**Nombre: Kevin de Jesús Velázquez Cabrera** _____ **_No. de Matrícula.: ZAP473**_____

**Materia:** _Fundamentos de Programación _____ Grupo**: Dev 22-1** _____ Turno**: Matutino** _____

**Carrera:** Desarrollo de Software Interactivo y Videojuegos _____

**Tema:** Aplicación de Escritorio de Windows **No: R.12**____

**Fecha propuesta:** 05/12/2021_____ **Fecha de Entrega:** 05/12/2021_____

**Escuela: Instituto Universitario Amerike**_____ **Plantel: Guadalajara**_____

**Calle**: C. Montemorelos _ **No:** 3503__ **Colonia:** Rinconada de la Calma_ **C.P.:** 45080 _____

**Teléfono:** 33 3632 6100 _____ **Ciudad:** Zapopan_____

**Logotipo personal**

_____

**Firma del alumno (a)**

**Logotipo (de la escuela)**

_____

**Firma de revisión fecha**

| *Qué se evalúa:* | 10 pts. | 7 pts. | 4pts. | Pts. |
|---|---|---|---|---|
| Entrega electrónica | Es en tiempo y forma al iniciar la clase. (1 pts.) | Después de 30 minutos de iniciada la clase. (.7 pts.) | Al minuto 40. (Posteriormente ya no se reciben) (.4pts.) | |
| *Del formato.* | Cumple con todos los elementos solicitados. (1 pts.) | No cumple con dos elementos solicitados. (.7 pts.) | No cumple con tres o más elementos solicitados. (.4pts.) | |
| *La ortografía.* | Tiene dos errores ortográficos. (1 pts.) | Tiene de tres a cuatro errores ortográficos. (.7 pts.) | Tiene cinco o más errores ortográficos. (.4pts.) | |
| *Del tema y objetivo.* | La teoría y ejemplos corresponden al tema tratado. (1 pts.) | La teoría o ejemplos no corresponden al tema tratado. (. 7 pts.) | La teoría y ejemplos no corresponden al tema tratado. (.4pts.) | |
| *El programa y los cálculos.* | Los parámetros y componentes corresponden al 100% de lo planeado. (1 pts.) | El programa arroja un error o componente no corresponden al 100% de lo planeado. (7 pts.) | El programa arroja dos errores o componentes no corresponden al 100% de lo calculado. (.4pts.) | |
| *Diagramas.* | Los diagramas a bloques, de flujo y esquemáticos son acorde al de la práctica y siguen una secuencia lógica. (1 pts.) | Los diagramas a bloques, o de flujo o esquemáticos no son acorde al de la práctica y o no siguen una secuencia lógica. (.7 pts.) | Los diagramas a bloques, de flujo y esquemáticos no son acorde al de la práctica y o no siguen una secuencia lógica. (.4pts.) | |
| *La tabla de valores.* | Los valores calculados y medidos presentan una desviación máxima del 10%. (1 pts.) | Los valores calculados y medidos presentan una desviación máxima del 15%. (. 7 pts.) | Los valores calculados y medidos presentan una desviación máxima del 20%. (.4pts.) | |
| *Las observaciones y conclusiones.* | Son específicas y congruentes con la práctica. (1 pts.) | Las observaciones o conclusiones son específicas y congruentes con la práctica. (.7 pts.) | Las observaciones y las conclusiones no son específicas y congruentes con la práctica. (.4pts.) | |
| *Bibliografía.* | Es acorde al (los) tema (s) tratado (s) y está completa (1 pts.) | Es acorde a algún (os) tema (s) tratado (s), le falta algún elemento que la conforman (.7 pts.) | No es acorde al (los) tema (s) tratado (s), le faltan 2 elementos que la conforma (.4pts.) | |
| *Fuentes de consulta.* | Es acorde al (los) tema (s) tratado (s) (1 pts.) | Es acorde a algún (os) tema (s) tratado (s) (.7 pts.) | Es acorde a algún (los) tema (s) tratado (s) (.4pts.) | |

**Índice**

## Teoría

En el desarrollo de esta práctica se ha generado una aplicación básica de Windows usando el formato Windows API

## Windows API

Windows API es una aplicación de programación por interfaz (también conocida como Win32 API, Windows Desktop API y Windows Classic API) está basado en el lenguaje C para crear aplicaciones de Windows. Existe desde la década de 1980 y se ha usado para crear aplicaciones Windows durante décadas. Se han creado marcos de trabajo más avanzados y fáciles de programar sobre Windows API. Por ejemplo, MFC, ATL o .NET frameworks. Incluso el código más moderno Windows Runtime para UWP y aplicaciones de la Tienda escritos en C++/WinRT usan Windows API en su interior.

Una principal ventaja de usar Windows API es que se pueden desarrollar aplicaciones que corran y se ejecuten exitosamente en todas las versiones de Windows, a la par que se toman las características, capacidades y funcionalidades de cada versión en específico.

Windows API puede ser divido en diferentes áreas:

- Servicios de Base
- Seguridad
- Gráficos
- Interfaz de Usuario
- Multimedia
- Windows Shell
- Networking

El Servicio de Base provee acceso a los recursos fundamentales en Windows. Esto incluye archivos de sistema, dispositivos, procesos, registros o manejo de errores. La Seguridad proporciona funciones, interfaces, objetos y otros elementos de programación para la autenticación, autorización, criptografía y otras actividades relacionadas con la seguridad.

El subsistema de Gráficos dispone funcionalidades de salida de información gráfica hacia los monitores y otros dispositivos de salida.
La Interfaz de Usuario presenta la funcionalidad de crear ventanas y controles en las aplicaciones.

El componente Multimedia brinda herramientas para el trabajo con video, sonido y entrada de dispositivos.
Las características de la interfaz de Windows Shell permiten que las aplicaciones tengan acceso a la funcionalidad proporcionada por el sistema operativo de interpretación por comandos.
Las características de servicios Network da acceso a las capacidades de red del sistema operativo de Windows.

Windows API es entonces una especificación abstracta de la interfaz de programación para el sistema operativo de Windows. Consiste en funciones de declaración, uniones, estructuras, tipos de datos, macros, constantes y otros elementos de programación. Windows API es descrito principalmente por el MSDN (Microsoft Developer Network) y reside en las cabeceras de Windows en C. La implementación oficial de las funciones de Windows API está localizada en las bibliotecas dinámicas (DLLs). Algunos ejemplos de estas bibliotecas son `kernel32.dll`, `user32.dll`, `gdi32.dll` o `shell32.dll` en el sistema de directorios de Windows.

En cuanto a la creación de Aplicaciones de Windows existen dos conceptos fundamentales para su correcta implementación. La función WinMain y la función WndProc.

Las aplicaciones de escritorio de Windows requieren las bibliotecas `<windows.h>` y `<tchar.h>`. Esta última define el macro TCHAR que resuelve en última instancia a `wchar_t` si el símbolo UNICODE está definido en el proyecto, de lo contrario se resuelve a `char`.

```
#include <windows.h>
#include <tchar.h>
```

**WinMain**

Como cualquier otra aplicación en C o C++ las aplicaciones de escritorio de Windows deben tener una función `main` como punto de partida. Sin embargo, las funciones de punto de entrada para las aplicaciones de Windows son un tanto distintas y se denominan `WinMain` de manera convencional. Las funciones `WinMain` deben inicializar la aplicación, desplegar una ventana principal, además de presentar un bucle de recuperación y envío de mensajes que es la estructura de control de nivel superior para el resto de la ejecución de la aplicación. Al igual que debe finalizar el bucle de mensajes cuando se reciba la función `WM_QUIT`. En este punto la función `WinMain` debe salir de la aplicación

La sintaxis de la función `WinMain` es la siguiente:

```
int __clrcall WinMain(
  [in] HINSTANCE hInstance,
  [in] HINSTANCE hPrevInstance,
  [in] LPSTR     lpCmdLine,
  [in] int       nShowCmd
);
```

A continuación, se detallan sus parámetros:

`[in] hInstance`: Identificador de la instancia actual de la aplicación.

`[in] hPrevInstance`: Identificador de la instancia anterior de la aplicación, este parámetro es siempre NULL.

`[in] lpCmdLine`: Comando de línea de la aplicación, excluye el nombre del programa.

`in] nShowCmd`: Controles de cómo la ventana es mostrada.

En el siguiente ejemplo se muestra el uso de la función `WinMain`.

```
#include <windows.h>

int APIENTRY WinMain(HINSTANCE hInst, HINSTANCE hInstPrev, PSTR cmdline, int
cmdshow)
{
    return MessageBox(NULL, "hello, world", "caption", 0);
}
```

**WndProc**

Junto con la función `WinMain`, cada aplicación de escritorio de Windows debe incluir una función de procedimiento. Esta función es típicamente declarada `WndProc`. Dentro de esta función se escribe el código que maneja los mensajes que la aplicación recibe de Windows cuando un evento ocurre. Un ejemplo de estos eventos es cuando el usuario decide interactuar con el botón OK de la aplicación, Windows enviará un mensaje, éste se procesará dependiendo del código descrito dentro de la función `WndProc` que hará lo más apropiado para la situación en específico. Estos procesos son llamados Manejo de Eventos.

La sintaxis de de la función `WndProc` es la siguiente:

```
LRESULT CALLBACK WindowProc(
  _In_ HWND   hwnd,
  _In_ UINT   uMsg,
  _In_ WPARAM wParam,
  _In_ LPARAM lParam
);
```

A continuación, se detallan sus parámetros:

`hwnd [in]`: Referencia a la ventana.
`uMsg [in]`: Referencia al mensaje.
`wParam [in]`: Información adicional del mensaje. Los contenidos de este parámetro dependen del valor de `uMsg`.
`lParam[in]`: Información adicional del mensaje. Los contenidos de este parámentro dependen de igual forma de `uMsg`.

## Objetivos

Crear una aplicación tradicional del sistema operativo de Windows que al ser ejecutada se despliegue una ventana en la interfaz del monitor, presentando el mensaje “Hello, Windows Desktop!”.

## Cálculos

En el desarrollo de esta práctica no se ha requerido el uso de cálculos matemáticos para la comprobación de los resultados.

## Diagramas

Diagrama de flujos para la creación de un proyecto de escritorio de Windows en Visual Studio.

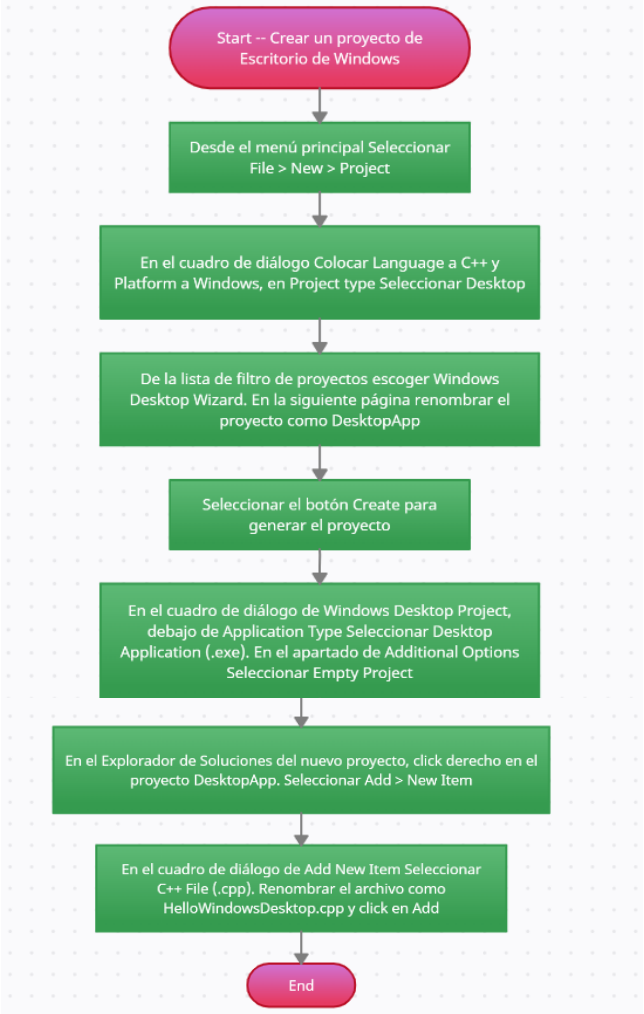


*Ilustración 1* Proceso de creación de un proyecto de Windows Desktop al que se le añade un archivo fuente .cpp.

Imagen del menú de atajos del Explorador de Soluciones en donde se añade el archivo fuente .cpp

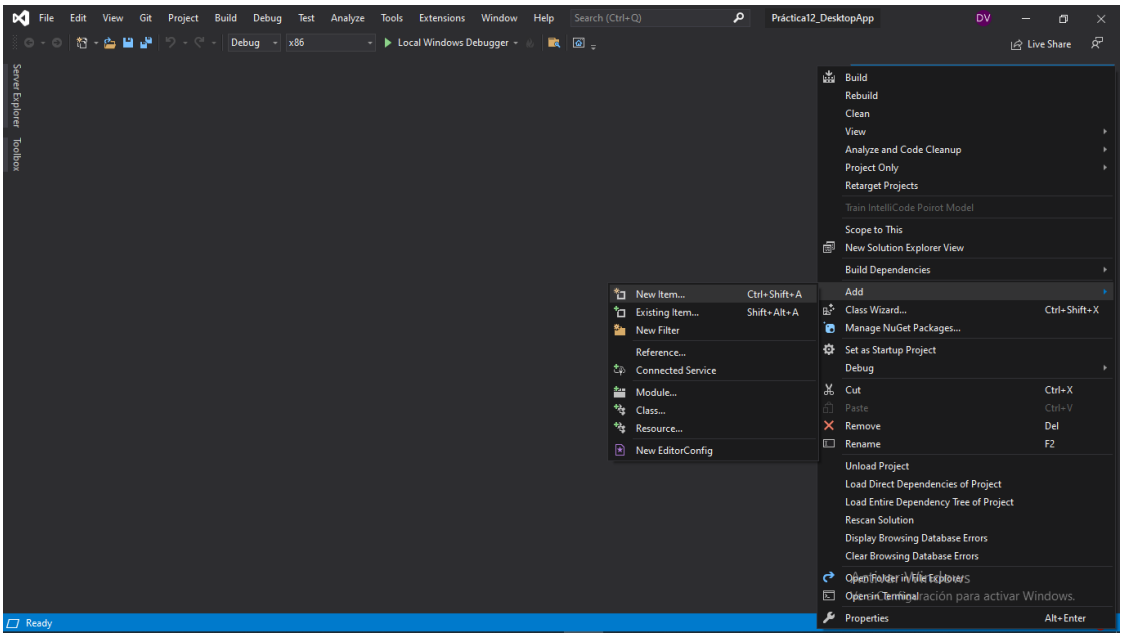


*Ilustración 2* Agregando un nuevo elemento al proyecto.

Imagen del cuadro de diálogo Add New Item, en donde se añade un archivo .cpp y se le cambia el nombre al archivo por “HelloWindowsDesktop.cpp”

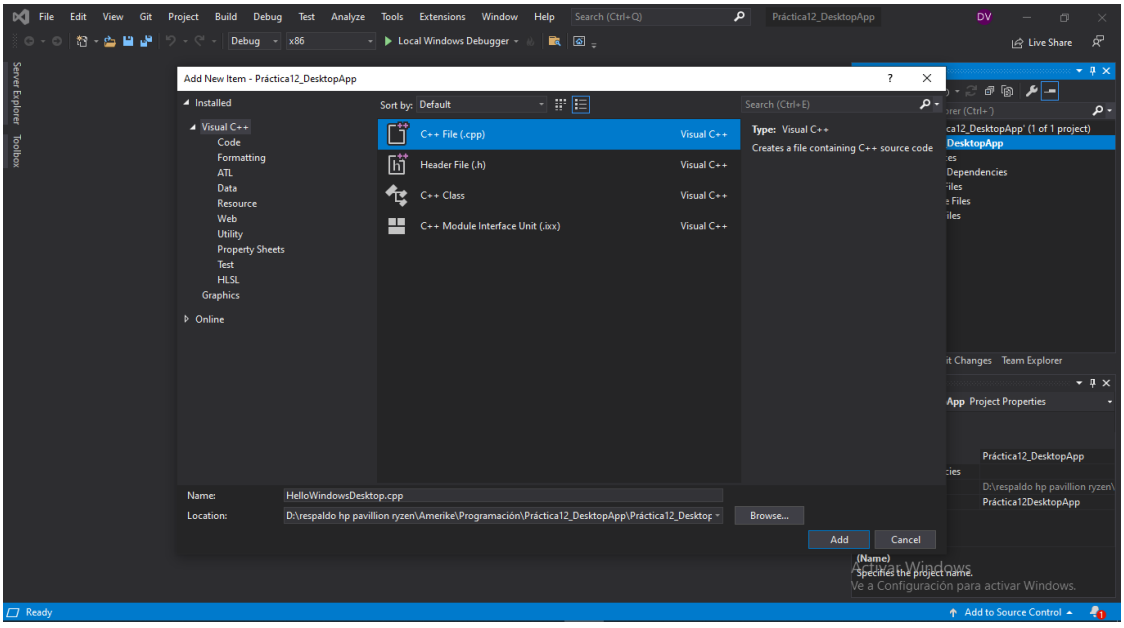


*Ilustración 3* El elemento a añadir al proyecto es un archivo .cpp.

Diagrama de flujos para la creación del código de una aplicación de Windows Desktop.

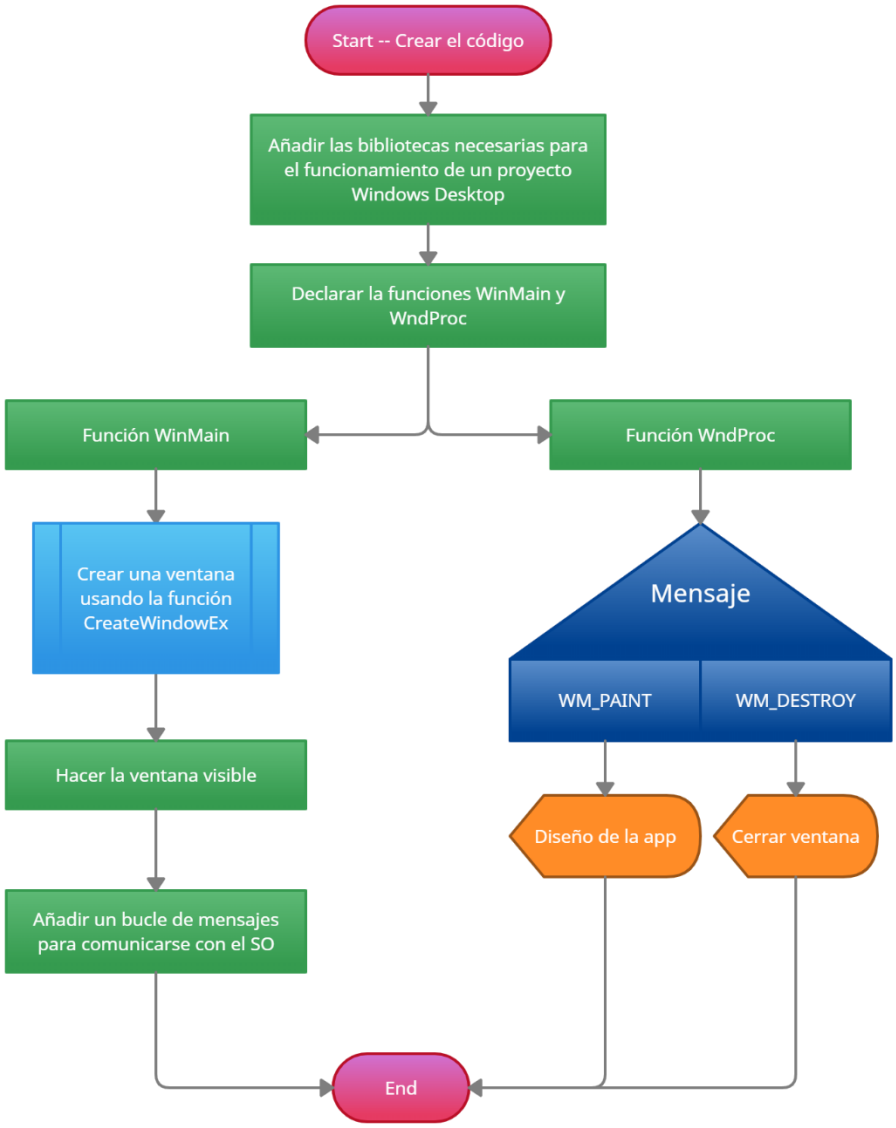


*Ilustración 4* Para la creación del código se deberán hacer dos procesos a la par, primeramente, añadir funcionalidad a la función `WinMain` y seguido a esto, describir la función que realizará `WndProc`.

Como primeros dos pasos en la creación del código se agregan las librerías correspondientes, así como la función `WinMain`:

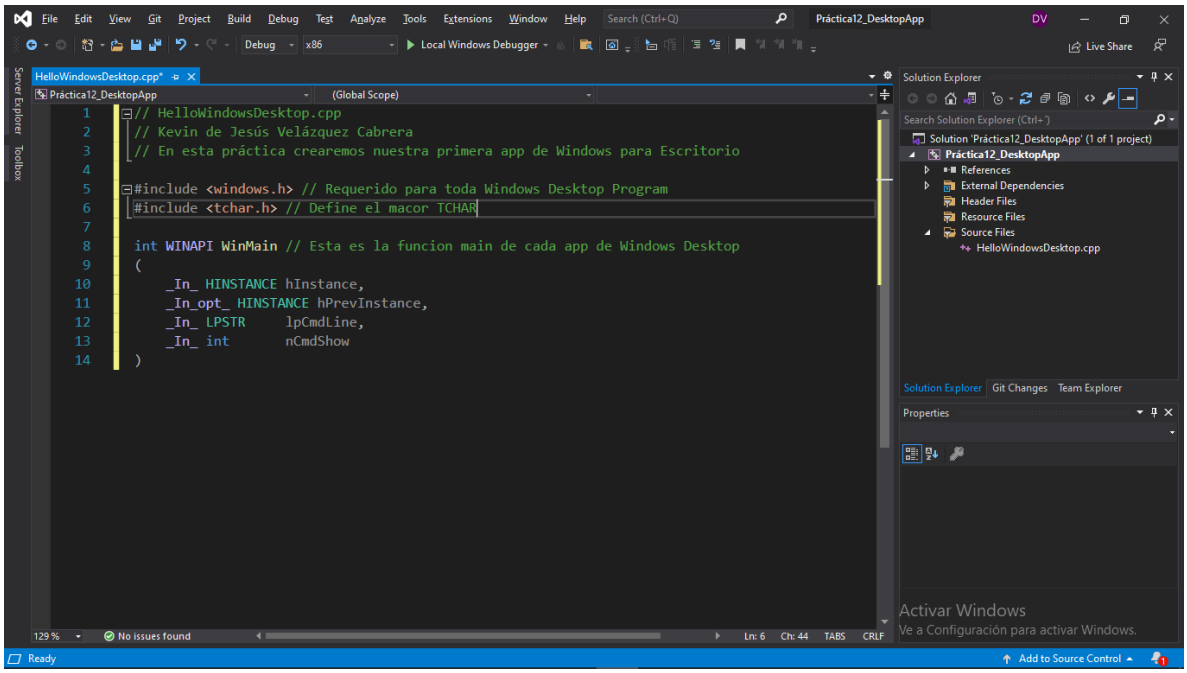


*Ilustración 5* Se incluyen las bibliotecas correspondientes, al igual que la función `WinMain`.

Seguidamente se agrega funcionalidad a la función `WinMain` con una estructura que contendrá información acerca de la ventana.

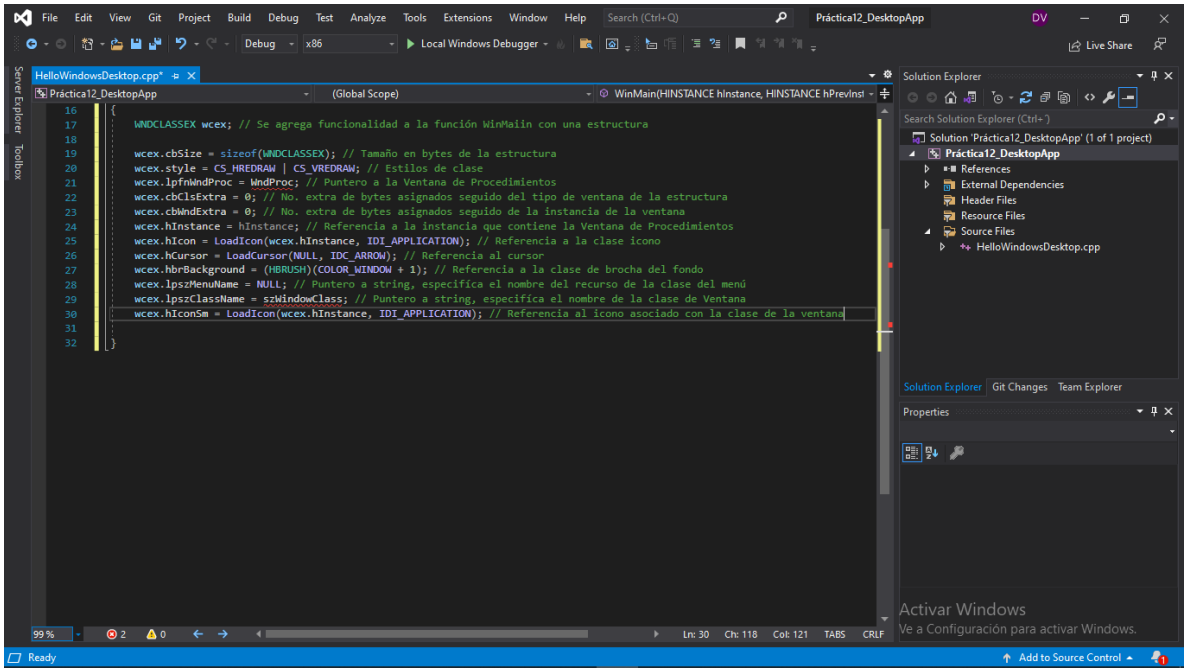


*Ilustración 6* Función `WNDCLASSEX` con la descripción de cada elemento que añade propiedades a la ventana.

Registro de la clase creada en el paso anterior `WNDCLASSEX` con Windows, de esta manera se tendrá conocimiento acerca de la ventana y cómo enviar mensajes a esta.

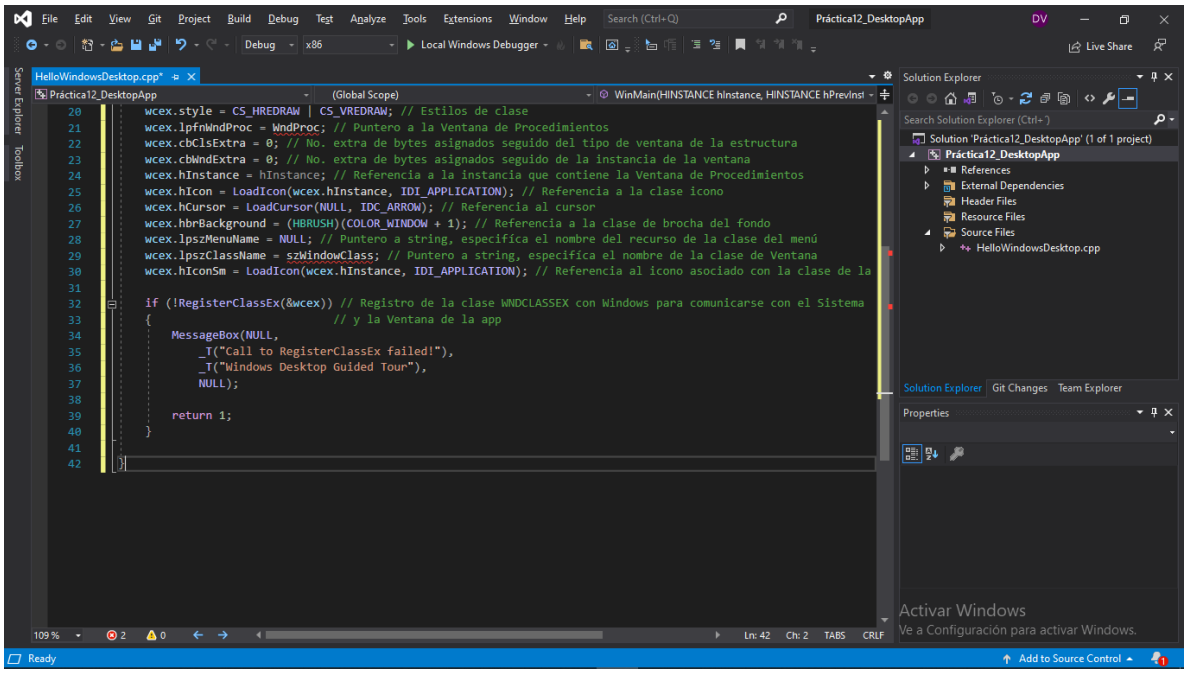


*Ilustración 7* En el proceso de registro de la clase `WNDCLASSEX` con Windows, la aplicación podrá comunicarse con el Sistema Operativo.

Proceso de creación de una ventana. Esta función regresa un `HWND` para hacer referencia a la ventana. Similar a un puntero que Windows usa para mantener el seguimiento de la ventana abierta.

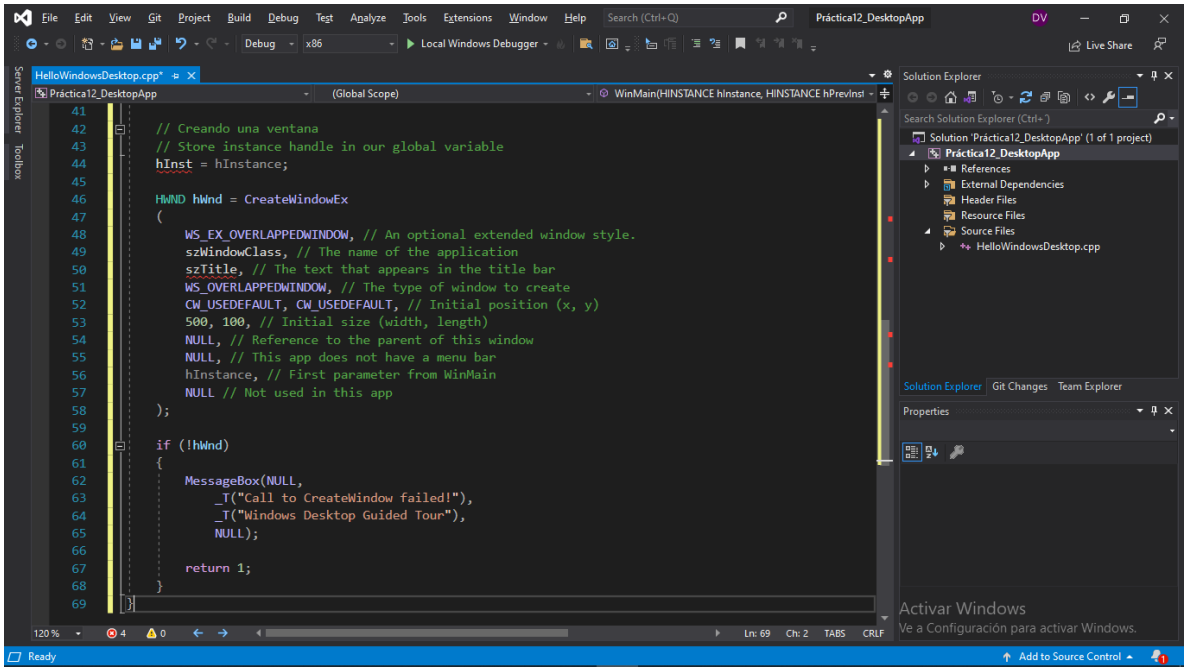


*Ilustración 8* Función `CreateWindowEx` con una descripción de sus elementos.

Una vez que la ventana ha sido creada, se deberá hacer visible.

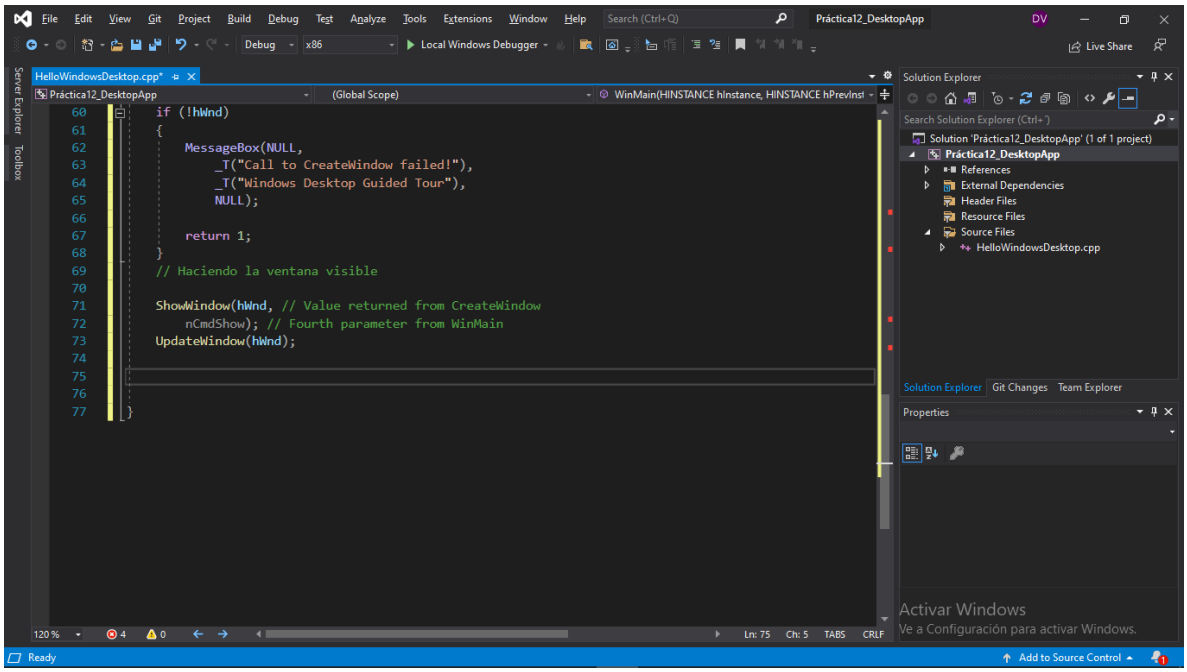


*Ilustración 9* Funciones `ShowWindow` y `UpdateWindow`.

A este punto, la aplicación aún no recibe referencia del mensaje que Windows le manda, así que se agrega un ciclo de mensaje para todos aquellos mensajes que Windows le mande a la ventana. De la misma forma, cuando se implemente la función `WndProc` el ciclo entregará el mensaje para ser referenciado.

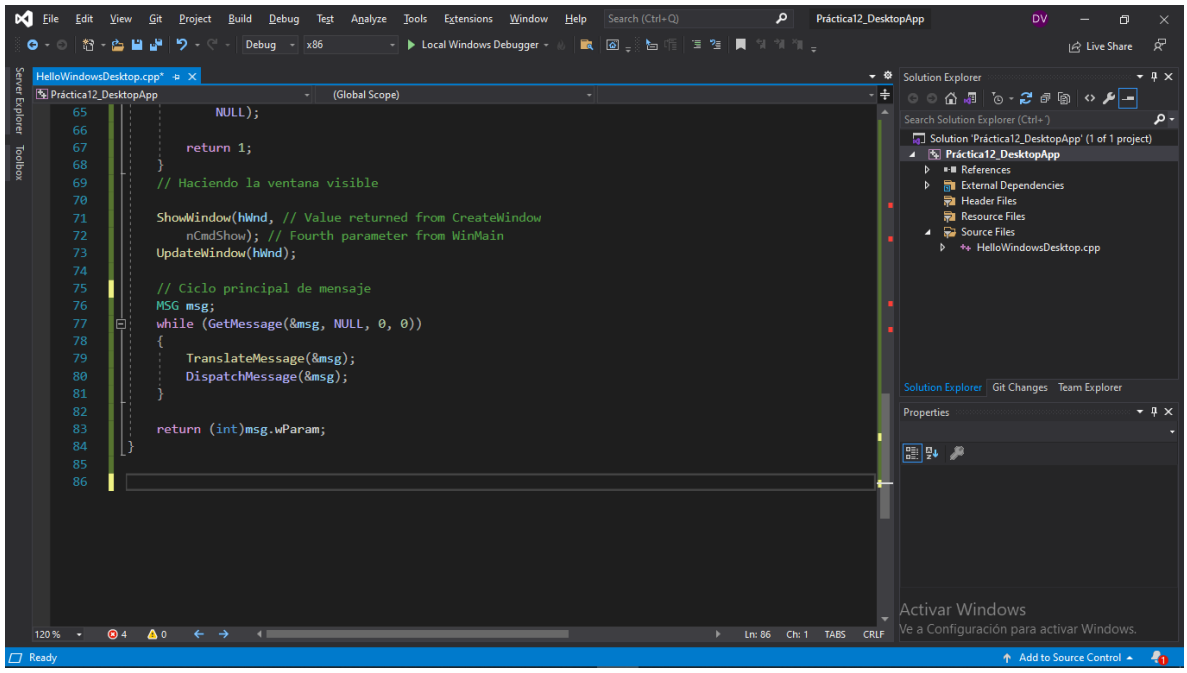


*Ilustración 10* Ciclo de comunicación entre el Sistema Operativo y la aplicación.

Ahora se agregará funcionalidad a la función `WndProc`. La aplicación deberá recibir un mensaje `WM_PAINT` para poder brindar información de cuándo se deberán actualizar algunas partes de ventana desplegada.

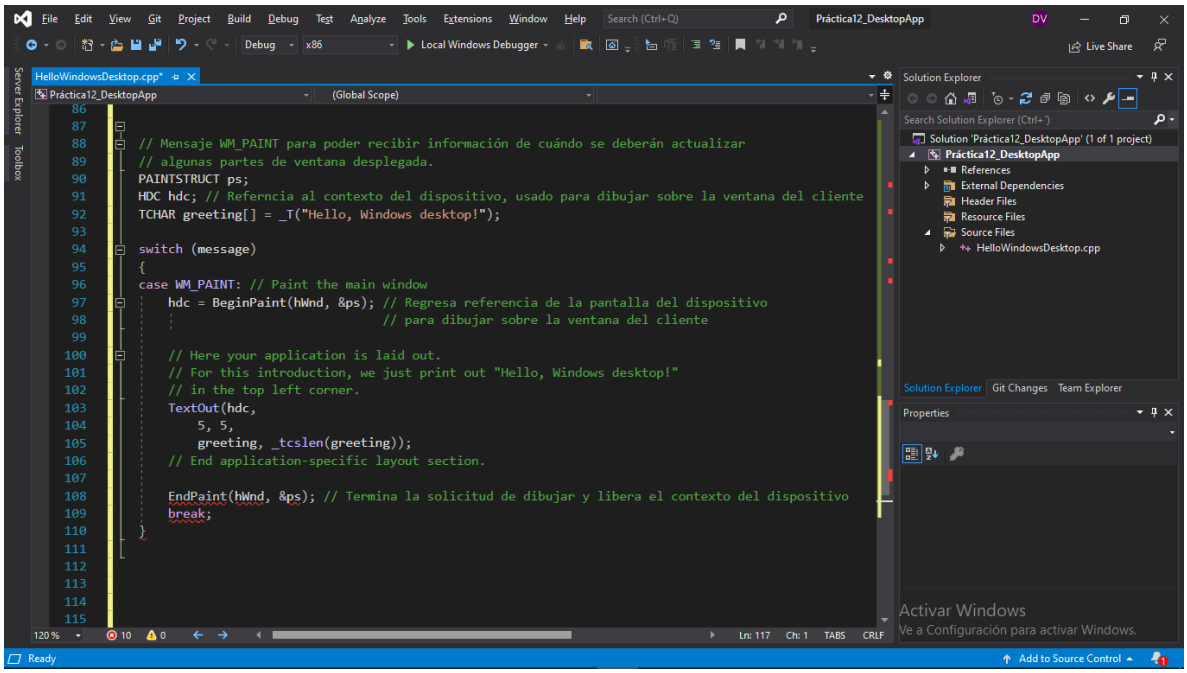


*Ilustración 11* Funcionalidad en `WndProc` que permite actualizar la pantalla de la ventana.

Seguido a esto, se agrega al código el mensaje `WM_DESTROY` para referenciar cuando la ventana es cerrada.

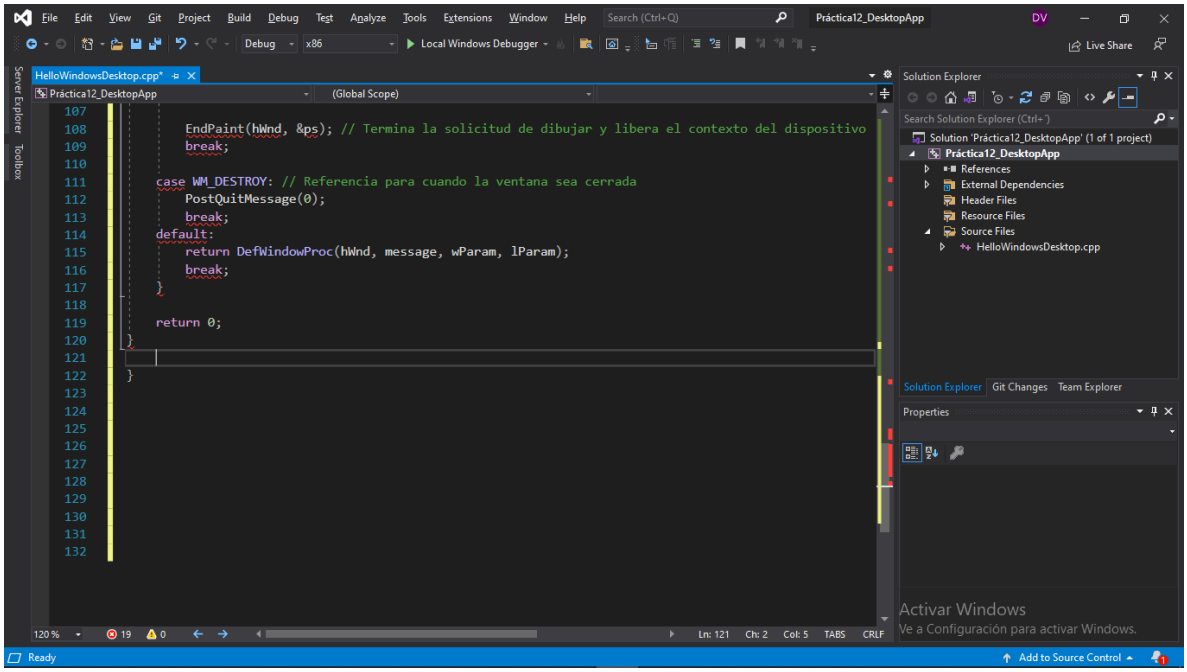


*Ilustración 12* Caso `WM_DESTROY` para comunicar que la ventana ha sido cerrada.

Al igual, se declara formalmente la función `WndProc` al inicio de este bloque de código de mensajes.

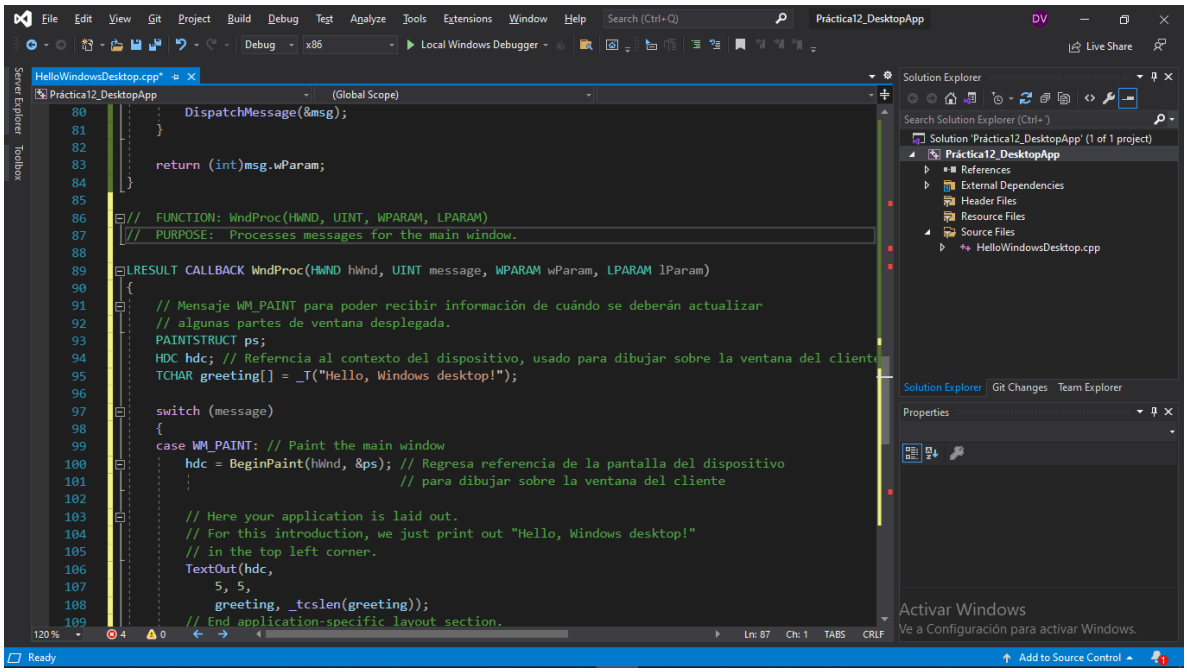


*Ilustración 13* Principio del bloque de código de la función `WndProc`.

Resultado del programa una vez ha sido compilado y ejecutado, generando una ventana de aplicación de Windows:


*Ilustración 14* Resultado de la ejecución del programa Windows Desktop.

**Tabla (comparativa)**

| Tipo de Dato | Observaciones |
|---|---|
| `HWND` | Dentro del marco de este proceso de creación de una aplicación Windows Desktop. En la función `CreateWindowEx` tiene una función similar a un puntero, puesto que brinda información al Sistema Operativo para mantener el seguimiento de la ventana de nuestra aplicación |
| Función `WNDCLASSEX` | Esta función es implementada dentro de `WinMain` para brindarle características como estructura. Haciendo referencias al tamaño en el almacenamiento, a las instancias contenidas en nuestra ventana, íconos, cursor, incluso al nombre del recurso de la clase del menú |
| Mensaje `WM_PAINT` | Este mensaje se agrega dentro de la función `WNDPROC` y es el que indica la frecuencia de actualización en ciertas partes de la ventana de la aplicación |
| Mensaje `WM_DESTROY` | Este mensaje se utiliza dentro de la función `WNDPROC` para hacer referencia de cuándo nuestra aplicación es cerrada, dando indicaciones de cierre del proceso |

## Conclusiones

En la creación de una aplicación típica de Windows Desktop es importante mencionar que, aunque el proceso está basado en lenguaje de programación C++, las funciones principales pueden diferir, esto se hace para que la aplicación interactúe en el entorno ofrecido por Windows. De igual manera, se deberán incluir los header files necesarios para el buen funcionamiento de la aplicación. Además de que las funciones `WinMain` y `WndProc` deberán ser descritas dependiendo de la funcionalidad que se requiera realizar. Por lo que este ejercicio sirve de base para la creación de otras aplicaciones de Windows Desktop.

## Bibliografía

Paul Deitel & Harvey Deitel. (2014). *C++ How to Program* (9th Edition). PHI.

Anónimo. (s. f.). *Principios de Programación. El Lenguaje C*.

## Fuentes de consulta

corob-msft. (s. f.). *Walkthrough: Create a traditional Windows Desktop application (C++)*. Recuperado de: https://docs.microsoft.com/en-us/cpp/windows/walkthrough-creating-windows-desktop-applications-cpp

GrantMeStrength. (s. f.). *Windows API index—Win32 apps*. Recuperado de: https://docs.microsoft.com/en-us/windows/win32/apiindex/windows-api-list

Jan Bodnar. (s. f.). *Introduction to Windows API*. ZetCode. Recuperado de: https://zetcode.com/gui/winapi/introduction/

GrantMeStrength. (s. f.). *WinMain function (winbase.h)—Win32 apps*. Recuperado de: https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-winmain

*WindowProc callback function (Windows)*. (s. f.). Recuperado de: https://docs.microsoft.com/en-us/previous-versions/windows/desktop/legacy/ms633573(v=vs.85)