| | Práctica 3 |
|---|---|
| Facultad de Ingeniería | Laboratorios de docencia |

# Laboratorio de Computación Salas A y B

*Profesor(a):* M.I. Heriberto García Ledezma

*Asignatura:* Estructura de datos y algoritmos I

*Grupo:* 15

*No de Práctica(s):* Práctica 4

*Integrante(s):* Fuentes Llantada Marco Antronio

Rojas Contreras Aaron

*No. de lista o brigada:* 11 y 31

*Semestre:* 2025-2

*Fecha de entrega:*

*Observaciones:*

CALIFICACIÓN: _____

## Objetivos de la práctica

Utilizar funciones en lenguaje C que permiten reservar y almacenar información de manera dinámica (en tiempo de ejecución).

## Ejercicios de la práctica

### Ejercicio 1.

Programa que tenga capacidad para para almacenar 7 caracteres. Enseguida que le pregunte al usuario uno a uno cada carácter a almacenar. Después que le pregunte si quiere almacenar más caracteres. Si el usuario dice que “SI”, que le pregunte cuántos caracteres más desea guardar…

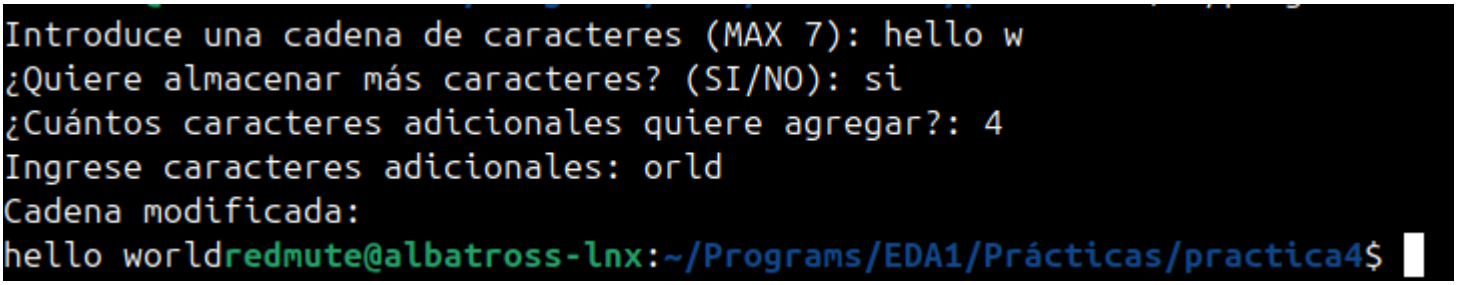
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main () {
    int i;
    char string [8];
    char *add_string = NULL;
    printf("ingrese caracter:\n");
    for ( i = 0; i < 7; i++)
    {
        printf("%d:", i + 1);
        scanf("%c", &string[i]);
        getchar();
    }

    char ans[3];
    printf("¿Quiere almacenar más caracteres?: ");
    scanf("%s", ans);
    int add;

    if (strcmp(ans, "SI") == 0 || strcmp(ans, "si") == 0 || strcmp(ans, "Si") == 0)
    {

        printf("¿Cuántos caracteres adicionales quiere agregar?:");
        scanf("%d", &add);
        if (add<=0)
        {
            printf("error:no se puede almacenar espacios menores o iguales a 0");
            return 1;
        }
        else
        add_string = (char*)calloc(add, sizeof(char));
        printf("ingrese caracter(es) adicional(es):\n");

        if (add_string == NULL) {
            printf("Error: No se pudo reservar memoria.\n");
            return 1;
        }
            else
        getchar();

        for ( i = 0; i < add; i++)
        {
            printf("%d:", i + 1);
            scanf("%c", &add_string[i]);
            getchar();
        }
        printf("Datos ingresados:\n");
        for (i = 0; i < 7; i++) {
            printf("%c", string[i]);
        }
        for ( i = 0; i < add; i++)
        {
            printf("%c", add_string[i]);
        }
        printf("\n");
        free(add_string);
        return 0;
    }
    else
    printf("Datos ingresados:\n");
    for (i = 0; i < 7; i++) {
        printf("%c", string[i]);
    }
    printf("\n");
    return 0;
}
```

```
redmute@albatross-lnx:~/Programs/EDA1/Prácticas/practica4$ ./programa0
ingrese caracter:
1:h
2:e
3:l
4:l
5:o
6:
7:w
¿Quiere almacenar más caracteres?: si
¿Cuántos caracteres adicionales quiere agregar?:4
ingrese caracter(es) adicional(es):
1:o
2:r
3:l
4:d
Datos ingresados:
hello world
```

## Ejercicio 2.

Programa que reserve espacio para almacenar una cadena de hasta 7 caracteres. Enseguida que le pregunte cuál es la cadena y la guarde en el espacio reservado…

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

int main() {
    char string[8];
    char *add_string = NULL;
    char *final_string = NULL;

    printf("Introduce una cadena de caracteres (MAX 7): ");
    fgets(string, 8, stdin);

    char ans[3];
    printf("¿Quiere almacenar más caracteres? (SI/NO): ");
    scanf("%s", ans);

    if (strcmp(ans, "SI") == 0 || strcmp(ans, "si") == 0 || strcmp(ans, "Si") == 0) {
        int add;
        printf("¿Cuántos caracteres adicionales quiere agregar?: ");
        scanf("%d", &add);

        if (add <= 0) {
            printf("Error: No se puede almacenar espacios menores o iguales a 0\n");
            return 1;
        }

        while (getchar() != '\n');

        add_string = (char*)malloc((add + 1) * sizeof(char));

        if (add_string == NULL) {
            printf("Error: No se pudo reservar memoria.\n");
            return 1;
        }

        printf("Ingrese caracteres adicionales: ");
        fgets(add_string, add + 1, stdin);

        final_string = (char*)malloc((strlen(string) + strlen(add_string) + 1) * sizeof(char));

        if (final_string == NULL) {
            printf("Error: No se pudo reservar memoria para la cadena final.\n");
            free(add_string);
            return 1;
        }

        strcpy(final_string, string);
        strcat(final_string, add_string);

        printf("Cadena modificada:\n%s", final_string);

        free(add_string);
        free(final_string);
    } else {
        printf("Cadena ingresada:\n%s", string);
    }

    return 0;
}
```




## Ejercicio 3.

Para la siguiente carrera nocturna de la UNAM se estima que asistan más de 5,000 corredores. Es posible que no todos los participantes lo termine, pero se desea registrar los tiempos de quienes sí lleguen a la meta y calcular el promedio. Haga un programa que permita registrar el tiempo de llegada en horas de cada participante que llegue a la meta. Para hacer un uso de memoria conforme se requiera, considere registrar los tiempos en bloques de cinco participantes a la vez, y cuando ya se hayan registrado los datos para cinco atletas preguntar al usuario si se desea ingresar las calificaciones de otros cinco. En caso de que diga que sí, ajustar el tamaño para guardar los tiempos de otras cinco personas. Esto se deberá repetir mientras el usuario diga que sí quiere seguir registrando tiempos para otros cinco participantes. Cuando ya no haya datos por registrar, el programa mostrará en pantalla los tiempos registrados, el número de tiempos registrados y el promedio.

Nota: No olvide verificar sí se pudo reservar la memoria extra en cada ocasión y liberarla cuando ya no se use.

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

int main() {
    float *tiempos = NULL;
    int capacidad = 0, contador = 0;
    char respuesta;

    do {
        // Aumentar el tamaño del arreglo en bloques de 5
        capacidad += 5;
        float *temp = realloc(tiempos, capacidad * sizeof(float));

        if (temp == NULL) {
            printf("Error: No se pudo asignar memoria.\n");
            free(tiempos);
            return 1;
        }

        tiempos = temp;

        // Registrar los tiempos de 5 participantes
        for (int i = 0; i < 5; i++) {
            printf("Ingrese el tiempo en horas del participante %d: ", contador
            scanf("%f", &tiempos[contador]);
            contador++;
        }

        // Preguntar si desea registrar más tiempos
        printf("Desea registrar otros 5 tiempos? (s/n): ");
        scanf(" %c", &respuesta);

    } while (respuesta == 's' || respuesta == 'S');

    // Calcular el promedio
    float suma = 0;
    for (int i = 0; i < contador; i++) {
        suma += tiempos[i];
    }
    float promedio = (contador > 0) ? suma / contador : 0;

    // Mostrar resultados
    printf("\nTiempos registrados:\n");
    for (int i = 0; i < contador; i++) {
        printf("Participante %d: %.2f horas\n", i + 1, tiempos[i]);
    }
    printf("Total de tiempos registrados: %d\n", contador);
    printf("Tiempo promedio: %.2f horas\n", promedio);

    // Liberar memoria
    free(tiempos);

    return 0;
}
```

```
$ ./a.exe
Ingrese el tiempo en horas del participante 1: 2
Ingrese el tiempo en horas del participante 2: 4
Ingrese el tiempo en horas del participante 3: 3
Ingrese el tiempo en horas del participante 4: 2
Ingrese el tiempo en horas del participante 5: 43
Desea registrar otros 5 tiempos? (s/n): n

Tiempos registrados:
Participante 1: 2.00 horas
Participante 2: 4.00 horas
Participante 3: 3.00 horas
Participante 4: 2.00 horas
Participante 5: 43.00 horas
Total de tiempos registrados: 5
Tiempo promedio: 10.80 horas
```

## Ejercicio 4

Programa que permita registrar los datos de las mascotas que recibe una guardería. De cada mascota se registrarán los siguientes datos:

• Nombre (una sola palabra)

• Tipo de animal

• Edad

• Descripción de color y tamaño. Hasta 100 caracteres.

• Nombre del propietario.

• Celular del propietario.

Primero deberá imprimir en pantalla la leyenda Guardería Cat y Perry. Enseguida deberá preguntar si se desea registrar una mascota. Si se indica que SI, deberá preguntar los datos de la mascota ya indicados. El programa deberá permitir registrar tantas mascotas como sea necesario. Los datos de cada mascota deberán almacenarse en un tipo de dato Mascota, es decir, mediante una estructura. Deberá reservar memoria dinámica para almacenar los datos de una mascota y cada que se deseé agregar los datos de una nueva deberá ampliar la memoria disponible para almacenar los datos respectivos. Cuando ya no se deseé agregar datos, el programa presentará en pantalla de forma numerada los nombres registrados de todas las mascotas. Y solicitará que se escriba el número de la mascota de la cual quiere ver los datos completos. Por último, imprimirá en pantalla los datos de la mascota seleccionada.

Notas: En este ejercicio almacenará en memoria dinámica los datos de variables de tipo estructura. Para acceder a tales estructuras utilizará un apuntador a la memoria dinámica asignada, en consecuencia, deberá acceder a cada campo de las estructuras mediante un apuntador. Revise las notas de clase para recordar cómo acceder mediante un apuntador a una estructura.

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

// Definición de la estructura Mascota
typedef struct {
    char nombre[30];
    char tipo[20];
    int edad;
    char descripcion[100];
    char propietario[50];
    char celular[15];
} Mascota;

int main() {
    Mascota *mascotas = NULL;
    int contador = 0;
    char respuesta;

    printf("Guardería Cat y Perry\n");

    do {
        // Reservar memoria para una nueva mascota
        Mascota *temp = realloc(mascotas, (contador + 1) * sizeof(Mascota));
        if (temp == NULL) {
            printf("Error: No se pudo asignar memoria.\n");
            free(mascotas);
            return 1;
        }
        mascotas = temp;

        // Ingreso de datos
        printf("Ingrese el nombre de la mascota (una palabra): ");
        scanf("%s", mascotas[contador].nombre);
        printf("Ingrese el tipo de animal: ");
        scanf("%s", mascotas[contador].tipo);
        printf("Ingrese la edad de la mascota: ");
        scanf("%d", &mascotas[contador].edad);
        getchar(); // Limpiar el buffer
        printf("Ingrese la descripción de color y tamaño: ");
        fgets(mascotas[contador].descripcion, sizeof(mascotas[contador].descripcion), stdin);
        strtok(mascotas[contador].descripcion, "\n"); // Eliminar el salto de línea
        printf("Ingrese el nombre del propietario: ");
        fgets(mascotas[contador].propietario, sizeof(mascotas[contador].propietario), stdin);
        strtok(mascotas[contador].propietario, "\n");
```

```c
        printf("Ingrese el celular del propietario: ");
        scanf("%s", mascotas[contador].celular);

        contador++;

        // Preguntar si se desea registrar otra mascota
        printf("¿Desea registrar otra mascota? (s/n): ");
        scanf(" %c", &respuesta);
    } while (respuesta == 's' || respuesta == 'S');

    // Mostrar las mascotas registradas
    printf("\nMascotas registradas:\n");
    for (int i = 0; i < contador; i++) {
        printf("%d. %s\n", i + 1, mascotas[i].nombre);
    }

    // Seleccionar mascota para ver detalles
    int seleccion;
    printf("\nIngrese el número de la mascota para ver detalles: ");
    scanf("%d", &seleccion);

    if (seleccion > 0 && seleccion <= contador) {
        seleccion--; // Ajustar al índice
        printf("\nDetalles de la mascota:\n");
        printf("Nombre: %s\n", mascotas[seleccion].nombre);
        printf("Tipo: %s\n", mascotas[seleccion].tipo);
        printf("Edad: %d años\n", mascotas[seleccion].edad);
        printf("Descripción: %s\n", mascotas[seleccion].descripcion);
        printf("Propietario: %s\n", mascotas[seleccion].propietario);
        printf("Celular: %s\n", mascotas[seleccion].celular);
    } else {
        printf("Número de mascota inválido.\n");
    }

    // Liberar memoria
    free(mascotas);

    return 0;
}
```

```
rexta@DESKTOP-J5AEL8D ~
$ ./a.exe
Guardería Cat y Perry
Ingrese el nombre de la mascota (una palabra): carolina
Ingrese el tipo de animal: gato
Ingrese la edad de la mascota: 6
Ingrese la descripción de color y tamaño: marron
Ingrese el nombre del propietario: Marco Antonio
Ingrese el celular del propietario: 5511544573
¿Desea registrar otra mascota? (s/n): n

Mascotas registradas:
1. carolina

Ingrese el número de la mascota para ver detalles:
```

## Conclusiones

El uso de funciones en lenguaje C para reservar y almacenar información de manera dinámica en tiempo de ejecución es una herramienta fundamental en la programación eficiente y flexible. A través de funciones como `malloc()`, calloc(), realloc() y free(), es posible gestionar la memoria de forma dinámica, optimizando el uso de los recursos del sistema y permitiendo la creación de estructuras de datos de tamaño variable. Esto resulta especialmente útil en aplicaciones que requieren un manejo eficiente de la memoria, como bases de datos, simulaciones o algoritmos que procesan grandes volúmenes de información. Sin embargo, un uso incorrecto de estas funciones puede generar errores como fugas de memoria o accesos indebidos, por lo que es crucial liberar correctamente la memoria asignada para garantizar un programa seguro y estable.