Temario:

- Punteros
- Listas enlazadas simples
- Listas circulares

Aclaración: se debe modularizar y reutilizar código cuando sea adecuado.

Ejercicios:

Ejercicio 1.

Dado el siguiente programa, indicar la salida en cada punto señalado.

```
#include <iostream>
using namespace std;

int main() {
    int *p1, *p2;

    p1 = new int;
    *p1 = 42;
    p2 = p1;
    cout << *p1 << endl;
    cout << *p2 << endl;  // A.Luego de ejecutar esta línea

    *p2 = 53;
    cout << *p1 << endl;
    cout << *p2 << endl;  //B. Luego de ejecutar esta línea

    p1 = new int;
    *p1 = 88;
    cout << *p1 << endl;
    cout << *p2 << endl;
    cout << "FIN";        //C. Luego de ejecutar esta línea
    return 0;
}
```

Ejercicio 2.

¿Qué salidas produce el siguiente código?

```
int *p1, *p2;
p1 = new int;
p2 = new int;
*p1 = 10;
*p2 = 20;
cout << *p1 << " " << *p2 << endl;
*p1 = *p2;
cout << *p1 << " " << *p2 << endl;
*p2 = 30;
cout << *p1 << " " << *p2 << endl;
```

Ejercicio 3.

Dados los siguientes punteros a char:

```
char* x = new char;
char* z = x;
```

Escribir y ejecutar las instrucciones para hacer lo siguiente:

a) Almacenar el carácter 'M' en el espacio de memoria apuntado por x.
b) Imprimir el carácter apuntado por x.
c) Almacenar el carácter 'P' en el espacio de memoria apuntado por z.
d) Imprimir el carácter apuntado por z y luego el apuntado por x.
   ¿Qué carácter crees que imprimirá cada una y por qué?
e) Eliminar el dato apuntado por x.
   ¿Qué sucedería si a continuación se intentara imprimir *z y por qué?
f) Obtener un nuevo espacio de memoria para un char, almacenando su dirección en x.
g) Permitir al usuario ingresar un char por teclado y almacenarlo en el nuevo espacio apuntado por x.
h) Si el carácter ingresado por el usuario en el inciso anterior es '*', imprimir "asterisco".

Ejercicio 4.

Dados los siguientes structs:

```
struct Producto {
    string descripcion;
    float precio;
};
struct Venta {
    Producto producto;
    int cantidad;
};
```

Escribir y luego ejecutar las instrucciones necesarias para:

a) Declarar una variable puntero a Venta y reservar memoria.
b) Pedir al usuario que ingrese una cantidad vendida, descripción de producto y precio de producto, almacenando los datos en la Venta apuntada por el puntero declarado en el inciso anterior.
c) Restar un 15% al precio del Producto guardado en la Venta apuntada por el puntero.
d) Imprimir la descripción del producto dentro de la Venta y monto total (precio del producto * cantidad).

Ejercicio 5.

Se lee de teclado una secuencia de números enteros, finalizando cuando el usuario ingresa un cero (que no debe insertarse). Los números leídos deberán ser almacenados en una lista enlazada simple. Hacer tres versiones, insertando en cada una con diferente criterio, a saber:

a) Al inicio de la lista.
b) Al final de la lista.
c) En la posición correcta para que la lista quede ordenada de menor a mayor. Si el número a insertar ya se encontraba en la lista, deberá quedar antes que todas sus repeticiones.

Finalmente, imprimir la lista.

Ejercicio 6.

Permitir al usuario guardar en una lista enlazada simple los nombres de los alumnos de un curso, insertando al final de la lista y cortando la carga cuando se inserte como nombre "x", ya sea en mayúscula o en minúscula (el cual no debe insertarse en la lista).
A continuación, solicitar al usuario el nombre de un alumno e informar si se encuentra en la lista o no.

Por último, eliminar a la alumna "Josefina Ortega" de la lista (se asume que no se han almacenado nombres repetidos).

Ejercicio 7.

Se lee de teclado una secuencia de números que finaliza con la primera ocurrencia de un número negativo, el cual no debe ser procesado. Los números deben ser almacenados en una lista enlazada simple. Se solicita:

A. Leer de teclado un número e informar si existe en la lista.
B. Leer de teclado un número e informar la cantidad de ocurrencias del número en la lista.
C. Implementar una función que reciba la lista de números y genere dos nuevas listas, una con los números pares y otra con los impares. Las nuevas listas deberán estar formadas con los nodos de la lista original, la que quedará vacía después de invocar a la función. Imprimir las listas resultantes.

Ejercicio 8.

Implementar una función que reciba dos parámetros: una lista enlazada simple de números enteros y un número entero. La función debe eliminar todas las ocurrencias del número indicado en el segundo parámetro. Retornar la cantidad de ocurrencias eliminadas. Nota: se debe pasar una sola vez por cada uno de los elementos de la lista.

Ejercicio 9.

Desarrollar un programa que realice las siguientes operaciones en listas enlazadas simples de números enteros.

A. Cargar una lista insertando ordenado en forma descendente. Finaliza luego de ingresado el cero, que también se debe agregar a la lista.
B. Eliminar las ocurrencias de los números que son múltiplos de 3.
C. Imprimir la lista antes y después de la eliminación.

Ejercicio 10.

Cargar una lista enlazada simple con palabras ingresadas por el usuario, las cuales deben almacenarse en minúsculas independientemente de cómo las ingrese. Luego, a partir de esa lista, generar una nueva lista con las palabras que se encuentran repetidas. Finalmente, imprimir ambas listas.

Ejercicio 11.

Implementar una función que reciba dos listas enlazadas simples con datos de tipo string ordenadas según el alfabeto y retorne una tercera lista ordenada por el mismo criterio, conformada por todos los elementos de las listas originales (merge). Agregar las funciones auxiliares que sean necesarias para cumplir con este propósito.

Ejercicio 12.

Generar una lista enlazada simple con la nómina de artículos de un supermercado. De cada artículo se conoce: código, descripción, precio y stock. La carga finaliza con el código -1. Se solicita:

A. Leer de teclado un porcentaje e incrementar el precio de todos los artículos en esa cantidad.
B. Incrementar el stock de un artículo. Se lee de teclado el código y la cantidad con la que se debe incrementar el stock. Si no existe el artículo en la lista se debe informar.
C. Eliminar de la lista los artículos que no tienen stock.

Ejercicio 13.

Dos sucursales de Fiestísima disponen cada una de una lista enlazada simple con el detalle de las ventas realizadas. De cada venta se conoce: código de producto y cantidad vendida (un producto pudo haber sido vendido 0, 1 o más veces en la misma sucursal). Las listas de ventas están ordenadas de menor a mayor por código de producto. Realizar una función que reciba las dos listas y retorne una nueva lista unificando todas las ventas, ordenada por el mismo criterio. Utilizar un algoritmo destructivo (que reutilice los nodos de las listas originales). Agregar las funciones auxiliares que sean necesarias.

Ejercicio 14.

Generar una lista con los datos personales de los alumnos de un colegio (legajo, apellido y nombre, DNI) y las notas de tres materias: matemática, literatura y geografía (para cada alumno se cargarán las tres materias). Las notas pueden ir de 1 a 10 y se debe validar que el usuario no ingrese notas incorrectas, volviendo a pedir una nueva nota cada vez que ingrese una incorrecta, hasta que ingrese un valor en el rango indicado. Se pide informar la cantidad de alumnos que aprobaron las tres materias (se aprueba con 7) y el porcentaje que representan los aprobados sobre el total de alumnos.

Ejercicio 15.

"The Golden Gate Bridge" es el puente más famoso de la ciudad de San Francisco (California). Cuenta con una longitud aproximada de 1280 metros y posee tres radares de toma de velocidad. Por cada móvil que se desplaza por las vías principales se registra la siguiente información: sensor que lo captó (puede ser "norte", "sur" o "medio"), patente (no tiene un formato único ya que difiere de un estado a otro y pueden transitar autos extranjeros, aunque se sabe que están formadas sólo por letras y números) y velocidad en km/h.

A. Cargar las lecturas que hacen los sensores en una lista enlazada simple ordenando por patente y, para la misma patente, ordenando por sensor (ya que un mismo automóvil puede ser captado por más de un sensor). Finaliza con la patente "aaa99", que no se debe ingresar.

B. Dada la lista generada en el inciso anterior, armar tres listas: una por cada sensor (el orden de los elementos tiene que ser el mismo que en la lista original). Luego, imprimir las tres listas.

Ejercicio 16.

Se lee de teclado una secuencia de números enteros, finalizando cuando el usuario ingresa un cero (que no debe insertarse). Los números leídos deberán ser almacenados en una lista circular. Hacer dos versiones, insertando en cada una con diferente criterio, a saber:

a) Al inicio de la lista.
b) Al final de la lista.

Finalmente, imprimir la lista.

Ejercicio 17.

Dada una lista circular que contiene palabras, solicitar al usuario el ingreso de una posición (numérica, mayor a 0 y menor o igual que N, siendo N la cantidad de nodos en la lista) y luego el ingreso de una nueva palabra para insertarla en la posición indicada. Ejemplo: si la lista contiene las siguientes palabras, en este orden: "chocolate" - "merengue" - "frutilla" - "caramelo", y el usuario quiere insertar la palabra "menta" en la posición 4, la lista quedará: "chocolate" - "merengue" - "frutilla" - "menta" - "caramelo". Para ello, realizar dos funciones:

a) **Nodo* buscar_posicion(Nodo* fin, int posicion)**
   Parámetros: el puntero al final de la lista y un número que indica la posición dada por el usuario.
   Retorno: puntero al nodo anterior a la posición (si es que existe). En el ejemplo dado, el puntero apuntará al nodo con la palabra "merengue". Si no existe la posición, retornará nullptr.
b) **void insertar_despues(Nodo* anterior, string palabra)**
   Parámetros: puntero al nodo anterior (el retornado por la función buscar_posicion) y la palabra a insertar en un nuevo nodo (a continuación de anterior).

Ejercicio 18.

Cargar en una lista circular, insertando al final, los títulos de las charlas que se darán en una conferencia de tecnología. Independientemente de cómo las ingrese el usuario, se deben almacenar con la primera letra en mayúscula y el resto en minúsculas.

a) Imprimir la lista resultante.
b) Solicitar al usuario el ingreso de un título e informar si existe en la lista. La búsqueda debe ser "case insensitive" (es decir, sin importar que se ingrese con diferentes mayúsculas y minúsculas, si el título coincide debe informarse que existe).

Ejercicio 19.

Cargar en una lista circular, insertando al inicio, los números de CUIT de los proveedores de un comercio.

a) Dado un número de CUIT, eliminarlo de la lista.
b) Informar cuántos elementos hay en la lista después de la eliminación.

Ejercicio 20.

Crear una lista circular con los datos de los procesos en ejecución en un sistema operativo, donde cada uno consta de: PID (número de proceso), nombre de usuario, segundos que se ejecutó, fecha en que se empezó a ejecutar (numérica, formato DDMM). Crear la lista insertando al principio.

a) Informar el PID del proceso que más segundos se ejecutó.
b) Eliminar el proceso que se empezó a ejecutar el 14 de marzo (si hay más de uno, sólo el primero).
c) Aumentar en 2 la cantidad de segundos que se ejecutó el proceso con PID 61, si existe.
d) Listar todos los procesos del usuario "root" (imprimiendo el PID y segundos de ejecución de cada uno).

Ejercicio 21.

Cargar de teclado, en una lista circular, los datos de los jugadores de un juego de rol de mesa, para administrar sus turnos. De cada uno se almacenará: nombre de jugador, puntaje, raza de su personaje (dato numérico, donde 1 es mago, 2 es guerrero, 3 es elfo).

a) Se debe permitir cargar, como máximo, 10 jugadores, insertando cada nuevo jugador al final. Validar también que la raza sólo sea 1, 2 ó 3, dejando al usuario en un bucle hasta que ingrese una raza válida.
b) Dado el nombre de un jugador, indicar las características de la raza de su personaje: si es 1 (mago) se debe imprimir "1 punto de sanación por cada 5 de daño recibido"; si es 2 (guerrero) se debe imprimir "20% más de daño infligido"; si es 3 (elfo) se debe imprimir "capacidad de resucitar a un compañero a cambio de 200 puntos".
c) Dada una raza y una cantidad de puntos de penalidad, restar del puntaje la cantidad dada, para todos los jugadores de esa raza.
d) Imprimir los nombres de todos los jugadores, junto a su puntaje.