Alocação estática de matrizes

Na alocação estática, o tamanho das matrizes é definido durante a compilação, ou seja, o compilador reserva um espaço da memória, o que garante que a matriz tenha um tamanho fixo durante toda a execução do programa. Dentre as vantagens de sua utilização, destaca-se a simplicidade de manipulação, o acesso rápido aos elementos e a ausência de necessidade de gerenciamento da memória.

```c
#include<stdio.h>

int main (){
  //Alocação fixa
  int matriz[3][3];

  for (int i=0; i<3; i++ ){
    for (int j=0; j<3; j++ ){

      printf ("\nElemento[%d][%d] = ", i, j);
      scanf ("%d", &matriz[ i ][ j ]);
    }
  }

  return 0;
}
```

Alocação dinâmica de matrizes

Na alocação dinâmica, o tamanho das matrizes pode ser definido em tempo de execução, ou seja, a memória é alocada na heap, área separada separada da pilha de memória do programa em que objetos e estruturas de dados alocados dinamicamente durante a execução do programa, a partir de funções como malloc() ou calloc(). Além disso, embora esse método evite o desperdício de memória, a mesma deve ser liberada manualmente com a função free(), buscando evitar vazamentos.

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

int main() {
    int linhas = 3, colunas = 3;

    // Alocação dinâmica de um array de ponteiros (linhas)
    int **matriz = (int **)malloc(linhas * sizeof(int *));
    if (matriz != NULL) {
        for (int i = 0; i < linhas; i++) {
        matriz[i] = (int *)malloc(colunas * sizeof(int));
            if (matriz[i] == NULL) {
                printf("Erro ao alocar memória!\n");
                return 1;
            }
        }


        // Preenchendo a matriz
        int valor = 1;
        for (int i = 0; i < linhas; i++) {
            for (int j = 0; j < colunas; j++) {
                matriz[i][j] = valor++;
            }
        }

    }
```

```c
    // Impressão da matriz
    for (int i = 0; i < linhas; i++) {
        for (int j = 0; j < colunas; j++) {
            printf("%d ", matriz[i][j]);
        }
        printf("\n");
    }

    // Liberação da memória
    for (int i = 0; i < linhas; i++) {
        free(matriz[i]);  // Liberando cada linha
    }
    free(matriz);  // Liberando o array de ponteiros

    return 0;
}
```