Laboratorio 9

Programación de Computadores (2025-2)

Temas del Laboratorio 9

- Memoria dinámica
- Uso de punteros en memoria dinámica
- Funciones Calloc y Realloc
- Ejercicios

Memoria dinámica (Heap)

A diferencia de la memoria estática, que se asigna durante la compilación y tiene un tamaño fijo, la memoria dinámica permite a un programa solicitar y liberar memoria según sea necesario durante la ejecución.

La gestión de la memoria dinámica en C se realiza principalmente a través de dos funciones: **malloc()** y **free().**

Malloc recibe el tamaño en bytes del bloque de memoria a asignar y devuelve un puntero genérico (*void**) al inicio del bloque de memoria asignado.

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

Estructura de la función ***malloc()***

```
void free(void *ptr);
```

Estructura de la función ***free()***

Ejemplo de uso de malloc y free

Puntero al tipo char

Tamaño de memoria a asignar (1 byte)

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

int main() {
    // Asignar memoria para un char
    char *p = malloc(sizeof(char));

    // Utilizar la memoria asignada
    *p = 42;

    // Imprimir el valor
    printf("Valor: %d\n", *p);

    // Liberar la memoria cuando ya no sea necesaria
    free(p);

    return 0;
}
```

Uso de punteros en memoria dinámica

El uso de punteros es fundamental al trabajar con memoria dinámica.

Cuando se asigna memoria dinámica para arreglos, se utilizan las propiedades de los punteros para recorrer el contenido del arreglo

```c
int main(int argc, char *argv[]) {

    if (argc != 2) {
        printf("Uso: %s <numero>\n", argv[0]);
        return 1;
    }

    int *p;
    int n = atoi(argv[1]);

    // Asignar memoria para un arreglo de n enteros
    p = malloc(n * sizeof(int));

    // Utilizar la memoria asignada
    for (int i = 0; i < n; i++) {
        p[i] = rand() % 100;
    }

    // Imprimir los valores
    for (int i = 0; i < n; i++) {
        printf("%d ", p[i]);
    }
    printf("\n");

    // Liberar la memoria cuando ya no sea necesaria
    free(p);

    return 0;
}
```

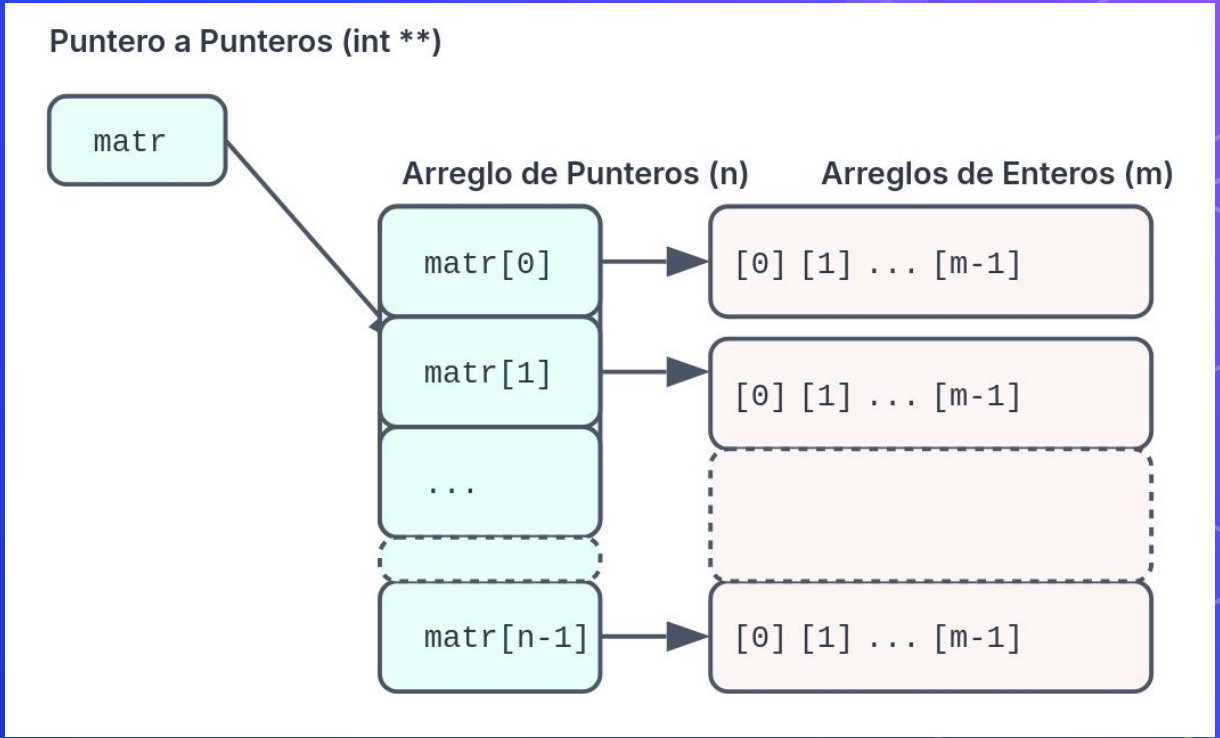
Matriz en memoria dinámica

Ejemplo matriz de n*m

matr es una variable que almacena una dirección de memoria.

Esa dirección apunta al inicio de un bloque de memoria en el heap que contiene n punteros (matr[0], matr[1], etc.). **Uno por cada fila.**

Cada uno de esos n punteros almacena la dirección de memoria de otro bloque en el heap. **Representan las columnas de esa fila.**



Matriz en memoria dinámica

```c
int main() {
    int filas, columnas;

    printf("Ingrese el número de filas: ");
    scanf("%d", &filas);

    printf("Ingrese el número de columnas: ");
    scanf("%d", &columnas);

    // Declarar un puntero para la matriz
    int **matriz;

    // Asignar memoria dinámica para las filas
    matriz = malloc(filas * sizeof(int*));

    // Asignar memoria dinámica para cada columna
    for (int i = 0; i < filas; i++) {
        matriz[i] = malloc(columnas * sizeof(int));
    }
```

```c
    // Inicializar y trabajar con la matriz
    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            matriz[i][j] = i * j;
        }
    }

    // Imprimir la matriz
    printf("Matriz:\n");
    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            printf("%d\t", matriz[i][j]);
        }
        printf("\n");
    }

    // Liberar la memoria asignada
    for (int i = 0; i < filas; i++) {
        free(matriz[i]);
    }
    free(matriz);

    return 0;
}
```

Calloc

La función calloc asigna bloques de memoria de tamaño especificado e inicializa todos los bytes de la memoria asignada a cero. A diferencia de malloc, que no inicializa el contenido de la memoria asignada, calloc garantiza que todos los bits de la memoria asignada se establezcan en cero.

```
void* calloc(size_t num_elements, size_t element_size);
```

Estructura de la función ***calloc()***

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

int main() {

    int *p;

    // Asignar memoria para un array de 5 enteros inicializados a 0
    p = calloc(5, sizeof(int));

    // Utilizar la memoria asignada (por ejemplo, imprimir los valores)
    for (int i = 0; i < 5; i++) {
        printf("%d ", p[i]);
    }

    printf("\n");

    // Liberar la memoria cuando ya no sea necesaria
    free(p);

    return 0;
}
```

Realloc

Realloc se utiliza para cambiar el tamaño de un bloque de memoria previamente asignado, ya sea para aumentar o disminuir su tamaño.

```
void* realloc(void* ptr, size_t new_size);
```

Estructura de la función ***realloc()***

Ejemplo de Realloc

```c
int main() {

    // Asignar memoria para un array de 3 enteros
    int *p = malloc(3 * sizeof(int));

    // Utilizar la memoria asignada
    for (int i = 0; i < 3; i++) {
        p[i] = i + 1;
    }

    // Imprimir los valores originales
    printf("Valores originales: ");
    for (int i = 0; i < 3; i++) {
        printf("%d ", p[i]);
    }
    printf("\n");

    // Redimensionar el array a 5 enteros
    p = realloc(p, 5 * sizeof(int));

    // Utilizar la memoria redimensionada
    for (int i = 3; i < 5; i++) {
        p[i] = i + 1;
    }

    // Imprimir los valores después de redimensionar
    printf("Valores después de redimensionar: ");
    for (int i = 0; i < 5; i++) {
        printf("%d ", p[i]);
    }
    printf("\n");

    // Liberar la memoria cuando ya no sea necesaria
    free(p);

    return 0;
}
```

Ejercicio 1

Realice un programa donde se le pida al usuario ingresar números, uno por uno, los cuales sean almacenados en un arreglo en memoria dinámica. Si el usuario ingresa el número 0 se muestra la lista completa de números y termina el programa.

Pista: *Usar la función realloc para aumentar el tamaño del arreglo en memoria dinámica*

```
leonardo@leonardo-acer:~/ProgramacionDeComputadores/Lab9$ ./ejercicio1
Ingrese un nuevo número (o ingrese 0 para terminar): 12
Ingrese un nuevo número (o ingrese 0 para terminar): 24
Ingrese un nuevo número (o ingrese 0 para terminar): 36
Ingrese un nuevo número (o ingrese 0 para terminar): 42
Ingrese un nuevo número (o ingrese 0 para terminar): 0
Elementos de la lista: 12 24 36 42
```

Ejercicio 2

Crear un programa que reciba dos números enteros n y m mayores a 4, y cree una matriz nxm en memoria dinámica y la rellene de la forma matr[i][j] = i*m + j, luego intercambie las posiciones matr[2][3] con matr[3][1], y matr[3][m-2] con matr[n-1][m-1].

Imprimir la matriz antes y después de los intercambios.