ЛАБОРАТОРНА РОБОТА №3. ДИНАМІЧНЕ КОМПОНУВАННЯ. СТВОРЕННЯ DLL-БІБЛІОТЕК ТА ЇХ ВИКОРИСТАННЯ ПРИ ЯВНОМУ І НЕЯВНОМУ ЗВ'ЯЗУВАННІ. (2 год.)

Мета: Ознайомитись з технологією та оволодіти навиками створення та використання бібліотек динамічного компонування з використанням неявного і явного зв'язування.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Завершальним етапом створення програмного продукту є процес збирання (компонування) завантажувального модуля (.exe – файлу). *Компонуванням* (linking) називають процес створення фізичного або логічного виконуваного файлу (модуля) із набору об'єктних файлів бібліотек для подальшого виконання або під час виконання і вирішення проблеми неоднозначності імен, що виникає при цьому.

У разі створення фізичного виконуваного файлу для подальшого виконання компонування називають *статичним*, коли у такому файлі міститься все потрібне для виконання програми. У випадку створення логічного виконуваного файла під час виконання програми компонування називають *динамічним*, у цьому випадку образ виконуваного модуля збирають "на ходу".

Статичне компонування виконуваних файлів має низку недоліків:

- якщо кілька застосувань використовують спільний код, кожний виконуваний файл міститиме окрему копію цього коду в результаті такі файли займатимуть значне місце на диску і у пам'яті;
- під час кожного оновлення застосування, його потрібно наново перекомпонувати і перевстановити;
- неможливо реалізувати динамічне завантаження програмного коду під час виконання.

Для вирішення цих і подібних проблем було запропоновано концепцію *динамічного компонування* із використанням динамічних або розділюваних бібліотек.

Динамічна бібліотека (англ. Dynamic-Load Library — динамічно завантажувана бібліотека) - набір функцій, скомпонованих разом у вигляді бінарного файлу, який може бути динамічно завантажений в адресний простір процесу, що використовує ці функції. *Динамічне завантаження* – завантаження під час виконання процесу. *Динамічне компонування* – компонування образу виконуваного файлу під час виконання процесу із використанням динамічних бібліотек.

До переваги використання динамічних бібліотек, слід віднести:

- оскільки бібліотечні функції містяться в окремому файлі, розмір виконуваного файлу стає меншим;
- якщо динамічну бібліотеку використовують кілька процесів, у пам’ять завантажують лише одну її копію, після чого сторінки коду бібліотеки відображаються в адресний простір кожного з цих процесів;
- оновлення застосування може бути зведене до встановлення нової версії динамічної бібліотеки без необхідності перекомпонування тих його частин, які не змінилися;
- динамічні бібліотеки дають змогу застосуванню реалізовувати динамічне завантаження модулів на вимогу;
- динамічні бібліотеки дають можливість спільно використовувати ресурси застосування;
- оскільки динамічні бібліотеки є двійковими файлами, можна організувати спільну роботу бібліотек, розроблених із використанням різних мов програмування.

Використання динамічних бібліотек не позбавлене недоліків:

- динамічне зв’язування сповільнює завантаження застосування. Що більше таких бібліотек потрібно процесу, то більше файлів треба йому відобразити у свій адресний простір під час завантаження, а відображення кожного файлу забирає час;
- при відсутності спільного використання динамічної бібліотеки іншими застосування зовнішня пам’ять може використовуватися не ефективно. На відміну від статичного зв’язування, коли зі загальної бібліотеки вибираються тільки ті функції, що використовуються застосування, при використанні динамічного зв’язування до застосування необхідно додавати повну версію бібліотеки, навіть, якщо використовуються тільки декілька функцій. При значних обсягах бібліотеки втрати пам’яті відчутні;
- найбільшою проблемою у використанні динамічного компонування є проблема зворотної сумісності динамічних бібліотек. Ця проблема виникає в ситуації, коли застосування встановлює нову версію DLL поверх попередньої. Якщо нова версія не має зворотної сумісності із попередніми, застосування, розраховані на використання попередніх версій бібліотеки, можуть припинити роботу;
- ускладнюється процес інсталювання програмного продукту, в процесі якого необхідно досліджувати які з DLL вже інстальовано в ОС і які їх версії. Інстальована нова версія DLL з некоректною зворотною сумісністю, може негативно вплинути на виконання інших програм, що її використовують і навіть не підозрюють про підміну бібліотеки.

Використання DLL у Windows

Загальні бібліотеки функцій в ОС Windows реалізовані компанією Microsoft за DLL технологією. Як правило, ці бібліотеки мають розширення файлу *.dll, *.ocx (для бібліотек, що містять елементи керування ActiveX) або *.drv (драйвери старих версій ОС). Структура DLL така сама, як і в PE-файлів (Portable Executable) для 32-, 64-розрядних Windows, та New-Executable (NE) для 16-бітових Windows.

DLL може містити 2 типи функцій: експортні та внутрішні. Експортні функції можуть бути викликані зі зовнішніх прикладних та визначаються за допомогою ключового слова `__declspec(dllexport)`. Внутрішні – це функції, які використовуються в середині DLL і не можуть бути викликані ззовні.

DLL є модулем (module). Тобто, вона складається з: сегментів коду, сегментів ресурсів та одного сегменту даних. Крім цього DLL може містити точку входу. Точка входу – це функція `DllMain()`, яка викликається при завантаженні або вивантажені бібліотеки потоком або процесом. Ця функція має наступний прототип:

```
BOOL WINAPI DllMain(
    HINSTANCE hinstDLL,   // handle to DLL module
    DWORD fdwReason,      // reason for calling function
    LPVOID lpvReserved )  // reserved
```

Якщо `DllMain()` повертає `FALSE`, то бібліотека вважається такою, що не завантажилася. При неявному зв'язуванні це призведе до відмови запуску програми, а при явному – помилки завантаження лише цієї бібліотеки.

У процесі виконання вміст бібліотеки залишається незмінним (сегменти коду та сегменти ресурсів), що дозволяє завантажувати її в пам'ять в єдиному примірнику і використовувати багатьма завданнями одночасно. Використання DLL дозволяє економити пам'ять, забезпечити модульність програм, полегшити процес встановлення програм.

Можливі 2 способи використання динамічних бібліотек. Вони називаються "явним" та "неявним" зв'язуванням. "Явне" та "неявне" зв'язування бібліотеки з програмою мають суттєві відмінності в процесі написання та компіляції програми.

Неявне зв'язування бібліотеки з програмою (Load-time dynamic linking) полягає в тому, що бібліотека (яка міститься у файлі з розширенням .dll) завантажується в пам'ять в момент завантаження програми. При відсутності бодай однієї з бібліотек при запуску програми відбудеться збій та припинення виконання програми.

Щоб реалізувати неявне зв'язування необхідно до проекту програми включити прототипи функцій, що містяться в бібліотеці та бібліотеку імпорту (має розширення .lib). На даному етапі наявність файлу з розширенням .dll не є необхідною. При компоновці створюється виконавчий файл, який містить код, що забезпечує систему інформацією, яка необхідна для автоматичного

завантаження бібліотеки з .dll файлу та інформацією, яка необхідна для зв'язування імен функцій у програмі з їх адресами у бібліотеці.

Неявне зв'язування дозволяє здійснювати виклик функцій з бібліотеки написанням коду програми в стилі притаманному мовам C\C++.

Явне зв'язування бібліотеки з програмою (Run-time dynamic linking) полягає в тому, що бібліотека (яка міститься у файлі з розширенням .dll) завантажується в пам'ять в момент часу, що визначається розробником, за допомогою виклику API функцій `LoadLibrary()` або `LoadLibraryEX()`. При успішному виконанні функція повертає адресу точки входу. При відсутності бібліотеки, яку необхідно завантажити, або при помилках її завантаження функція поверне `NULL`, а сама програма, може продовжити виконання. Звичайно, якщо функції, що містяться у відсутній бібліотеці не є критичними для її подальшої роботи.

Для виклику бібліотечної функції необхідно оголосити вказівник на функцію, та присвоїти йому адресу бібліотечної функції. Для цього необхідно використати API функцію `GetProcAddress()`, яка повертає адресу вказаної їй у параметрі бібліотечної функції.

Після завершення роботи з функціями бібліотек, програмі необхідно вивантажити бібліотеки за допомогою функції `FreeLibrary()`.

Для успішної компіляції необхідно мати лише dll файл бібліотеки. Запуск програми відбудеться навіть за відсутності бібліотечного файлу, оскільки його наявність при використанні "явного" зв'язування не перевіряється.

До переваг "неявного" зв'язування в порівнянні з "явним" відноситься:

- простота програмування. Розробник не вникає в проблеми зв'язування назв функцій з адресами за якими завантажена їх реалізація;
- прогнозованість поведінки застосування. Якщо застосування успішно завантажено, то усі проблеми перехрестних зв'язків уже вирішено;

Однак у "неявного" зв'язування існує ряд недоліків:

- при відсутності бодай однієї з бібліотек при запуску програми відбудеться збій та припинення виконання програми;
- значні затрати часу на завантаження та старт застосування, пов'язані з необхідністю завантаження усіх динамічних бібліотек;
- відсутня можливість вивантаження непотрібних в даний час динамічних бібліотек;
- на час компонування необхідно мати додаткові файли з прототипами функцій та бібліотеку імпорту (.lib).

Отже, основними перевагами "явного" зв'язування, є можливість тонко керувати процесами завантаження та вивантаження динамічних бібліотек, а отже і використовуваною пам'яттю, хоча і за рахунок складності програмування.

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке DLL?
2. Що може міститися у DLL?
3. Що таке неявне зв’язування?
4. Що таке явне зв’язування?
5. Які переваги явного зв’язування у порівнянні з неявним?
6. Які переваги неявного зв’язування у порівнянні з явним?
7. Як реалізувати неявне зв’язування?
8. Як реалізувати явне зв’язування?

ЛІТЕРАТУРА

1. What is a DLL [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/troubleshoot/windows-client/setup-upgrade-and-drivers/dynamic-link-library.
2. Yosifovich P. Windows 10 System Programming, Part 2 / Pavel Yosifovich.. – 229 с.

ЗАВДАННЯ

Написати функцію, яка вирішує поставлене завдання. Функцію оформити у вигляді DLL бібліотеки. Написати дві програми, які викличуть функцію з створеної бібліотеки при явному і неявному зв‘язуваннях.

Виконання лабораторної роботи складається з 3-ох етапів:

1. На першому етапі створюється динамічна бібліотека.
2. На другому етапі – створюється програма, яка викликатиме функцію зі створеної бібліотеки за допомогою неявного зв’язування.
3. На третьому етапі – створюється програма, яка викликатиме функцію зі створеної бібліотеки за допомогою явного зв’язування.

Примітки:

- Заданий рядок тексту – рядок символів у форматі ASCII.
- Слова у рядку тексту відділяються один від одного за допомогою одного або декількох пробілів.
- Вхідні дані вводяться з клавіатури, результат виводиться на екран.
- Вхідні дані вводяться у функції main() і передаються через параметри у функцію, яка вирішує поставлене завдання.
- Функція, яка вирішує поставлене завдання, має повернути результат у функцію main() і вивід результату на екран виконується у функції main().

ВАРІАНТИ ЗАВДАНЬ

1. Підрахувати кількість слів у заданому рядку тексту.
2. Підрахувати кількість букв “A” в останньому слові заданого рядка тексту.
3. Знайти кількість слів у заданому рядку тексту, що починаються з заданої літери.
4. Знайти кількість слів у заданому рядку тексту, в яких перший і останній символи співпадають між собою.
5. Знайти довжину найдовшого слова в у заданому рядку тексту.
6. Циклічно переставити букви в словах заданого рядку тексту так, що *i*-а буква слова стала *i*+1-ою, а остання – першою (перестановка має бути у кожному слові).
7. Знайти довжину найкоротшого слова в заданому рядку тексту.
8. Вилучити з заданого рядка тексту усі слова, що починаються на символ “A”.
9. Замінити у заданому рядку тексту слова з непарними номерами на слово “Hello”.
10. Знайти кількість слів у заданому рядку тексту, що закінчуються на літеру “A”.
11. Відредагувати заданий рядок тексту, видаляючи з нього всі слова з непарними номерами.
12. Відредагувати заданий рядок тексту, перевертаючи слова з парними номерами. (Наприклад, HOW DO YOU DO -> HOW OD YOU OD.)
13. Підрахувати кількість слів у заданому рядку тексту, що починаються з приголосної.
14. Перетворити заданий рядок тексту, залишивши в словах тільки перші входження кожної букви.
15. Перетворити заданий рядок тексту, видаливши середню букву в словах непарної довжини.
16. Підрахувати, скільки разів у заданому рядку тексту зустрічається слово “Hello”.
17. У заданому рядку тексту замінити великі латинські літери на маленькі у словах з парними номерами.
18. Перетворити заданий рядок тексту шляхом викреслювання із слів всіх букв, що стоять на непарних місцях.
19. Відредагувати заданий рядок тексту, видаляючи з нього всі слова з парними номерами.
20. Для заданого рядка тексту визначити довжину слова, яке містить в собі найбільше латинських символів.
21. Визначити, скільки символів з заданого слова міститься в заданому рядку тексту.

22. Підрахувати кількість слів у заданому рядку тексту, що починаються з голосної.
23. Відредагувати заданий рядок тексту, перевертаючи слова з непарними номерами.
24. Знайти кількість слів у заданому рядку тексту, в яких перший і останній символи різні.
25. Підрахувати у заданому рядку тексту кількість слів, у яких є цифри.
26. Підрахувати кількість букв “A” в першому слові заданого рядка тексту.
27. У заданому рядку тексту замінити цифри на символ “?” у кожному слові, яке починається з літери “A”.
28. Для заданого рядка тексту визначити довжину слова, яке містить в собі найбільше цифр.
29. У заданому рядку тексту замінити маленькі латинські літери на великі у словах з непарними номерами.
30. Замінити у заданому рядку тексту слова з парними номерами на слово “Yes”.

ПРИКЛАД ВИКОНАННЯ

1. Для створення бібліотеки створюємо новий проект та в налаштуваннях вибираємо тип проекту DLL, так як показано на рис. 3.1:

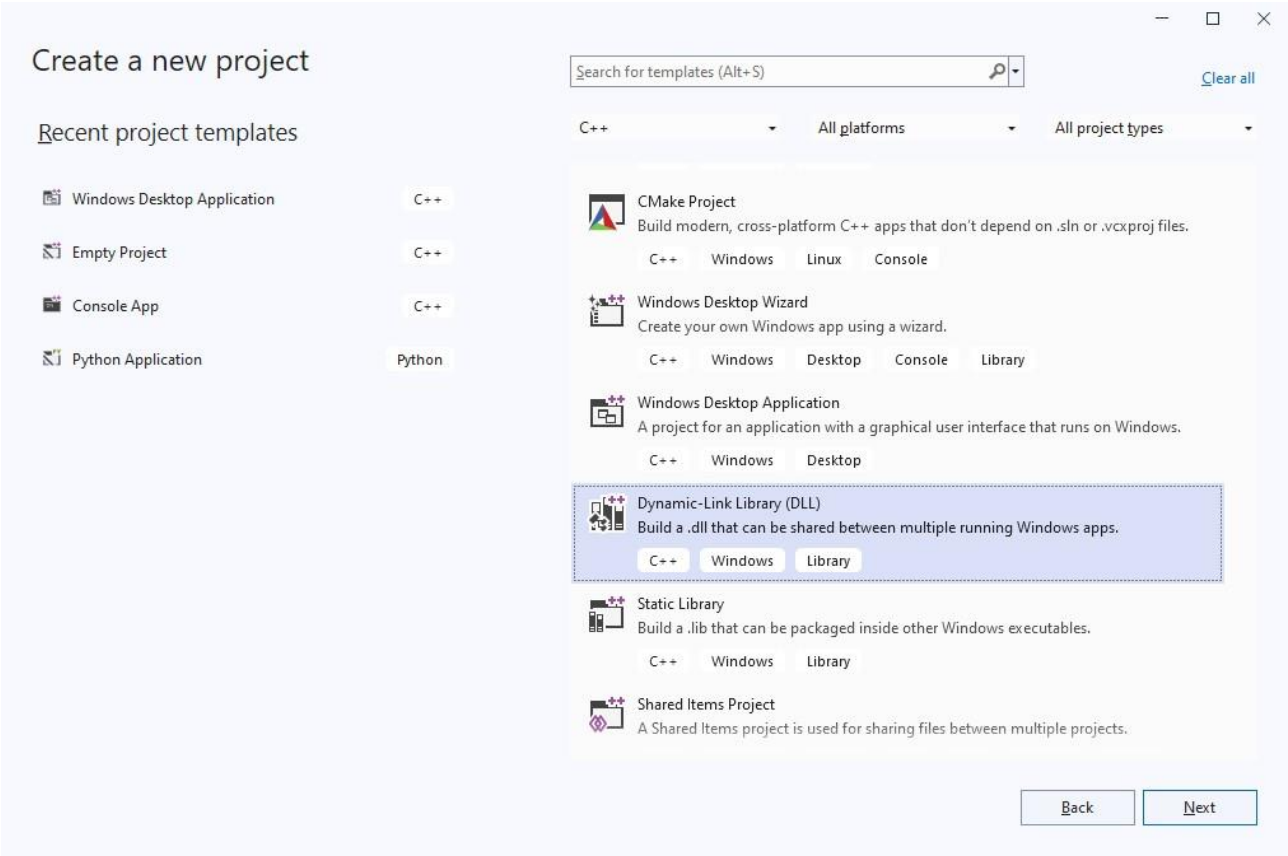

Рис.3.1. Вибір типу проекту для створення DLL засобами Microsoft Visual Studio

Створюємо файл “lab3.h” з заголовком функції, яка буде в бібліотеці. Приклад файлу наведено нижче.

```
#pragma once
extern "C" _declspec(dllexport) int stringLength(const char* str);
```

Створюємо файл з вмістом коду бібліотеки. Приклад коду наведено нижче.

```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "pch.h"
#include "lab3.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
```

```
}

extern "C" _declspec(dllexport) int stringLength(const char* str)
{
    int length = 0;
        while (str[length] != '\0')
        {
        length++;
    }
    return length;
}
```

Бібліотека містить 2 функції. Перша – точка входу в бібліотеку `DllMain()`, друга `stringLength()` рахує кількість символів у рядку тексту.

Після збірки ми одержимо DLL – бібліотеку, яка буде складатися з 2-ох файлів: lab.dll, lab.lib.

2. Далі створюємо консольний проект, у якому напишемо програму, яка викличе неявно і продемонструє роботу функції `stringLength()`. Приклад коду програми наведено нижче.

```
#include "lab3.h"
#include <stdio.h>
#include <string.h>


int main()
{
    char myString[128];
    printf("Enter string:");
    gets_s(myString, (sizeof(myString) - 1));

    // Виклик функції для обчислення довжини рядка
    int length = stringLength(myString);

    printf("Lenght of the string \"%s\" is %d symbols", myString,
length);


    return 0;
}
```

Щоб проект міг використовувати функцію з DLL необхідно зробити такі дії:

- Вибрати *Properties* для консольного проекту в *“Solution Explorer”*. Перейти до *“C/C++→General→Additional Include Directories”* і додати шлях до заголовного файлу `lab3.h`.

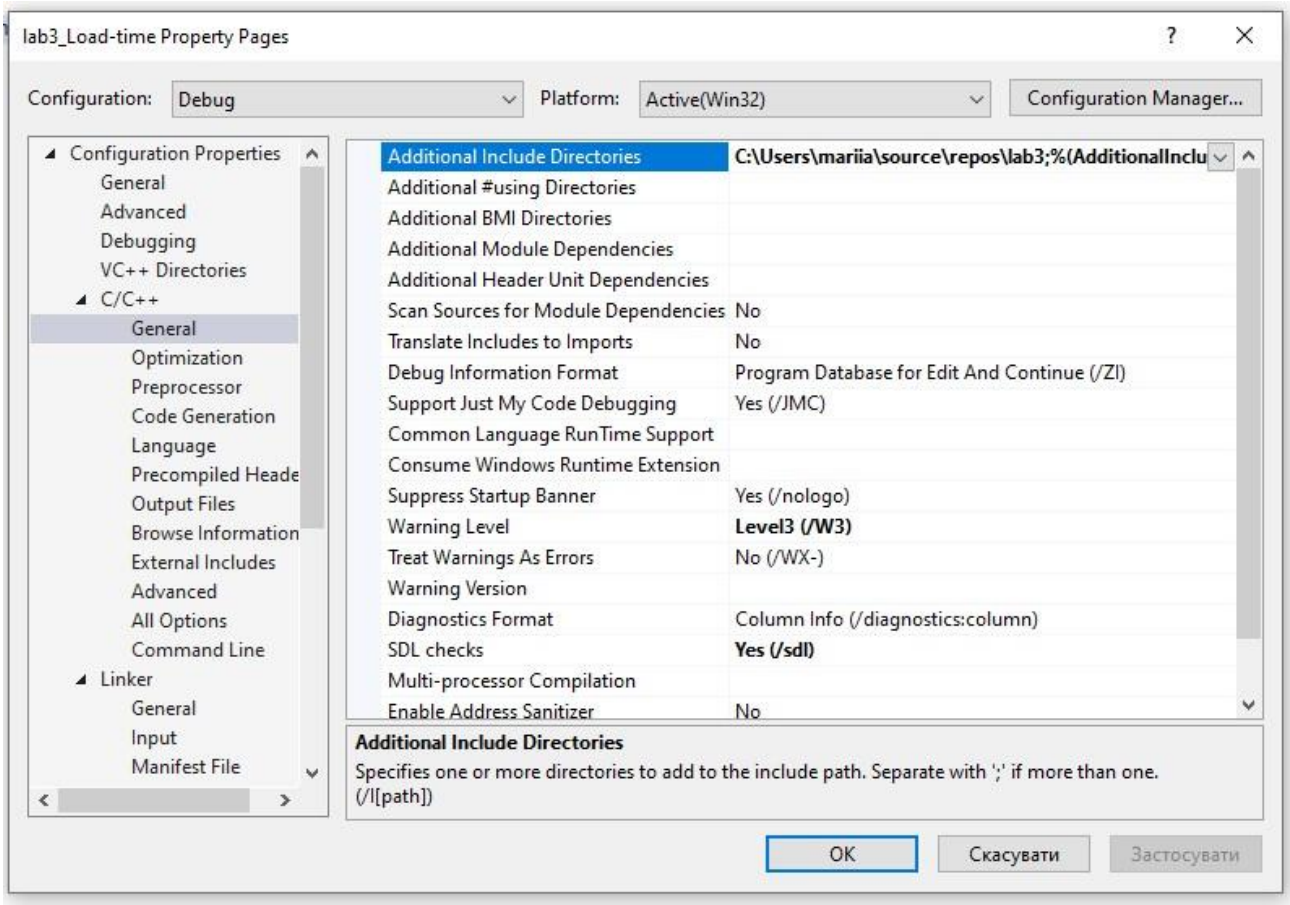

Рис.3.2. Додавання шляху до заголовного файлу lab3.h

- У розділі *“Linker→General→Additional Library Directories”* необхідно додати шлях до бібліотеки .lib, яка була створена для DLL (імпортна бібліотека).

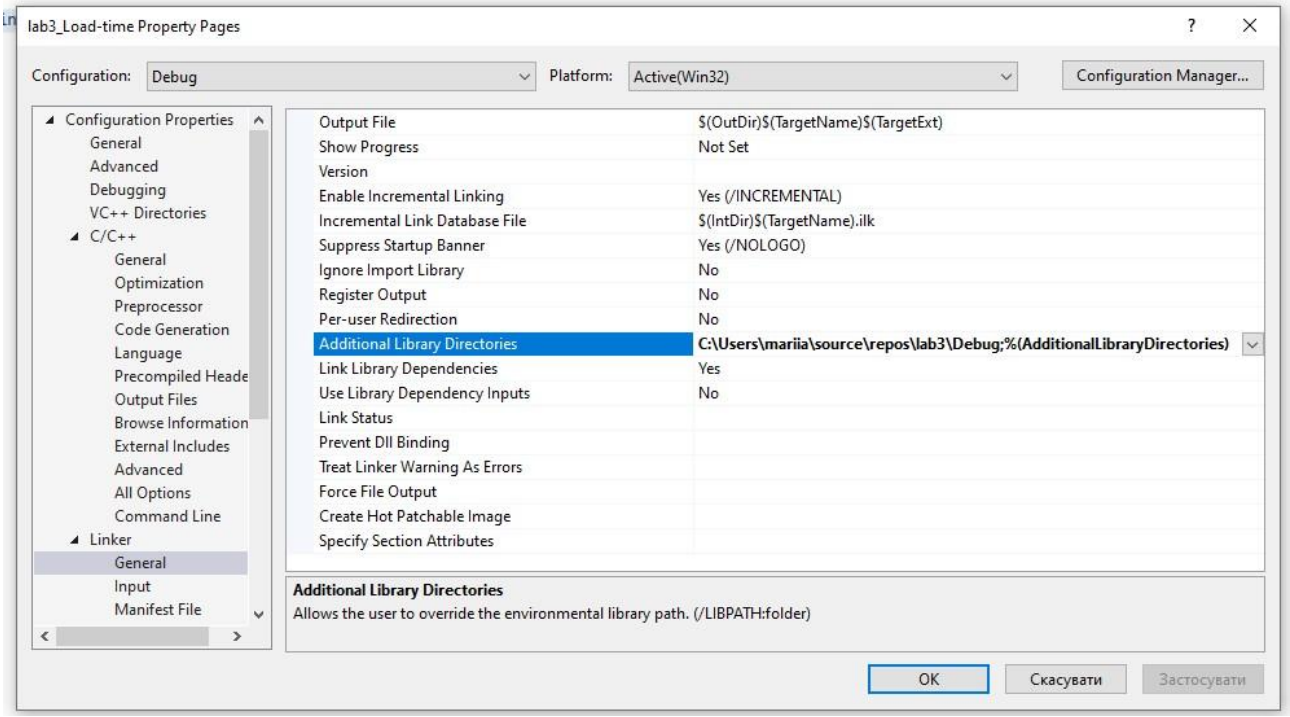

Рис.3.3. Додавання шляху до файлу бібліотеки lab3.lib

- У розділі *“Linker→Input→Additional Dependencies”* треба додати ім'я бібліотеки .lib.

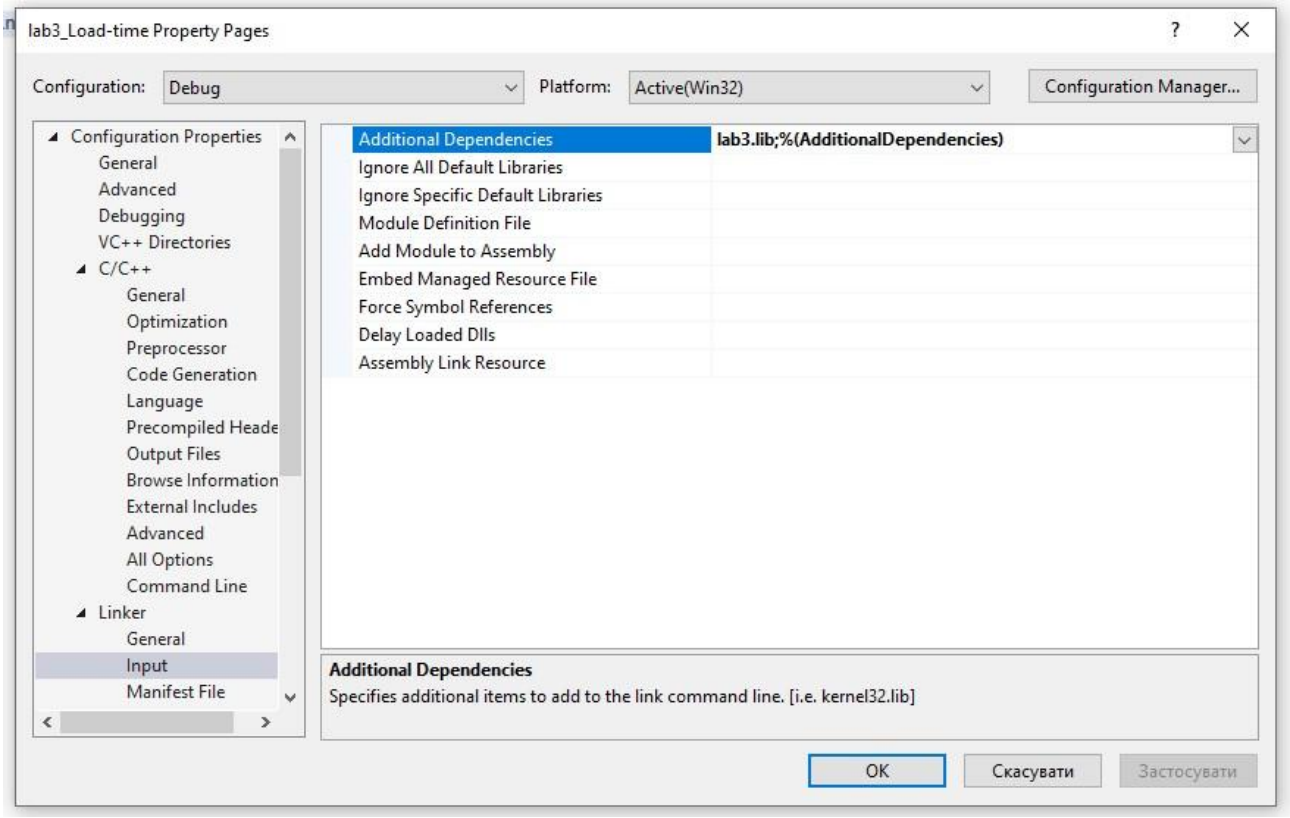

Рис.3.4. Додавання бібліотеки lab3.lib

Після цього програму можна запустити на виконання.

3. Далі створюємо ще один консольний проект, у якому напишемо програму, яка викличе явно і продемонструє роботу функції `stringLength()`. Приклад коду програми наведено нижче.

```
#include <stdio.h>
#include <string.h>
#include <windows.h>

// Тип функції, яка буде викликатися з DLL
typedef int (*StringLengthFunc)(const char*);

int main() {
    // Шлях до DLL
    HMODULE hLib = LoadLibrary(TEXT("lab3.dll"));

    if (hLib == NULL)
    {
        printf("Failed to load DLL!");
        return 1;
    }

    // Отримуємо адресу функції stringLength з DLL
```

```
    StringLengthFunc stringLength =
(StringLengthFunc)GetProcAddress(hLib, "stringLength");

    if (stringLength == NULL) {
        printf("Could not find stringLength function in DLL!");
        FreeLibrary(hLib);
        return 1;
    }


    char myString[128];
    printf("Enter string:");
    gets_s(myString, (sizeof(myString) - 1));

    // Виклик функції для обчислення довжини рядка
    int length = stringLength(myString);

    printf("Lenght of the string \"%s\" is %d symbols", myString,
length);

    // Вивантажуємо DLL після завершення роботи
    FreeLibrary(hLib);

    return 0;
}
```