, unsigned int num);
```

A função modifica o tamanho da memória previamente alocada apontada por ***ptr** para aquele especificado por **num**.

Alocação Dinâmica de Memória

- realloc (continuação)

O valor de **num** pode ser maior ou menor que o original. Um ponteiro para o bloco é devolvido porque **realloc()** pode precisar mover o bloco para aumentar seu tamanho. Se isso ocorrer, o conteúdo do bloco antigo é copiado no novo bloco, e nenhuma informação é perdida. Se **ptr** for nulo, aloca **size** bytes e devolve um ponteiro; se **size** é zero, a memória apontada por **ptr** é liberada. Se não houver memória suficiente para a alocação, um ponteiro nulo é devolvido e o bloco original é deixado inalterado.

```
#include <stdio.h>
#include <stdlib.h>
main (void)
{
    int *p, a, i;
    ...
    a = 30;
    p=(int *)malloc(a*sizeof(int));
    if (!p)
    {
        printf ("** Erro: Memoria Insuficiente **");
        exit (1);
     }
```

```
    for (i=0; i<a ; i++)
      p[i] = i*i;
    a = 100;
    p = (int *)realloc (p, a*sizeof(int));
    if (!p) { printf (“\nERRO!\n”); exit (1); }
    for (i=30; i<a ; i++)
      p[i] = a*i*(i-6);
    ...
}
```

Alocação Dinâmica de Memória

- free

Quando alocamos memória dinamicamente é necessário que nós a liberemos quando ela não for mais necessária. Para isto existe a função **free()** cuja forma é:

```
void free (void *p);
```

```
#include <stdio.h>
#include <stdlib.h>
main (void)
{
    int *p, a;
    ...
    p=(int *)malloc(a*sizeof(int));
    if (!p)
    {
      printf ("** Erro: Memoria Insuficiente **");
      exit (1);
     }
    ...
    free(p);
}
```

Alocação Dinâmica de Memória - Exercício

Escreva um programa em C que manipule um vetor de inteiros não nulos alocado dinamicamente. O programa recebe inteiros, através da entrada padrão, e os insere no vetor. A cada inteiro que é inserido a área de memória necessária para armazenar um inteiro é incrementada ao número de bytes necessários para armazenar o vetor. O vetor não ocupa memória inicialmente. Quando o usuário entrar com o inteiro 0 (zero), o programa será finalizado e o mesmo não pertencerá ao vetor. Após o processo de inserção o vetor deve ser impresso na saída padrão. Libere a memória utilizada antes do final do processamento.

Alocação Dinâmica de Memória - Exercício

Com base no que vimos, construa um programa que aloque dinamicamente memória para um vetor de strings. O processamento se dará da seguinte forma: o usuário fornecerá através da entrada padrão um conjunto de strings com tamanhos aleatórios. O final de uma string é identificado pelo pressionamento da tecla enter e o final do conjunto de strings é identificado pelo fornecimento de uma string vazia pelo usuário. Ao final do processamento o programa deve retornar na saída padrão as strings contidas no vetor.

Alocação Dinâmica de Memória - Exercício

Com o que foi visto sobre alocação dinâmica de memória, defina um TAD para representar uma matriz triangular inferior de reais, o qual contempla as operações de criação de uma matriz, atribuição, e consulta de um determinado elemento da matriz. Implemente o TAD em questão na linguagem C.