Profa. Michele Fúlvia Angelo

PGCA 028 – Tópicos Especiais em Tecnologia Computacional I – Introdução à Programação de Computadores

Aula 11

Universidade Estadual de Feira de Santana

Sumário

- Alocação Dinâmica de Memória

Alocação Dinâmica de Memória

- Muitas vezes usamos espaço desnecessário da memória declarando variáveis ou ainda ficamos limitados as variáveis declaradas.

- Alocação dinâmica (durante a execução do programa) de memória permite:

  - Usar somente a memória necessária, diminuindo o desperdício de memória;

  - Não impor limites quanto ao espaço de memória, a não ser físicos.

Alocação Dinâmica de Memória

- Com esta capacidade, podemos definir um vetor ou uma matriz cujo tamanho seja especificado em tempo de execução (pelo usuário, por exemplo).

- Funções da biblioteca <stdlib.h>
  - malloc();
  - calloc();
  - realloc();
  - free();

malloc()

- A função malloc aloca memória;
- malloc - abreviatura de *memory allocation*
- A sintaxe da função malloc() é dada por:

```
void *malloc(size_t size);
```

- Recebe como parâmetro “size” que é o número de bytes de memória que se deseja alocar.
- Retorna um ponteiro do tipo void;
- Se não houver memória suficiente para atender ao que foi requisitado, a função retorna um ponteiro nulo (um ponteiro contendo NULL).

Exemplo: suponha que seja necessário no meio do código alocar uma memória com 150 bytes, para isto seria necessário digitar as seguintes linhas de código:

```
char *str;
str= (char *) malloc(150);
```

malloc()

- sizeof(tipodedados)
  - indica quanto espaço um tipo de dados precisa.
  - sizeof(char), sizeof(int), sizeof(float), sizeof(tipopessoa),…
- conversão de tipo
  - malloc retorna um ponteiro sem tipo, ‘genérico’, e precisamos converter para o tipo necessário.
  - ponteiro = (tipo_de_dado *) malloc(...

free()

- Podemos também liberar dinamicamente a memória usando a função:

  void free(void* ptr);

- Basta passar para free o ponteiro que aponta para o início da memória a ser liberada.

- Evita desperdício, liberando o que não é mais usado.

Exemplo

```
#include <stdio.h>
#include <stdlib.h>

int main() {
 int *ptr;
 ptr = (int *) malloc (n * sizeof (int));
 if (ptr == NULL) {
    printf ("\nErro: nao foi possivel reservar a memoria
necessaria.\n");
 }
 else {
     /*ok – pode fazer o que há para fazer */
 }
}
```

Exemplo

```
#include <stdio.h>
#include <stdlib.h>

int main ( ){
 char *ptr ;
 ptr = (char *) malloc(sizeof(char));
 scanf(”%c ”, ptr) ;
 printf(”%c \n”, *ptr ) ;
 free(ptr) ;
}
```

Exemplo: Alocação Dinâmica de Vetor

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int i,x;
  int *vetor;

  scanf("%d",&x); //tamanho do vetor
  vetor = (int *) malloc( x * sizeof( int ));
  for ( i = 0 ; i < x ; i++ )
   scanf("%d",&vetor[i]);

  free(vetor); //liberando a memoria alocada.

}
```

EXEMPLO: Alocação Dinâmica de Matriz

- 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;

- No exemplo da implementação de alocação dinâmica para matriz real bidimensional, utilizou-se:
  - um vetor de ponteiros (correspondendo ao primeiro índice da matriz), sendo que cada ponteiro aponta para o início de uma linha da matriz.
  - Em cada linha existe um vetor alocado dinamicamente (compondo o segundo índice da matriz).

EXEMPLO: Alocação Dinâmica de Matriz

```
#include <stdio.h>
#include <stdlib.h>
int main()
{ int i,j;
  int x,y;
  int **matriz;

  scanf("%d",&x);
  scanf("%d",&y);

  matriz = (int **) malloc( x*sizeof(int) ); //aloca as linhas da matriz

  //aloca as colunas da matriz
  for ( i = 0 ; i < x ; i++ )
    matriz[i] = (int *) malloc( y*sizeof(int) );

  //lendo as celulas
   for ( i = 0 ; i < x ; i++ )
    for ( j = 0 ; j < y ; j++ )
        scanf("%d",&matriz[i][j]);

  free(matriz);
  return 0;
}
```