Lab # API — Assessment Worksheet

**Course Name and Number: IAM302**

**Student Name and Student Id:**

* **Tran Thanh Tuan – SE161095**
* **Diep Anh Vu – SE160365**
* **Shim Jun Woo – SE161064**

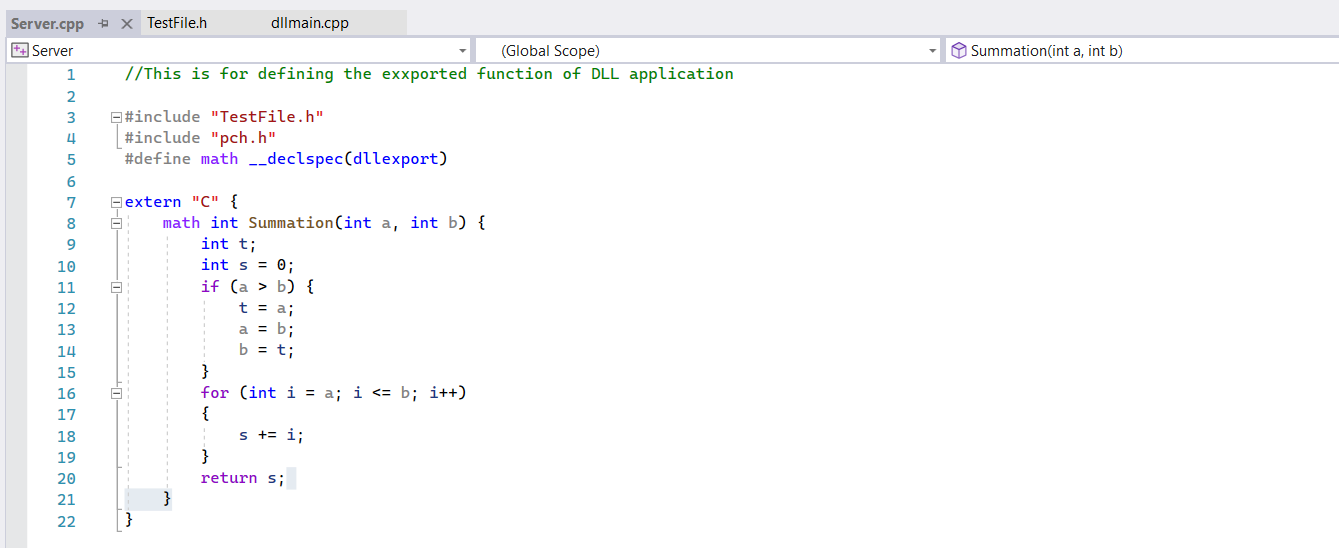
**Instructor Name: Vu Duc Ly**

**Write a program that uses Windows APIs**

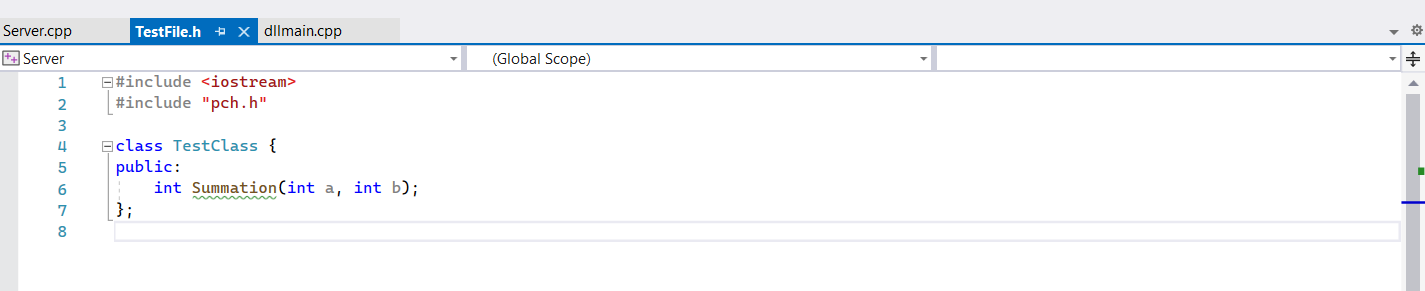
***GetProcAddress***

1. **Code**
2. **Server.cpp**

Server.cpp is used to contain the code to create the summation function.

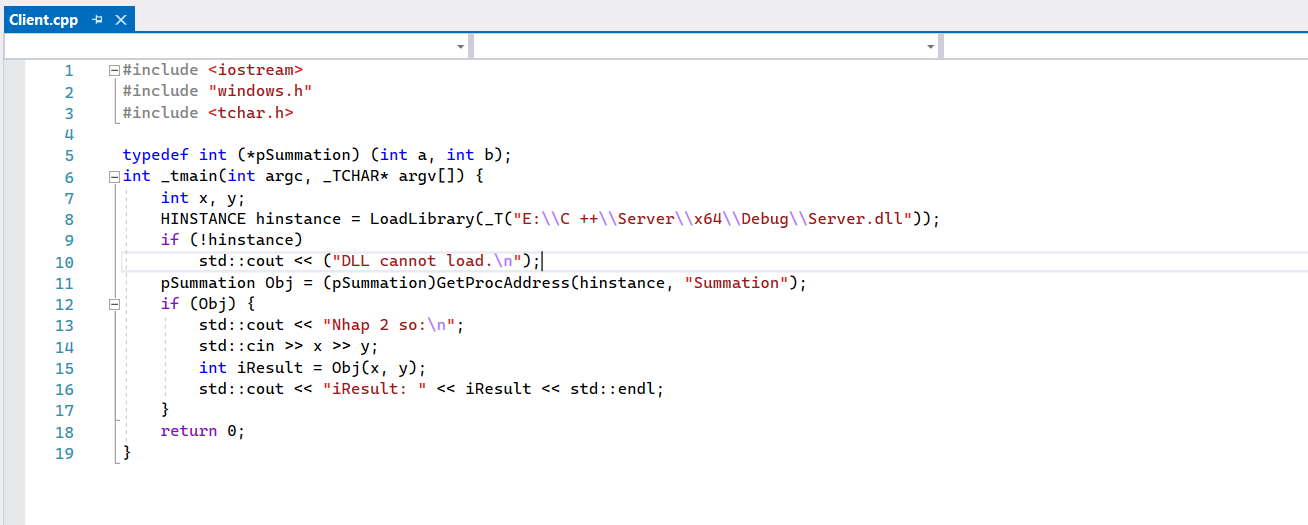
****

1. **TestFile.h**

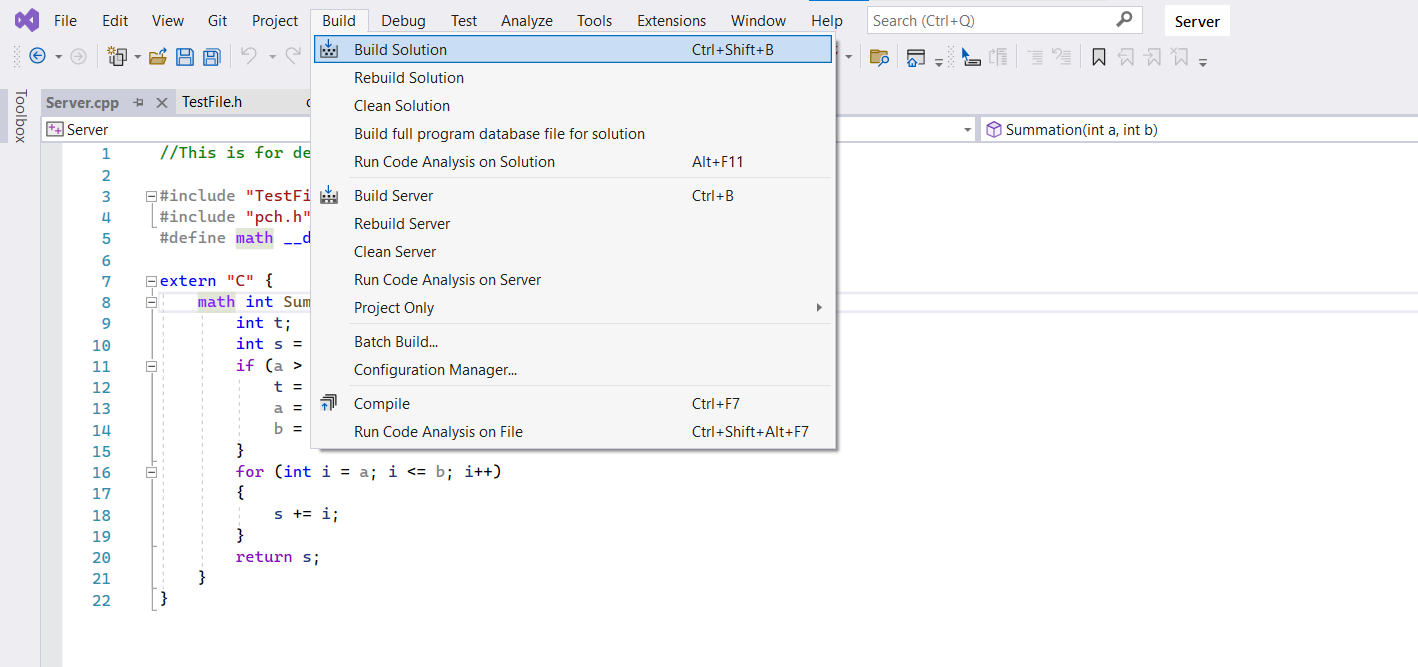
****

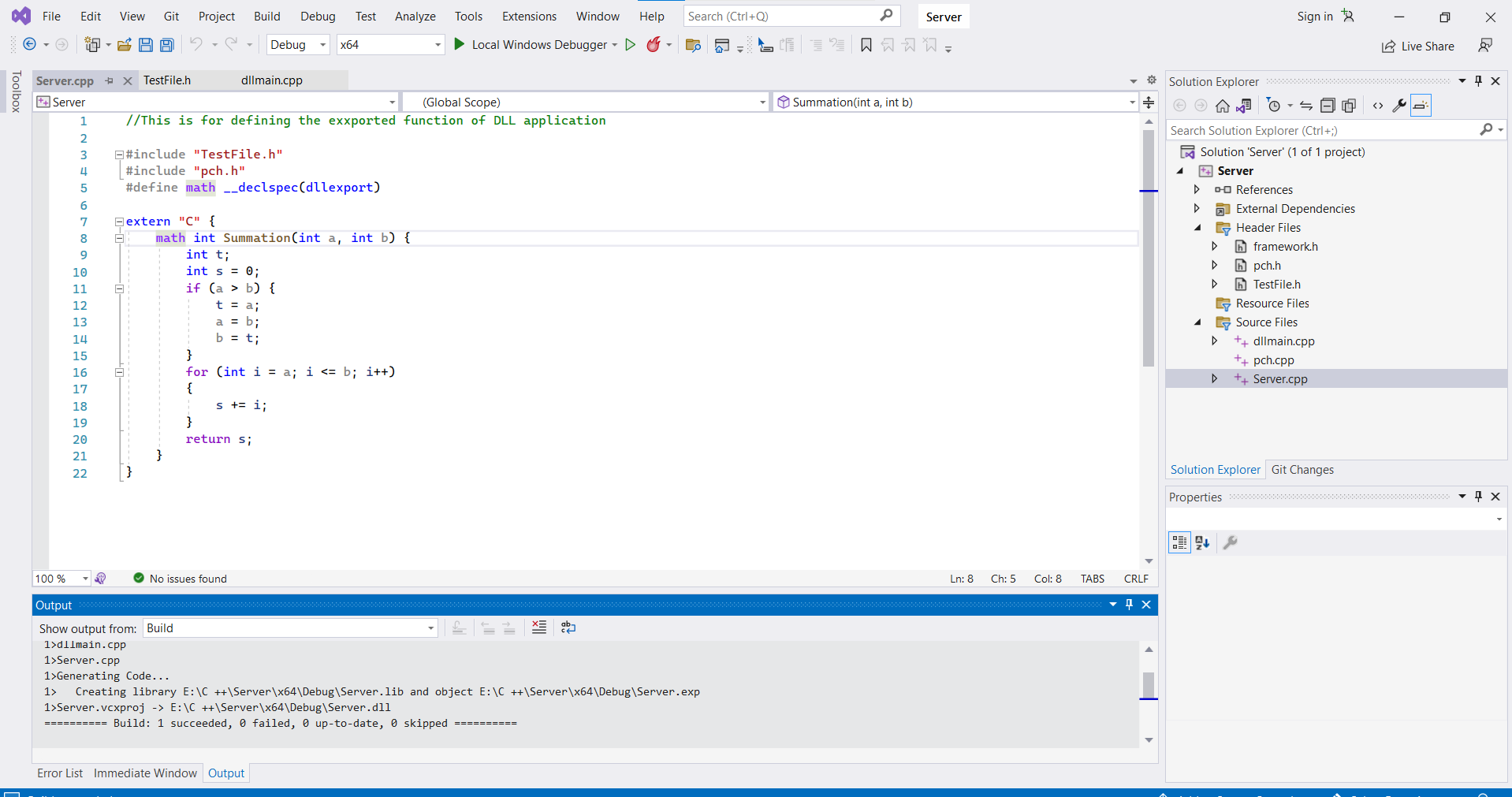
1. **Client.cpp**

LoadLibray will load the Server.dll and GetProcAddress will retrieve the address of an exported function Summation.

****

1. **Running**
2. **Build Server.cpp and TestFile.h**

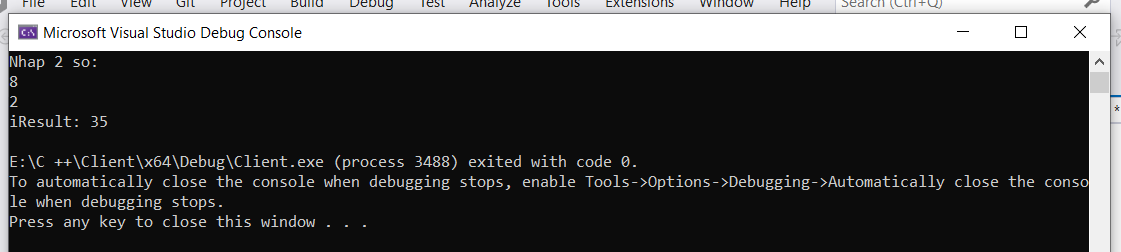
****

****

****

Make sure you copy the right path then running Client.cpp

1. **Running Client.cpp**

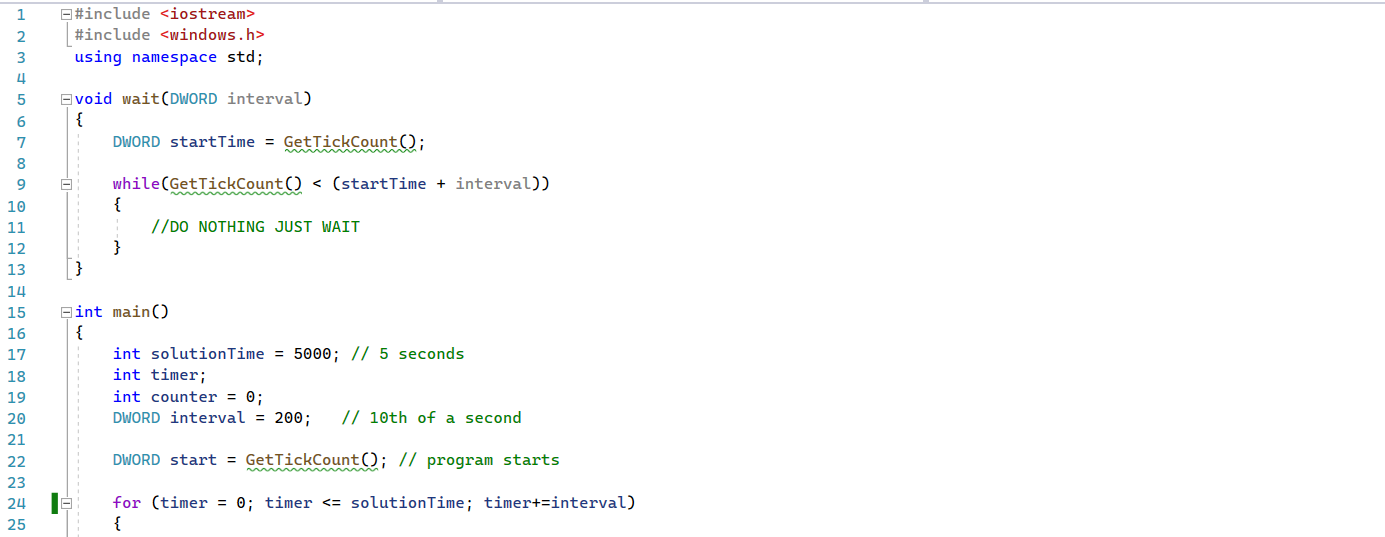
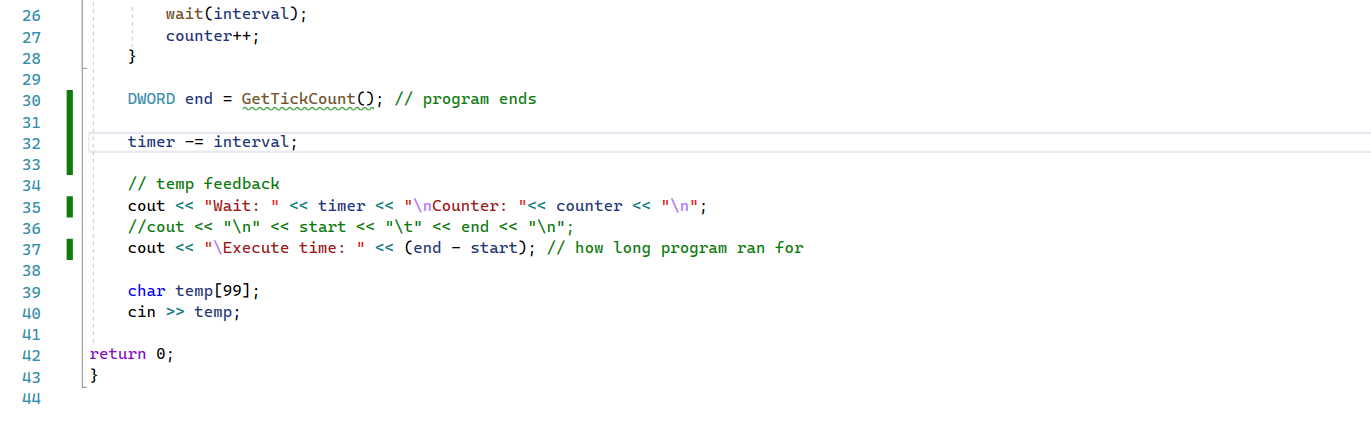
****

***GetTickCount***

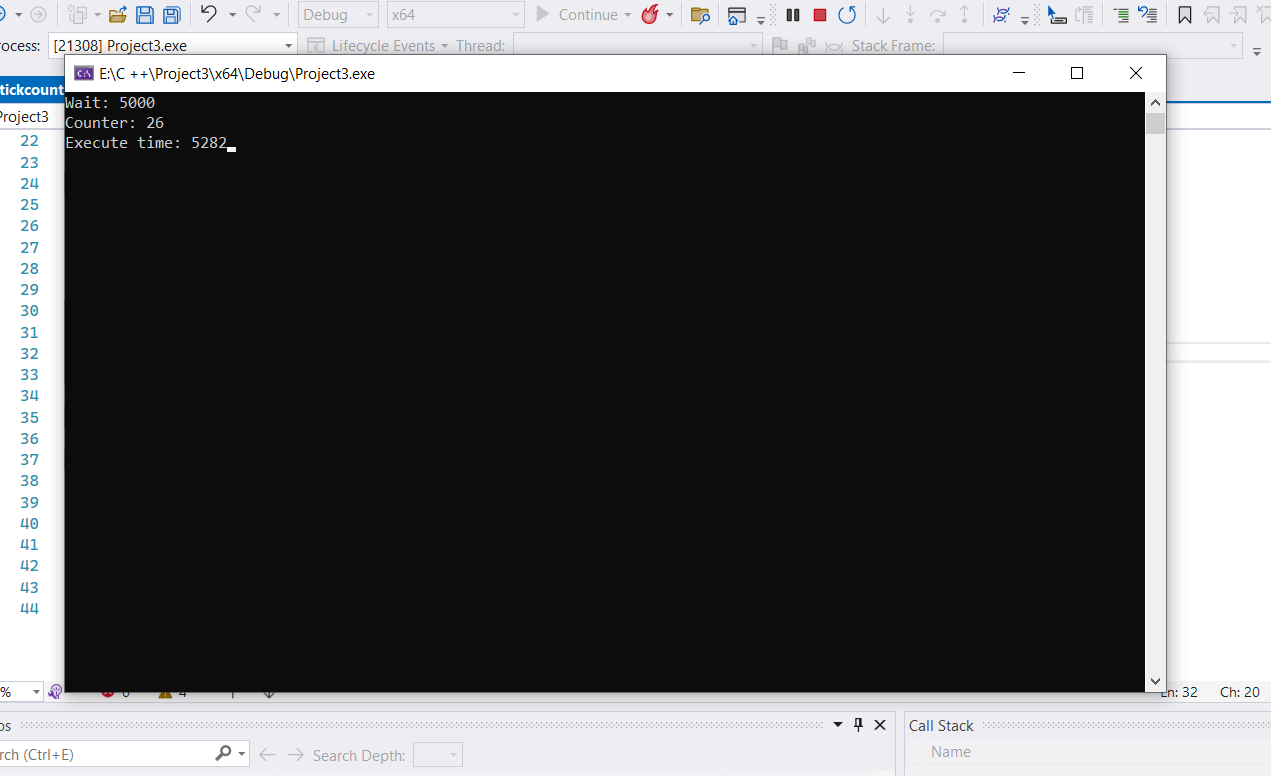
1. **Code**

This code is used to find the execution time of the for function.

The return value is the number of milliseconds that have elapsed since the system was started.

**** ****

1. **Running Getticlcount.cpp**

****

***RegisterHotKey***

Defines a system-wide hot key.

1. **Explaining**

**Syntax C++**

BOOL RegisterHotKey(

   [in, optional] HWND hWnd,

   [in]           int  id,

  [in]           UINT fsModifiers,

   [in]           UINT vk

);

**Parameters**

* **[in, optional] hWnd**

Type: **HWND**

A handle to the window that will receive [WM\_HOTKEY](https://learn.microsoft.com/en-us/windows/desktop/inputdev/wm-hotkey) messages generated by the hot key. If this parameter is **NULL**, **WM\_HOTKEY** messages are posted to the message queue of the calling thread and must be processed in the message loop.

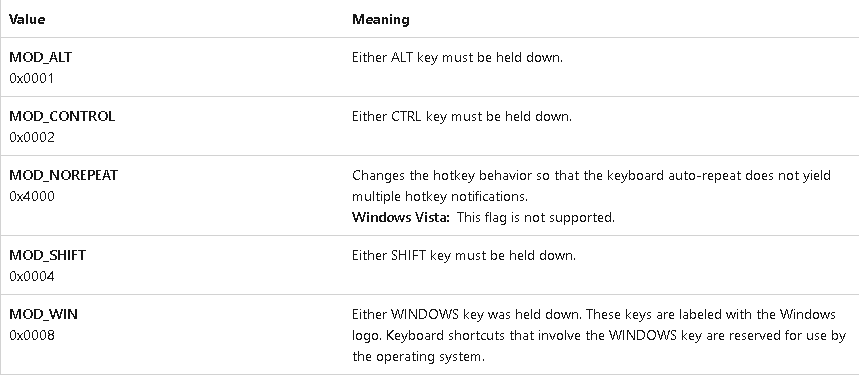
* **[in] id**

Type: **int**

The identifier of the hot key. If the *hWnd* parameter is NULL, then the hot key is associated with the current thread rather than with a particular window. If a hot key already exists with the same *hWnd* and *id* parameters, see Remarks for the action taken.

* **[in] fsModifiers**

Type: **UINT**

The keys that must be pressed in combination with the key specified by the *uVirtKey* parameter in order to generate the [WM\_HOTKEY](https://learn.microsoft.com/en-us/windows/desktop/inputdev/wm-hotkey) message. The *fsModifiers* parameter can be a combination of the following values. 

**[in] vk**

Type: **UINT**

The virtual-key code of the hot key. See [Virtual Key Codes](https://learn.microsoft.com/en-us/windows/desktop/inputdev/virtual-key-codes).

**Return value**

Type: **BOOL**

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call [GetLastError](https://learn.microsoft.com/en-us/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror).

**Remarks**

When a key is pressed, the system looks for a match against all hot keys. Upon finding a match, the system posts the [WM\_HOTKEY](https://learn.microsoft.com/en-us/windows/desktop/inputdev/wm-hotkey) message to the message queue of the window with which the hot key is associated. If the hot key is not associated with a window, then the **WM\_HOTKEY** message is posted to the thread associated with the hot key.

This function cannot associate a hot key with a window created by another thread.

**RegisterHotKey** fails if the keystrokes specified for the hot key have already been registered by another hot key.

If a hot key already exists with the same *hWnd* and *id* parameters, it is maintained along with the new hot key. The application must explicitly call [UnregisterHotKey](https://learn.microsoft.com/en-us/windows/desktop/api/winuser/nf-winuser-unregisterhotkey) to unregister the old hot key.

**Windows Server 2003:**If a hot key already exists with the same *hWnd* and *id* parameters, it is replaced by the new hot key.

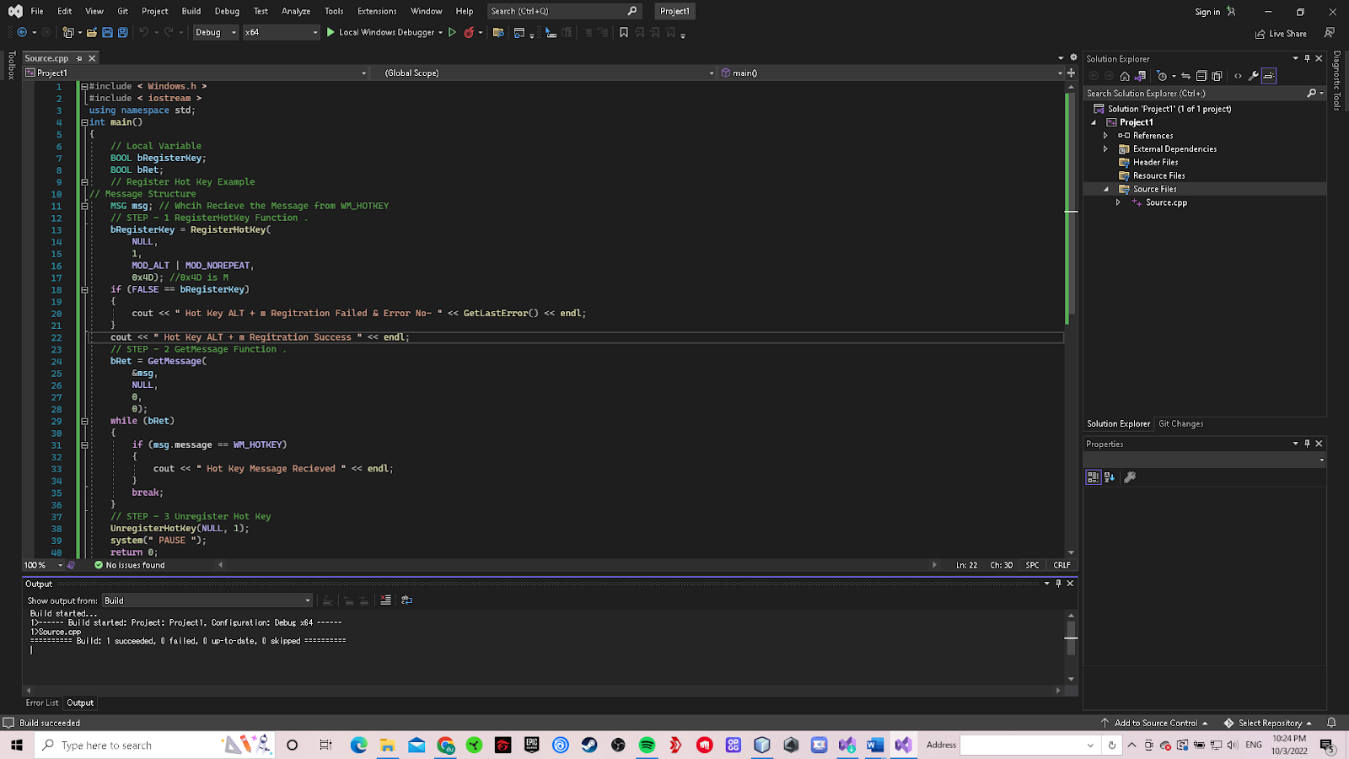
The F12 key is reserved for use by the debugger at all times, so it should not be registered as a hot key. Even when you are not debugging an application, F12 is reserved in case a kernel-mode debugger or a just-in-time debugger is resident.

An application must specify an id value in the range 0x0000 through 0xBFFF. A shared DLL must specify a value in the range 0xC000 through 0xFFFF (the range returned by the [GlobalAddAtom](https://learn.microsoft.com/en-us/windows/desktop/api/winbase/nf-winbase-globaladdatoma) function). To avoid conflicts with hot-key identifiers defined by other shared DLLs, a DLL should use the **GlobalAddAtom** function to obtain the hot-key identifier.

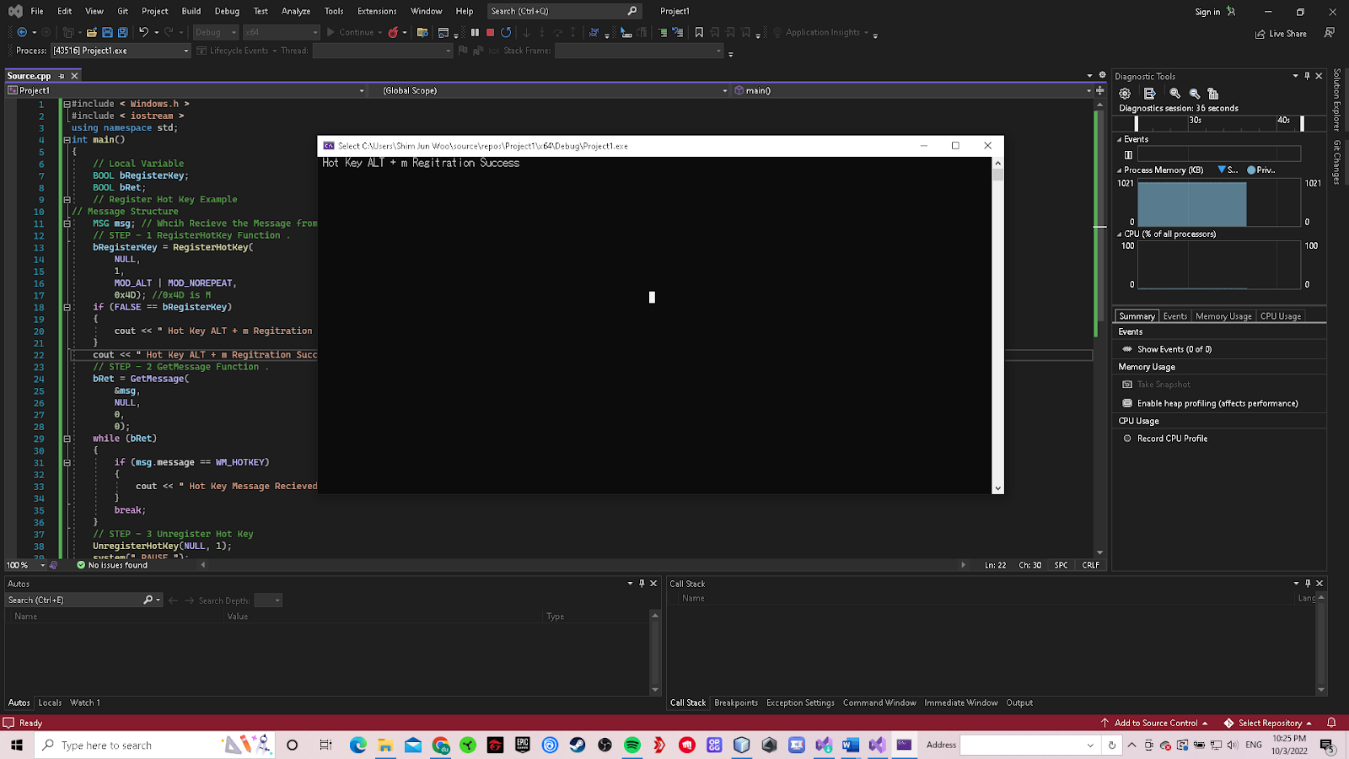
***Examples***

The following example shows how to use the **RegisterHotKey** function with the **MOD\_NOREPEAT** flag. In this example, the hotkey 'ALT+m' is registered for the main thread. When the hotkey is pressed, the thread will receive a [WM\_HOTKEY](https://learn.microsoft.com/en-us/windows/desktop/inputdev/wm-hotkey) message, which will get picked up in the [GetMessage](https://learn.microsoft.com/en-us/windows/desktop/api/winuser/nf-winuser-getmessage) call. Because this example uses **MOD\_ALT** with the **MOD\_NOREPEAT** value for *fsModifiers*, the thread will only receive another **WM\_HOTKEY** message when the 'm' key is released and then pressed again while the 'ALT' key is being pressed down.

1. **Code**



1. **Debugging**

Before press Alt+m: 

After press Alt+m:

