## I – CREACIÓN DE PROGRAMAS (I)

Realizar un programa que mediante 2 números de entrada se calcule, suma, resta, multiplicación y división.

1. Crear proyecto y ficheros.
2. Agregar librería iostream para trabajar.
3. Declaración de tu función main o principal.
4. Declaración de funciones para sumar, restar, multiplicar y dividir, con sus parámetros necesarios que solo recibirán dos números para todos y cada uno de ellos regresar el resultado de la operación interna dentro de cada función.

**NOTA:** No olvides agregar las funciones en base a la estructura básica de un programa.

5. Dadas las funciones ya definidas, es necesario agregar sus prototipos para su uso en todo el programa.
6. En la función principal agregar 2 entradas de información para posteriormente llamar las funciones para cada operación e imprimir en pantalla.
7. Los resultados deben de aparecer correctamente según cada operación.

## II – CREACIÓN DE PROGRAMAS (II)

Generar un programa que mediante con una variable, un puntero y un puntero doble.

Generar un programa en el cual se declare una variable normal con un determinado valor, posterior declarar un puntero que apunte a la dirección de la variable previa, y por ultimo declarar un puntero doble que a la dirección del puntero previamente declarado.

1. Crear proyecto y ficheros.
2. Agregar librería iostream para trabajar.
3. Declaración de tu función main o principal.
4. Declaración de las variables a utilizar:
    a. Variable entera con valor de 10.
    b. Puntero * que apunta la dirección de la variable.
    c. Puntero doble ** que apunta a la dirección del puntero.
5. Imprimir en pantalla el contenido de cada variable:
    Variable = 10, Puntero = 10, Puntero doble = 10
6. Imprimir en pantalla los contenidos y accesos a cada variable previamente declarada.
    a. Variable: dirección y contenido.
    b. Puntero: dirección de puntero, a donde apunta y contenido.
    c. Puntero doble: dirección de puntero doble, a donde apunta, dirección de la variable y contenido.
7. Por último, asignar un valor diferente a la variable, pero desde el acceso de los 2 punteros.

## III – CREACIÓN DE PROGRAMAS (III)

Generar un programa que genere una lista simple y mostrar en pantalla los elementos agregados a la lista.

Generar un programa en el cual se declare una variable normal con un determinado valor, posterior declarar un puntero que apunte a la dirección de la variable previa, y por ultimo declarar un puntero doble que a la dirección del puntero previamente declarado.

1. Crear proyecto y ficheros.
2. Agregar librería iostream para trabajar.
3. Declaración de tu función main o principal.
4. Declarar una estructura de lista simple con los siguientes valores:
    a) Valor: que será el valor que almacenará cada elemento de la lista.
    b) PtrSig: es el puntero que apuntara al siguiente nodo.
    c) Primero: es el puntero estático que siempre tendrá el inicio de la lista.
5. Declarar una función para Agregar un nodo a la lista simple, esta función podrá recibir un valor para almacenarlo en el elemento, y también podrá realizar las operaciones para ir ligando a la lista cada elemento que se va agregando.
6. Declarar una función para Mostrar todos los elementos de la lista e imprimirlos en pantalla con sus atributos.
7. En la función principal agregar al menos 3 elementos y mostrar en pantalla.

## IV – CREACIÓN DE PROGRAMAS (IV)

Generar un programa que genere una lista doble, se eliminen elementos, mostrar los elementos antes de eliminar de lista y también después de eliminar.

Generar un programa en el cual se declare una variable normal con un determinado valor, posterior declarar un puntero que apunte a la dirección de la variable previa, y por último declarar un puntero doble que a la dirección del puntero previamente declarado.

1. Crear proyecto y ficheros.
2. Agregar librería iostream para trabajar.
3. Declaración de tu función main o principal.
4. Declarar una estructura de lista simple con los siguientes valores:
    a) **Valor:** que será el valor que almacenará cada elemento de la lista.
    b) **PtrSig:** es el puntero que apuntara al siguiente nodo.
    c) **PtrAnt:** es el puntero que apuntara al siguiente nodo.
    d) **Primero:** es el puntero estático que siempre tendrá el inicio de la lista.
5. Declarar una función para Agregar un nodo a la lista simple, esta función podrá recibir un valor para almacenarlo en el elemento, y también podrá realizar las operaciones para ir ligando a la lista cada elemento que se va agregando.
6. Declarar una función para Mostrar todos los elementos de la lista e imprimirlos en pantalla con sus atributos.
7. Declarar una función para Eliminar un elemento de una lista doble, tomando en cuenta que puede ser alguno de los 4 casos en que el elemento a eliminar puede ser único, el primero, el de en medio o el ultimo.
8. En la función principal agregar al menos 4 elementos, mostrar en pantalla la lista doble, posteriormente eliminar al menos 2 nodos y mostrar de nuevo en pantalla todos los nodos de la lista, y deberá ser evidente el borrado de los elementos de la lista.

## V – CREACIÓN DE PROGRAMAS

Realizar un programa que genere una lista doble, compruebe si existe un archivo del cual cargar información, si no existe al finalizar el programa se guarde, y una vez iniciado de nuevo sea visible la lista que se generó previamente.

Realizar un programa como el anterior en el que se genere una lista ligada de al menos 4 elementos, detecte si encuentra un archivo binario para cargar información, si no existe, este no cargara nada, si existe, cargar todos los elementos y mostrar la lista de todos los elementos.

1. Crear proyecto y ficheros.
2. Agregar librería iostream para trabajar.
3. Declaración de tu función main o principal.
4. Declarar una estructura de lista simple con los siguientes valores:
   I) Valor: que será el valor que almacenará cada elemento de la lista.
   II) PtrSig: es el puntero que apuntara al siguiente nodo.
   III) PtrAnt: es el puntero que apuntara al siguiente nodo.
   IV) Primero: es el puntero estático que siempre tendrá el inicio de la lista.
5. Declarar una función para Agregar un nodo a la lista simple, esta función podrá recibir un valor para almacenarlo en el elemento, y también podrá realizar las operaciones para ir ligando a la lista cada elemento que se va agregando.
6. Declarar una función para Mostrar todos los elementos de la lista e imprimirlos en pantalla con sus atributos.
7. En la función principal agregar al menos 4 elementos, mostrar en pantalla la lista doble, posteriormente eliminar al menos 2 nodos y mostrar de nuevo en pantalla todos los nodos de la lista, y deberá ser evidente el borrado de los elementos de la lista.
8. También en la función principal agregar la carga del archivo binario, al finalizar, si encuentra elementos este mismo deberá mostrarse en pantalla, y al final de la función principal agregar el guardado de archivo en caso de tener elementos en la lista.

## VI – EVENTOS, MENSAJES DE WINAPI, CALLBACKS

Objetivo general: Familiarizar al estudiante en la creación de ventanas en WinApi.

1. Cree un nuevo proyecto de C++ en visual studio 2022 utilizando la plantilla de "Proyecto vacío".}
2. En el "Explorador de soluciones", seleccione "Archivos de origen" y dándole click derecho agregue un nuevo elemento y asegúrese de que sea un archivo ".cpp", ingrese un nombre para el archivo o déjelo como "Source.cpp".
3. Incluya los elementos de código necesarios para su programa de WinApi como librerías de recursos y WinMain.
4. En el "Explorador de soluciones", seleccione "Archivos de recursos" y dándole click derecho agregue un nuevo recurso y seleccione como nuevo recurso un "Dialog" para poder visualizarlo seleccione la "Vista de recursos".
5. Coloque un prototipo y callback del diálogo basándose en el siguiente:

```
BOOL CALLBACK ventanaMain(HWND, UINT, WPARAM, LPARAM);
```

y dentro del WinMain haga la creación del handler del diálogo basándose en lo siguiente:

```
HWND hVentana = CreateDialog(hInst, MAKEINTRESOURCE("Id del dialogo"), NULL, "nombre de tu callback");
```

A continuación, agregue lo siguiente:

```
MSG msg;

ZeroMemory(&msg, sizeof(MSG));

ShowWindow(hVentana, SW_SHOW);

while (GetMessage(&msg, hVentana, NULL, NULL)) {

        TranslateMessage(&msg);

        DispatchMessage(&msg);

}
```

Lo que hacemos es tener donde recibir lo que sucede dentro de la aplicación, que sería msg y aseguramos que no tiene información basura con el zeromemory, después hacemos un while que estará escuchando las acciones del usuario.

6. Debajo del WinMain agregue lo siguiente:

```
INT_PTR CALLBACK ventanaMain(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) {

switch (msg) {

	case WM_CLOSE:

		DestroyWindow(hwnd);

		break;

	case WM_DESTROY:

		PostQuitMessage(1);

		break;

	}

	return 0;

}
```

Esto sería el cuerpo del callback que está ligado el dialogo, el switch lo que hace es recibir la acción del WinMain y decide que hacer dependiendo de que mensaje es, en este caso tenemos el WM_CLOSE cuando se cierra el dialogo y WM_DESTROY que es lo que se realizará después de cerrar el dialogo. El WPARAM y LPARAM son utilizados para recibir más información como el que ocurrió y que lo ocasionó.

7. Corra el proyecto y vea su ventana.

## VII – CONTROLES DE CAPTURA. CONTROLES DE DESPLIEGUE.

Objetivo general: Familiarizar al estudiante con la captura de datos mediante mouse y teclado su vez que el despliegue de esa información.

1. Cree un nuevo proyecto de C++ en visual studio 2022 utilizando la plantilla de “Proyecto vacío”.}
2. En el “Explorador de soluciones”, seleccione “Archivos de origen” y dándole click derecho agregue un nuevo elemento y asegúrese de que sea un archivo “.cpp”, ingrese un nombre para el archivo o déjelo como “Source.cpp”.
3. Incluya los elementos de código necesarios para su programa de WinApi como librerías de recursos y WinMain.
4. En el “Explorador de soluciones”, seleccione “Archivos de recursos” y dándole click derecho agregue un nuevo recurso y seleccione como nuevo recurso un “Dialog”. Para poder visualizarlo seleccione la “Vista de recursos”.
5. Dentro de su ventana coloque un editcontrol, un botón, un textlabel y un picturecontrol.
6. Logre hacer que al dar click en el botón se abra el explorador de archivos para seleccionar una imagen “.bmp” Luego seleccionada la imagen, el texto y la imagen se coloquen dentro del textlabel y picturecontrol respectivamente.
7. Corra el proyecto y corrobore su funcionamiento.

## VIII – CONTROLES COMUNES (I)

Objetivo general: Familiarizar al estudiante con algunos controles comunes en winapi.

1. Cree un nuevo proyecto de C++ en visual studio 2022 utilizando la plantilla de “Proyecto vacío”.
2. En el “Explorador de soluciones”, seleccione “Archivos de origen” y dándole click derecho agregue un nuevo elemento y asegúrese de que sea un archivo “.cpp”, ingrese un nombre para el archivo o déjelo como “Source.cpp”.
3. Incluya los elementos de código necesarios para su programa de WinApi como librerías de recursos y WinMain.
4. En el “Explorador de soluciones”, seleccione “Archivos de recursos” y dándole click derecho agregue un nuevo recurso y seleccione como nuevo recurso un “Dialog”. Para poder visualizarlo seleccione la “Vista de recursos”.
5. Cree un menú y vincúlelo a su ventana. Después dentro de su ventana coloque, un botón, una slider y un datetimepicker.
6. Logre hacer que al dar click en el botón se despliegue un messagebox que diga el valor seleccionado de la slider y la fecha seleccionada en el datetimepicker.
7. Corra el proyecto y corrobore su funcionamiento.

## IX – CONTROLES COMUNES (II)

Objetivo general: Familiarizar al estudiante con algunos controles comunes en winapi.

1. Cree un nuevo proyecto de C++ en visual studio 2022 utilizando la plantilla de “Proyecto vacío”.
2. En el “Explorador de soluciones”, seleccione “Archivos de origen” y dándole click derecho agregue un nuevo elemento y asegúrese de que sea un archivo “.cpp”, ingrese un nombre para el archivo o déjelo como “Source.cpp”.
3. Incluya los elementos de código necesarios para su programa de WinApi como librerías de recursos y WinMain.
4. En el “Explorador de soluciones”, seleccione “Archivos de recursos” y dándole click derecho agregue un nuevo recurso y seleccione como nuevo recurso un “Dialog”. Para poder visualizarlo seleccione la “Vista de recursos”.
5. Dentro de su ventana coloque, un botón, un combobox y un listbox.
6. Agregue mediante código 5 elementos al combobox, después haga que al dar click en el botón se agregue el elemento seleccionado del combobox a la lista y por último con un timer haga que cada 30 segundos se despliegue un messagebox que diga los elementos dentro de la listbox y borre los elementos dentro de la listbox.
7. Corra el proyecto y corrobore su funcionamiento.

## X – MULTIVENTANAS.

1. Cree un nuevo proyecto de C++ en visual studio 2022 utilizando la plantilla de “Proyecto vacío”.
2. En el “Explorador de soluciones”, seleccione “Archivos de origen” y dándole click derecho agregue un nuevo elemento y asegúrese de que sea un archivo “.cpp”, ingrese un nombre para el archivo o déjelo como “Source.cpp”.
3. Incluya los elementos de código necesarios para su programa de WinApi como librerías de recursos y WinMain, con los handlers de sus dos diálogos de manera global para poder usarlos en toda la aplicación y puedas ver fácilmente como interactúan ambas ventanas, recuerda que ya haciendo tu proyecto los handlers los debes crear solo cuando los ocupas. Tenga en cuenta que para que pueda utilizar más de una ventana, en el while que escucha las acciones en el WinMain en vez de especificar un handler d dialogo, coloque NULL.
4. En el “Explorador de soluciones”, seleccione “Archivos de recursos” y dándole click derecho agregue un nuevo recurso y seleccione como nuevo recurso un “Dialog” a continuación vuelva a realizar esto para tener dos diálogos diferentes. Para poder visualizarlos seleccione la “Vista de recursos”.
5. Coloque los dos prototipos y callbacks de los diálogos necesarios y dentro del WinMain haga la creación de los handlers de los diálogos de manera global fuera del WinMain y cree los dialogos pero solo muestre uno como previamente se le enseño.
6. Dentro del Dialogo 1 agregue un “Edit Control” y un botón que diga “Enviar a la lista” y otro que diga “Ver lista”. En el Dialogo 2 agregue una “List box”.
7. En los switches de los callbacks del dialogo 1, agregue el caso “WM_COMMAND” para escuchar cuando ocurra una acción dentro de cada dialogo. Dentro del caso de WM_COMMAND agregue un “if” y dentro del if coloque lo siguiente:

LOWORD(wParam) == “Id de tu botón agregar a la lista” && HIWORD(wParam) == BN_CLICKED

Diciéndole al if que, si el botón de agregar lista se le hizo click, entre dentro del if. A continuación, agregue otro if igual, pero para su botón de ver lista.

8. Dentro del if del botón de agregar a la lista obtenga el texto dentro de su edit control y después coloque lo siguiente:

SendMessage(GetDlgItem(“handler de tu dialogo con la lista”, ”Id de la lista”), LB_ADDSTRING, NULL,(LPARAM) “tu variable con el texto” );

Con la finalidad de agregarle a la lista de tu Dialogo 2 un elemento con lo que le hayas ingresado al edit control de tu Dialogo 1. Después de haber hecho esto, en el if de tu botón de ver lista pon el Código que utilizas mostrar una ventana, pero con el handler de tu Dialogo 2 y observa los resultados.

## XI – GRAFICACIÓN EN LA VENTANA PRINCIPAL, SPRITES.

1. ¿Qué es un BITMAP?
2. ¿Qué es un picture control?
3. 3.- crea un nuevo proyecto con WinAPI y en la pantalla principal coloca un picture control en el cual muestres una imagen BITMAP.
4. Agrega un botón a la pantalla principal que abra un diálogo para seleccionar un archivo que muestre en el picture control el BITMAP seleccionado.
5. ¿Qué es un sprite?
6. ¿Qué es el canal alfa en una imagen?
7. En un proyecto con WinApi dibuja una imagen bitmap y posteriormente utiliza una imagen del canal alfa de esa imagen para darle transparencia.

## XII – GAMEPAD, MOVIMIENTO DE SPRITES CON EL GAME PAD.

1. ¿Qué es un gamepad?
2. ¿Qué es la “deadzone” de un gamepad?
3. Crea un proyecto que incluya la API “XInput”, agrega una imágen BITMAP de fondo para la pantalla principal y sobre ella muestra tu sprite.
4. usando un gamepad y las funciones de XInput asigna al D-pad del mando una manera de manipular la posición en la que el sprite será dibujado.

## XIII – INTRODUCCIÓN AL OPENGL, CONSTRUCCIÓN DE PRIMITIVOS, DESPLIEGUE DE PRIMITIVOS ALÁMBRICOS Y SÓLIDOS.

1. ¿Qué es OpenGL?
2. ¿Qué es un Shader?
3. ¿Qué es un Vertex Shader?
4. ¿Qué es un Pixel Shader?
5. 3- En un proyecto de OpenGL, define un arreglo de tres vértices [V1(-0.5,-0.5) V2(0.5,-0.5) V3(0,0.5)] para crear un triángulo y dibújalo en pantalla.
6. intercambia el orden en el que se declaran los vértices de modo que V1 sea el tercero y V3 sea el primero, observa cómo esto afecta al dibujar el triángulo y toma notas de lo observado.
7. Modifica la posición de los vértices del polígono para que sean las siguientes [V1(-0.5, -0.5) V2(0.5, -0.5) V3(0, 0.5)] y aumenta el número de vértices a graficar y agrega tres más y asigna las siguientes coordenadas [V4(0.5, 0.5) V5(-0.5, 0.5) V6(-0.5, -0.5)] y observa la figura creada.
8. Modifica el arreglo de vértices para definir las coordenadas en un entorno 3d (coordenadas X, Y, Z) basándose en la figura creada con anterioridad, crea un cubo. Recuerda que para dibujar apropiadamente un cubo el orden en que se conectan los vértices debe ser en contra de las manecillas del reloj.

## XIV – ANIMACIÓN DE PRIMITIVOS.

¿Qué es una matriz de proyección?

1. ¿Qué es una matriz de traslación?
2. ¿Qué es una matriz de rotación?
3. Utilizando el cubo que fue creado con anterioridad, haremos que en el código la matriz del modelo pase por una función que trasladará nuestro cubo a una nueva posición.
4. Prueba trasladar el cubo a distintas posiciones y realiza observaciones para orientarte en las posiciones positivas y negativas de cada eje de las coordenadas.
5. Dentro de la región de código en la que se dibuja el cubo (que se realiza cada frame), agrega el código correspondiente para que el cubo realice una traslación de ida y vuelta en un rango específico sobre uno de sus ejes.
6. Realiza lo mismo con algún otro de los ejes del cubo para que se traslade por dos ejes a la vez y observa el resultado.
7. Regresa el cubo a su posición original sin que se traslade, ahora prueba rotar la matriz del objeto sobre uno de sus ejes.
8. Coloca dentro de la región de dibujar el código para que el cubo esté constantemente rotando sobre un eje en específico.

## XV – LECTOR DE TICKS, MEDICIÓN DE UN INTERVALO DE CÓDIGO.

1. ¿Qué es un tick?
2. ¿De qué depende la cantidad de ticks por segundo?
3. Cree un nuevo proyecto que incluya la librería “time.h” y escriba una función que cuente la cantidad de números primos que son menores a 100,000, dentro del main iguala una variable a la función “clock()” justo antes de llamar la función para contar los números primos y después de llamar la función, iguala la variable previamente declarada a la función “clock()” menos la misma variable.
4. Posteriormente imprima en pantalla la variable y la misma variable a través de la función CLOCKS_PER_SEC para mostrar el tiempo en segundos que tomó la computadora en llevar a cabo los cálculos de la función.
5. Crea una función que cuente la cantidad de números que forman parte de la secuencia Fibonacci y que también son menores a 100,000.
6. Así como con la función anterior, imprima en pantalla la cantidad de clicks y los segundos que tardó la computadora en realizar el cálculo, compare los resultados con el punto anterior.