TP6-2025

El objetivo de esta práctica es que comiencen a trabajar con **memoria dinámica** en lenguaje C.

Forma de entrega

Se deberán realizar los ejercicios en clases para consultas

Se indicará un par de ejercicios que deberán ser subidos al campus para su verificación.

El nombre de cada archivo debe ser **`ejercicio`** seguido del número de ejercicio más la extensión **`.c`**, de esta manera, el primer ejercicio será entonces:

`ejercicio1.c`

Indicaciones generales

- Recuerden tener en cuenta las **Cuestiones de Estilo**.

- En ningún caso se aceptará el uso de **variables globales**.

- Toda la información necesaria para el funcionamiento de las funciones a desarrollar debe ser **pasada como argumentos**.
- Mantengan **separado lo que es entrada, algoritmo y salida**.
- No olviden **documentar las funciones** implementadas, indicando:
    - El propósito de la función.
    - Qué representan sus argumentos.
    - Qué valor retorna.

Ejercicio 1 – Reserva dinámica de enteros

Objetivo: uso de **malloc** y **free**.

Enunciado: Escriba un programa en C que solicite al usuario la cantidad de números enteros a almacenar.

Reserve dinámicamente la memoria necesaria utilizando malloc.

Permita que el usuario ingrese cada número, muéstrelos en orden inverso y libere la memoria antes de finalizar.

Conceptos clave: Uso de malloc y free. Acceso mediante punteros. Control de errores de asignación.

Ejercicio 2 – Layout de memoria y direcciones

Objetivo: uso de espacio de memoria.

Enunciado: Desarrolle un programa que defina:

1. Una variable global.

2. Una variable local.

3. Una variable estática.

4. Una variable dinámica (reservada con malloc).

Muestre por pantalla las direcciones de memoria de cada variable y determine a qué segmento pertenece (código, datos, stack o heap).

Conceptos clave: Layout de memoria de un programa C. Diferencias entre memoria estática, automática y dinámica.

Ejercicio 3 – Copia dinámica de cadenas

Objetivo: memoria dinámica aplicada a texto.

Enunciado: Implemente una función char *copiar_cadena(const char *src) que:

1. Reserve memoria dinámica suficiente para copiar src.

2. Copie el contenido original.

3. Devuelva el puntero a la nueva cadena.

Desde main, solicite una cadena, use la función para copiarla y muestre el resultado.

Libere la memoria antes de terminar.

Conceptos clave: Reserva dinámica proporcional a strlen. Copia y manejo de punteros a char. Liberación segura.

Ejercicio 4 – Punteros genéricos (void *)

Objetivo: uso de punteros genéricos y el casting de tipos.

Enunciado: Defina una función void imprimir_valor(void *ptr, char tipo) que reciba un puntero genérico y un carácter que indique el tipo de dato:

- 'i' para entero

- 'f' para flotante

- 'c' para carácter

Dentro de la función, realice el casting adecuado y muestre el valor correspondiente. Pruebe la función con tres variables diferentes.

Conceptos clave: Conversión de tipos con punteros. Representación de datos en memoria. Uso flexible de void *.

Ejercicio 5 – Cadena segura: verificación de palíndromo

Objetivo: cadenas con memoria dinámica.

Enunciado: Cree un programa que lea una frase, reserve memoria dinámica para almacenarla y determine si la frase es un palíndromo, ignorando espacios, mayúsculas y signos de puntuación.

El programa debe:

1. Leer una línea completa del usuario.

2. Limpiar la cadena (eliminar espacios y convertir a minúsculas).

3. Verificar si la cadena limpia se lee igual al derecho y al revés.

4. Mostrar el resultado (“Es palíndromo” o “No es palíndromo”).

5. Liberar la memoria antes de finalizar.

Ejemplo de entrada/salida:

Ingrese una frase: Anita lava la tina

Resultado: Es palíndromo

Conceptos clave: Manejo seguro de cadenas con malloc. Normalización de texto. Comparación de caracteres en memoria dinámica.