Alocação Dinâmica

- A alocação dinâmica permite ao programador criar variáveis em tempo de execução, ou seja, alocar memória para novas variáveis quando o programa está sendo executado.
- O padrão C ANSI define apenas 4 funções para o sistema de alocação dinâmica, disponíveis na biblioteca stdlib.h:

  malloc
  calloc
  realloc
  free

Alocação Dinâmica

malloc

- A função malloc() serve para alocar memória e tem o seguinte protótipo:

  *void *malloc (unsigned int num);*

- A função toma o número de bytes que queremos alocar (num), aloca na memória e retorna um ponteiro **void *** para o primeiro byte alocado. O ponteiro **void *** pode ser atribuído a qualquer tipo de ponteiro. Se não houver memória suficiente para alocar a memória requisitada a função malloc() retorna um ponteiro nulo.

Alocação Dinâmica

Exemplo

O fragmento de código mostrado aqui aloca 1000 bytes de memória livre.

```
char *p;
p = malloc(1000);
```

O próximo exemplo aloca espaço para 50 inteiros:

```
int *p;
p = malloc(50*sizeof(int));
```

Obs.: O operador **sizeof**() retorna o número de bytes de um inteiro.

Alocação Dinâmica

```
#include <stdio.h>
#include <stdlib.h>
main (void)
{
   int *p;
   int a;
   ... /* Determina o valor de a em algum lugar */
   p = (int *)malloc(a*sizeof(int));
   if (!p){
      printf ("** Erro: Memoria Insuficiente **");
      exit;
   }
   ...
}
```

Alocação Dinâmica

calloc

- A função calloc() também serve para alocar memória, mas possui um protótipo um pouco diferente:

  *void *calloc (unsigned int num, unsigned int size);*

- A funçao aloca uma quantidade de memória igual a num×size, isto é, aloca memória suficiente para uma matriz de num objetos de tamanho size. Retorna um ponteiro void * para o primeiro byte alocado. O ponteiro void * pode ser atribuído a qualquer tipo de ponteiro. Se não houver memória suficiente para alocar a memória requisitada a função calloc() retorna um ponteiro nulo.

Alocação Dinâmica

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

Alocação Dinâmica

realloc

A função realloc() serve para realocar memória e tem o seguinte protótipo:

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

A funçao modifica o tamanho da memória previamente alocada apontada por *ptr para aquele especificado por *num*. O valor de *num* pode ser maior ou menor que o original.

Alocação Dinâmica

realloc

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

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.

Alocação Dinâmica

free

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

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

- Basta então passar para free() o ponteiro que aponta para o início da memória alocada.
- Mas você pode se perguntar: como é que o programa vai saber quantos bytes devem ser liberados? Ele sabe pois quando você alocou a memória, ele guardou o número de bytes alocados numa "tabela de alocação" interna.

Alocação Dinâmica

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

Alocação Dinâmica de Vetores

```
#include <stdio.h>
#include <stdlib.h>
float *Alocar_vetor_real (int n)
{
  float *v;              /* ponteiro para o vetor */
  if (n < 1) {           /* verifica parametros recebidos */
     printf ("** Erro: Parametro invalido **\n");
     return (NULL);
     }
  /* aloca o vetor */
  v = (float *) calloc (n+1, sizeof(float));
  if (v == NULL) {
     printf ("** Erro: Memoria Insuficiente **");
     return (NULL);
  }
  return (v);    /* retorna o ponteiro para o vetor */
}
```

Alocação Dinâmica de Vetores

```
float *Liberar_vetor_real (int n, float *v)
{
    if (v == NULL) return (NULL);
    if (n < 1) {                /* verifica parametros recebidos */
        printf ("** Erro: Parametro invalido **\n");
        return (NULL);
    }
    free(v);        /* libera o vetor */
    return (NULL);  /* retorna o ponteiro */
}
```

Alocação Dinâmica de Vetores

```
void main (void)
{
   float *p;
   int a;
   ...    /* outros comandos, inclusive a inicializacao de a */
   p = Alocar_vetor_real (a);
   ...    /* outros comandos, utilizando p[] normalmente */
   p = Liberar_vetor_real (a, p);
}
```

Alocação Dinâmica de Matrizes

A alocação dinâmica de memória para matrizes é realizada da mesma forma que para vetores, com a diferença que teremos um ponteiro apontando para outro ponteiro que aponta para o valor final, o que é denominado indireção múltipla. A indireção múltipla pode ser levada a qualquer dimensão desejada.

Alocação Dinâmica de Matrizes

```
#include <stdio.h>
#include <stdlib.h>
float **Alocar_matriz_real (int m, int n)
{
   float **v;  /* ponteiro para a matriz */
   int   i;    /* variavel auxiliar      */
   if (m < 1 || n < 1) {                /* verifica parametros recebidos */
      printf ("** Erro: Parametro invalido **\n");
      return (NULL);
   }
   /* aloca as linhas da matriz */
   v = (float **) calloc (m, sizeof(float *));
   if (v == NULL) {
      printf ("** Erro: Memoria Insuficiente **");
      return (NULL);
   }
```

Alocação Dinâmica de Matrizes

```
/* aloca as colunas da matriz */
  for ( i = 0; i < m; i++ ) {
        v[i] = (float*) calloc (n, sizeof(float));
        if (v[i] == NULL) {
            printf ("** Erro: Memoria Insuficiente **");
            return (NULL);
        }
  }
  return (v); /* retorna o ponteiro para a matriz */
}
```

Alocação Dinâmica de Matrizes

```
float **Liberar_matriz_real (int m, int n, float **v)
{
  int  i;  /* variavel auxiliar */
  if (v == NULL)   return (NULL);
  if (m < 1 || n < 1) {              /* verifica parametros recebidos */
     printf ("** Erro: Parametro invalido **\n");
     return (v);
  }
  for (i = 0; i < m; i++) free (v[i]);     /* libera as linhas da matriz */
  free (v);      /* libera a matriz */
  return (NULL); /* retorna um ponteiro nulo */
}
```

Alocação Dinâmica de Matrizes

```
void main (void)
{
  float **mat;  /* matriz a ser alocada */
  int   l, c;       /* numero de linhas e colunas da matriz */
  ...          /* outros comandos, inclusive inicializacao para l e c */
  mat = Alocar_matriz_real (l, c);
  ...          /* outros comandos utilizando mat[][] normalmente */
  mat = Liberar_matriz_real (l, c, mat);
  ...
}
```