**LECTURE LAB WORK**

**System Programming**

**CREATING A SHARE OBJECT OR DLL FILE**

**Step No.1**

**First Create a source file of aa function that will take two numbers as paremater and will add them.**

**Source Code:**

**// addingNumber.cpp**

**#include <iostream>**

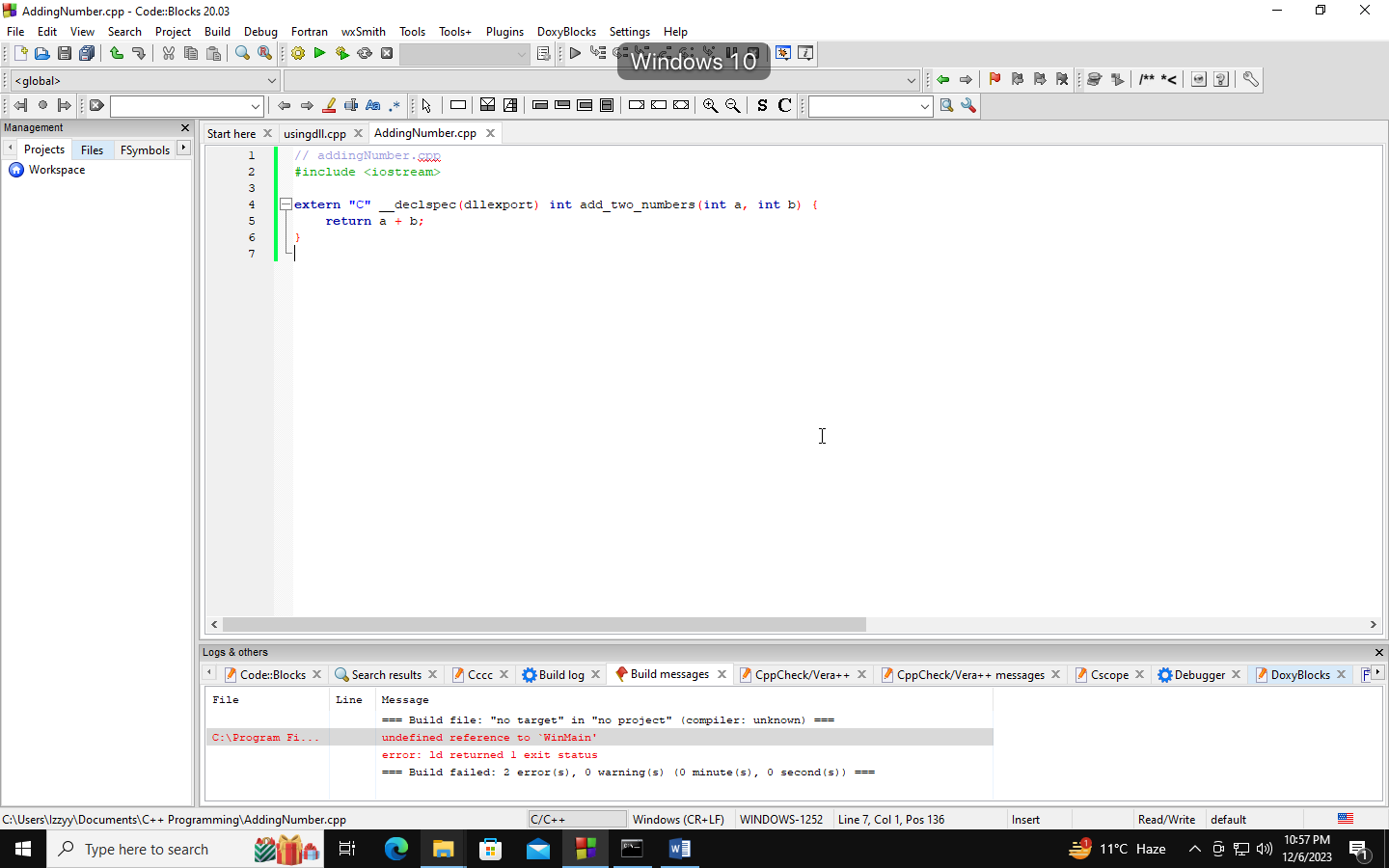
**extern "C" \_\_declspec(dllexport) int add\_two\_numbers(int a, int b) {**

**return a + b;**

**}**

**When this file will be compiled it will generate .o object file in the same directory .**

**The following figure shows the compiled code in CodeBlock.**

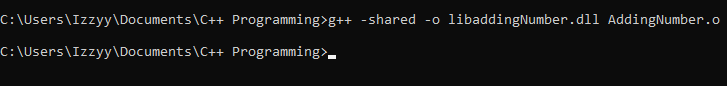
****

**Step No.2**

**We will create the .dll or.so file from the .o object by using the following command in the Command prompt.**

**g++ -shared –o libbaddingNumber.dll AddingNumber.o**

**The following figure shows the command in the command prompt.**

****

**A DLL file with the name libaddingNumber.DLL will be created in the same folder.**

**Step No.3**

**Now we will write a c++ source code that will use this dll file for adding two numbers.**

**Source Code :**

**#include <iostream>**

**#include <windows.h>**

**typedef int (\*AddFunction)(int, int);**

**int main() {**

**HMODULE dllHandle = LoadLibrary("libaddingNumber.dll");**

**if (dllHandle == nullptr) {**

**std::cerr << "Error loading DLL." << std::endl;**

**return 1;**

**}**

**AddFunction addFunction = reinterpret\_cast<AddFunction>(GetProcAddress(dllHandle, "add\_two\_numbers"));**

**if (addFunction == nullptr) {**

**std::cerr << "Error getting function from DLL." << std::endl;**

**FreeLibrary(dllHandle);**

**return 1;**

**}**

**int result = addFunction(10, 5);**

**std::cout << "Result: " << result << std::endl;**

**FreeLibrary(dllHandle);**

**return 0;**

**}**

**Explaination :**

* **#include <iostream> and #include <windows.h>:**

**These are include directives for including the necessary C++ standard library headers (iostream) and Windows API headers (windows.h) for working with dynamic-link libraries (DLLs) on Windows.**

* **typedef int (\*AddFunction)(int, int);**

**This line defines a typedef named AddFunction for a function pointer type that points to a function taking two int parameters and returning an int. This is used to declare the function pointer later.**

**int main() {**

**The entry point of the program. Execution starts from here.**

* **HMODULE dllHandle = LoadLibrary("libaddition.dll");**

**LoadLibrary is a Windows API function that loads a dynamic-link library (DLL) into the address space of the calling process.**

**The program attempts to load the DLL named "libaddition.dll" and assigns the module handle to dllHandle.**

* **if (dllHandle == nullptr) {**

**Checks if the DLL loading was successful. If dllHandle is nullptr, it indicates an error in loading the DLL.**

* **std::cerr << "Error loading DLL." << std::endl;**

**Prints an error message to the standard error stream if there is an issue loading the DLL.**

**return 1;**

**Exits the program with a return code of 1, indicating an error.**

* **AddFunction addFunction = reinterpret\_cast<AddFunction>(GetProcAddress(dllHandle, "add\_two\_numbers"));**

**GetProcAddress is another Windows API function that retrieves the address of an exported function or variable from the specified dynamic-link library (DLL).**

**The program attempts to get the address of the function named "add\_two\_numbers" from the DLL and assigns it to the function pointer addFunction.**

* **if (addFunction == nullptr) {**

**Checks if obtaining the function address was successful. If addFunction is nullptr, it indicates an error.**

* **std::cerr << "Error getting function from DLL." << std::endl;**

**Prints an error message to the standard error stream if there is an issue obtaining the function address.**

* **FreeLibrary(dllHandle);**

**FreeLibrary is a Windows API function that decrements the reference count of the loaded DLL. If the reference count reaches zero, the DLL is unloaded from the address space of the calling process.**

**return 1;**

**Exits the program with a return code of 1, indicating an error.**

**int result = addFunction(3, 4);**

**Calls the function pointed to by addFunction with arguments 3 and 4 and assigns the result to the variable result.**

**std::cout << "Result: " << result << std::endl;**

**Prints the result to the standard output.**

**FreeLibrary(dllHandle);**

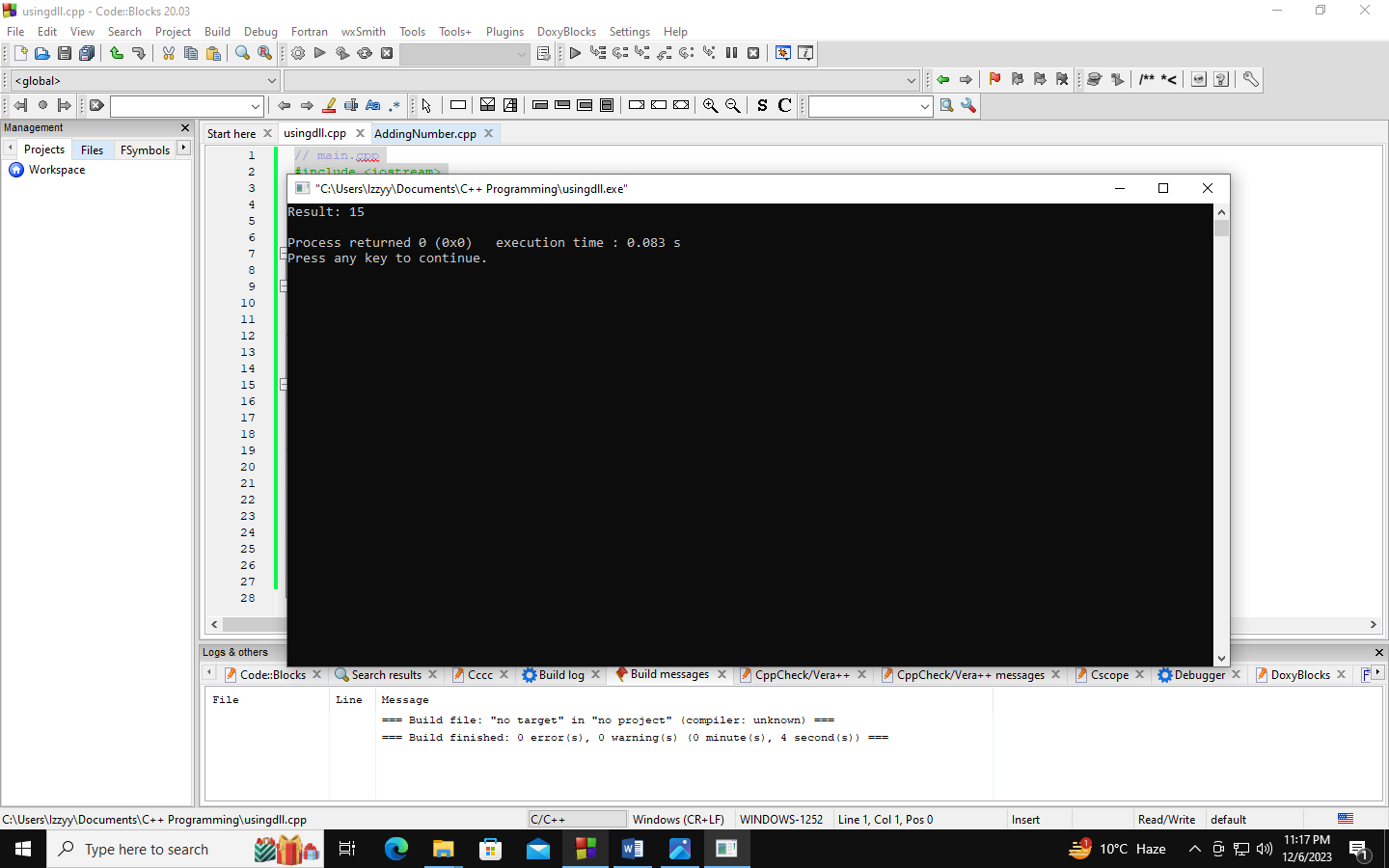
**Unloads the DLL. In this case, it is called at the end of the program, ensuring that the DLL is unloaded before the program exits.**

**return 0;**

**Exits the program with a return code of 0, indicating successful execution.**

**StepNo.4**

**After running the code and call the dll lib the it will create the following output that is shown in the below figure.**

****