Table of Contents

1. Lab 2: Conû guration of the Visual Studio Community 2019 for dll lib
MASM 2
1.1. Creating the project with dll library written in assembler 2
1.1.1. Creating of the new project 2
1.1.2. Adding the assembly part to the solution 4
1.1.3. Setting dependencies in the solution 7
1.1.4. Zamiana biblioteki z C++ na asemblerow 9
1.1.5. Ustawienie trybu kompilacji 15
1.1.6. KoDcowa konû guracja biblioteki asemblerowej 22
1.2. Wykorzystanie biblioteki dll napisanej w asemblerze 26
1.2.1. Biblioteka asemblerowa dll 26
1.2.2. OdwoBanie do biblioteki dll z programu C++ 27
1.2.3. Zadanie do zrealizowania w ramach laboratorium 28

1.Lab 2: Conû guration of the Visua Community 2019 for dll libraries wr MASM.

1.1Creating the project with dll library written assembler

In the beginning, we can create a project in which the main program w Language) like C++, and the library written in assembler will be linked



Creation of libraries written in assembly language, with application written in a chosen high-level language is cur assembler use.



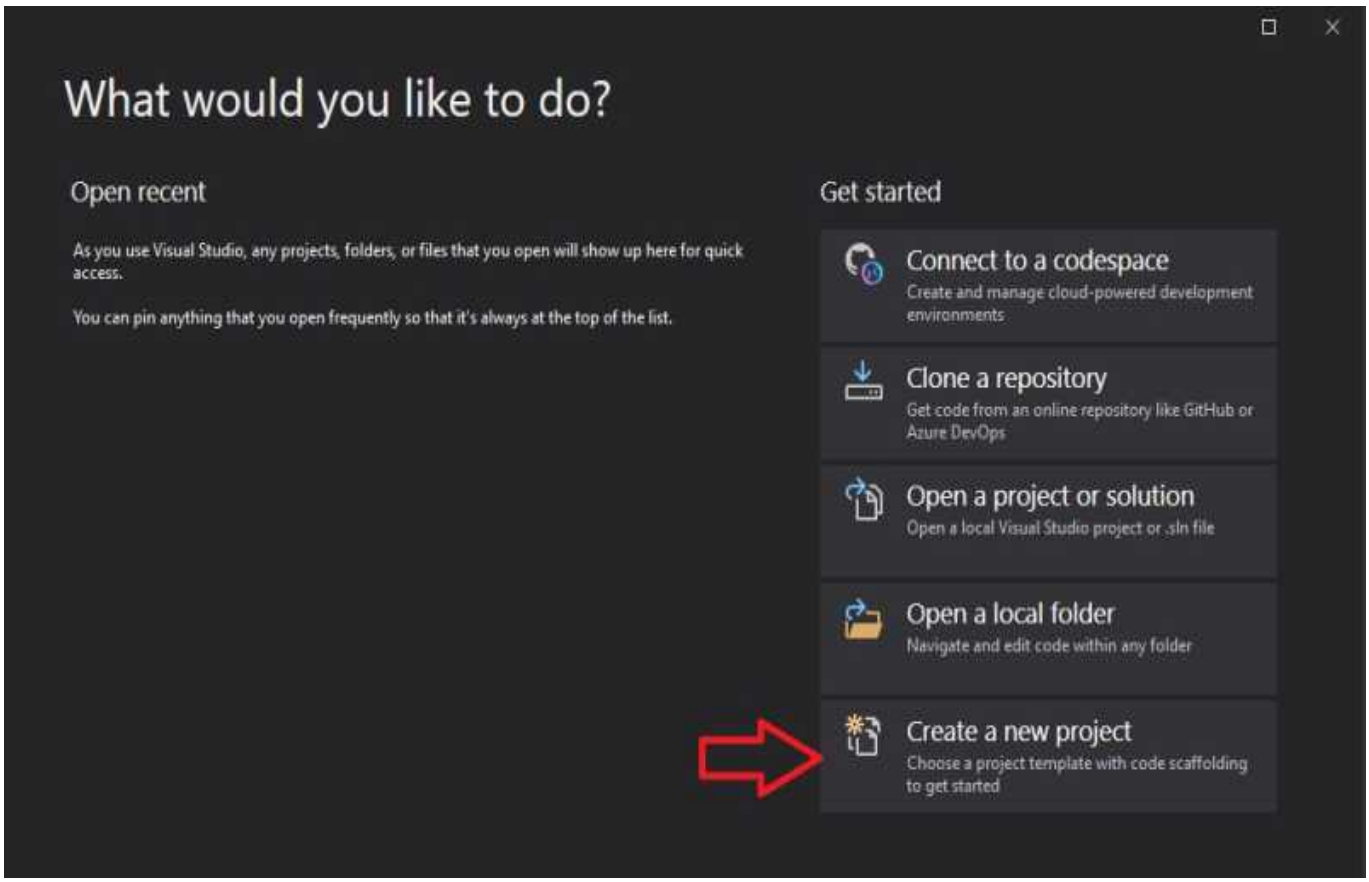
The conû guration process presented in the following part program written in C++ and C#. The screenshots used are fro for both programming languages, but the diû erences are usuall aû ect the environment conû guration process.

1.1.1Creating of the new project

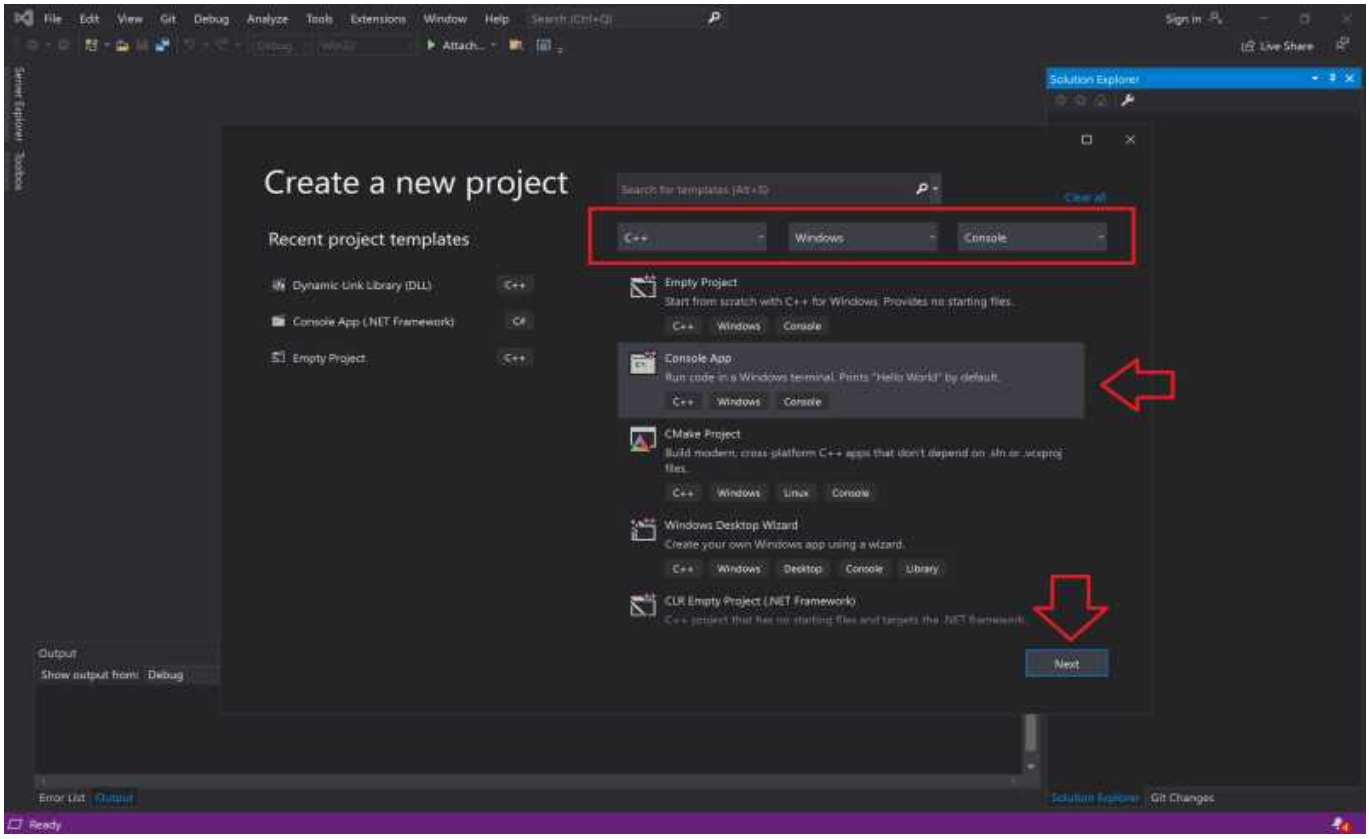
During this laboratory, we'll show how to create and use dll (dynamica assembler in that application written in C++ We start by creating a ne



We'll create the console application, written in C++, for execution in t



We need to deû ne that name for the project. We suggest the name La left with default values.

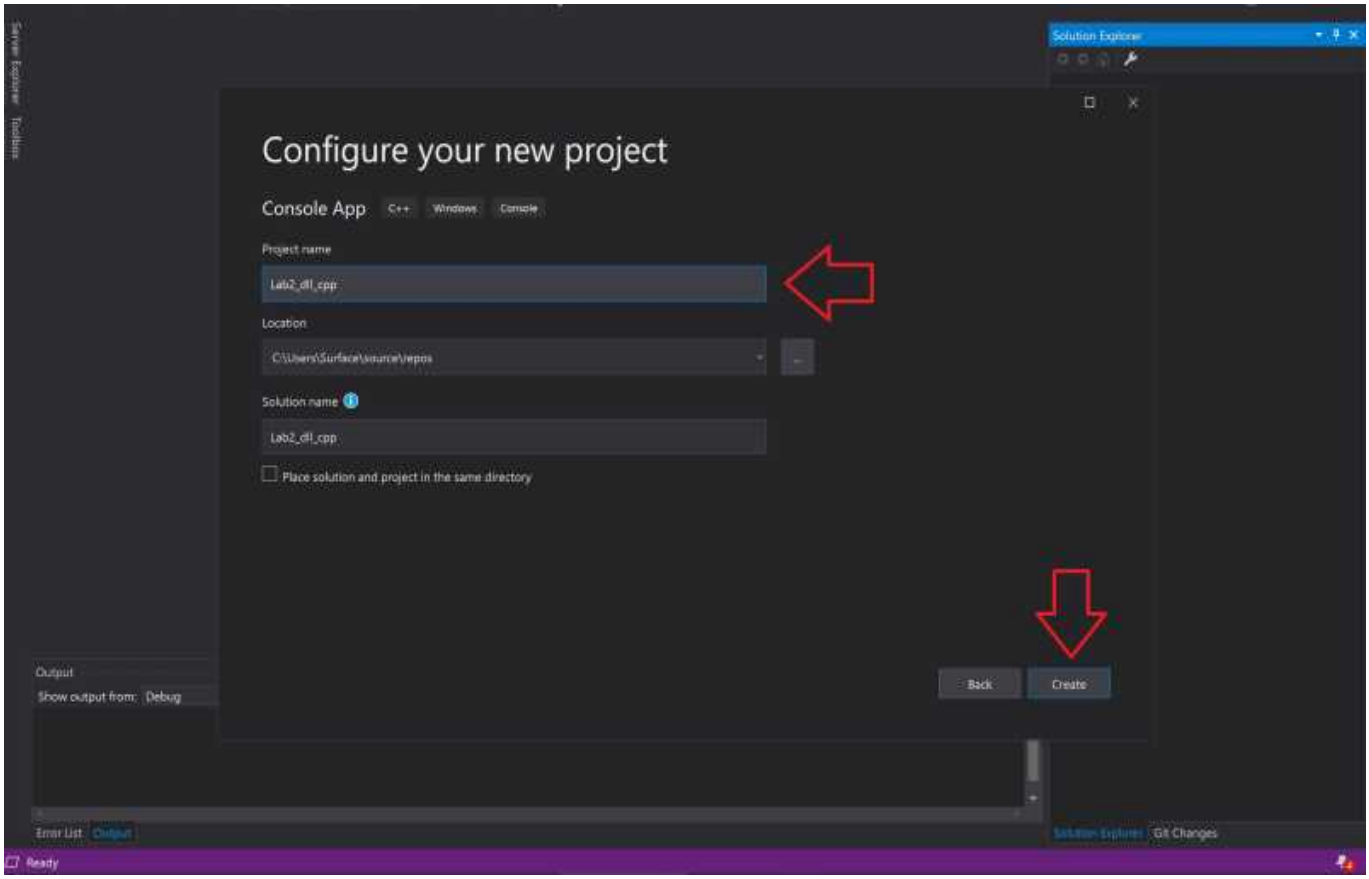


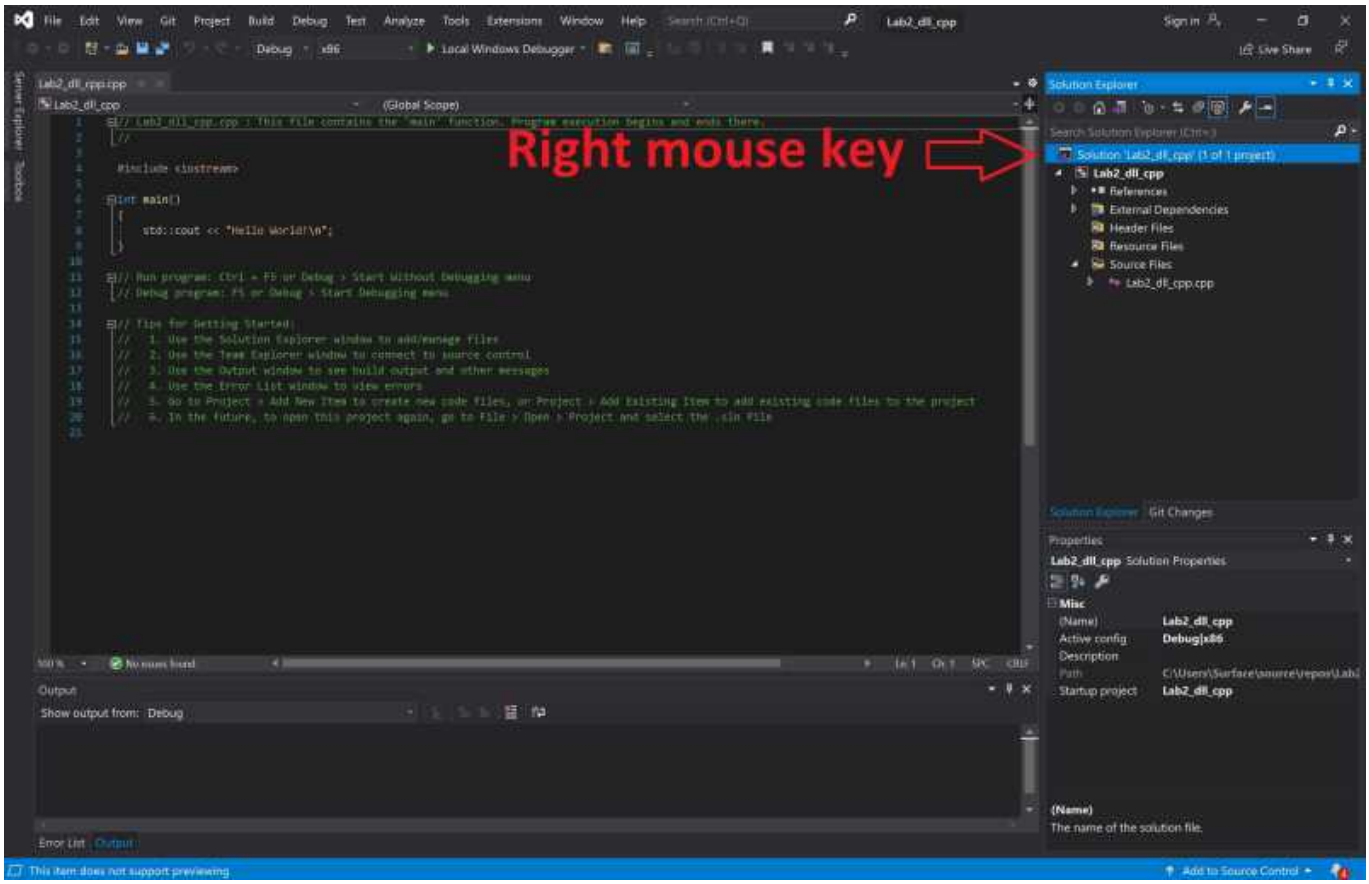


At this stage the solution will be created including the proje template.

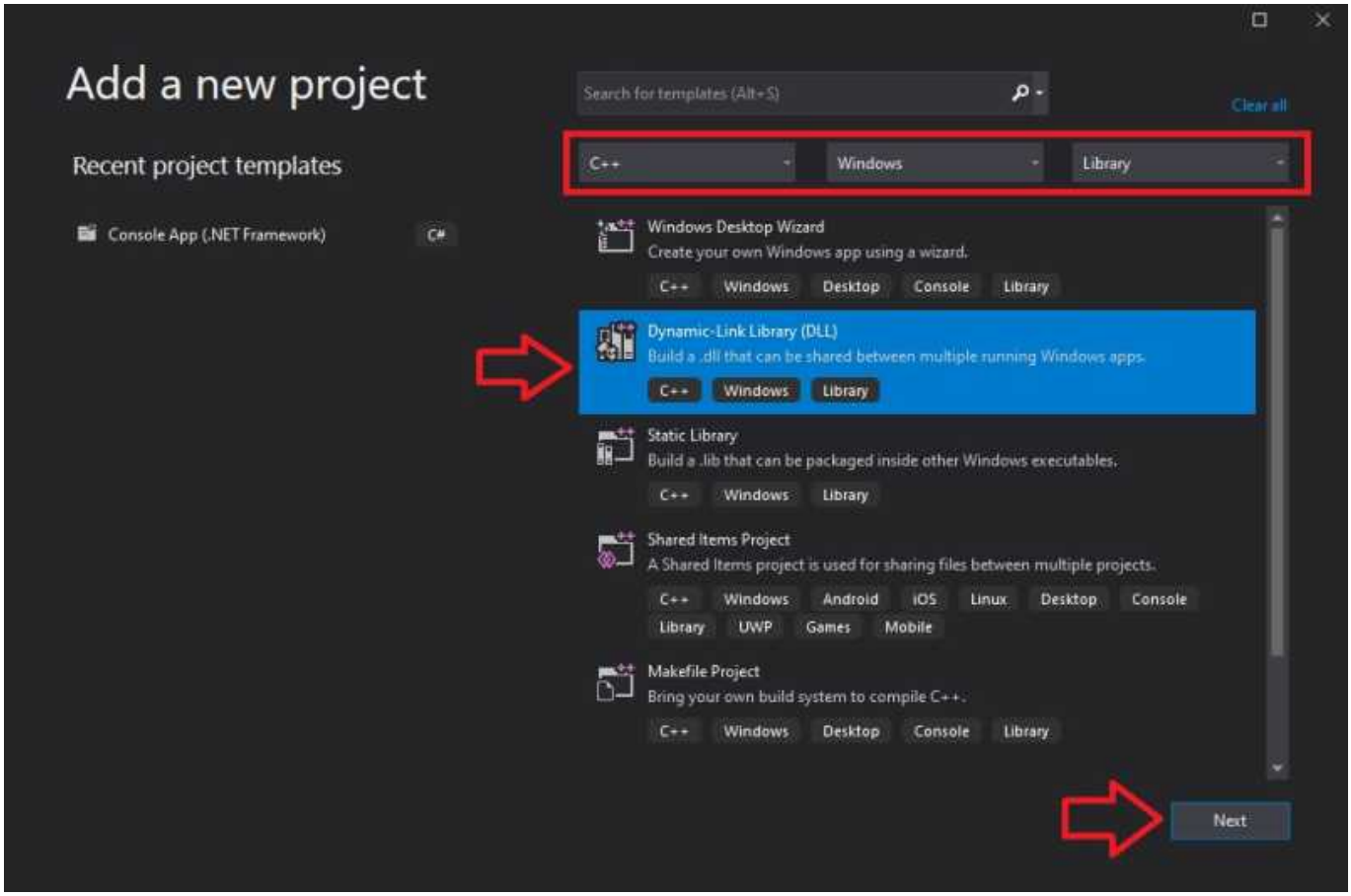
1.1.2 Adding the assembly part to the solution

To add a library written in assembler to our solution, we add a new pr on the solution and select add new project.

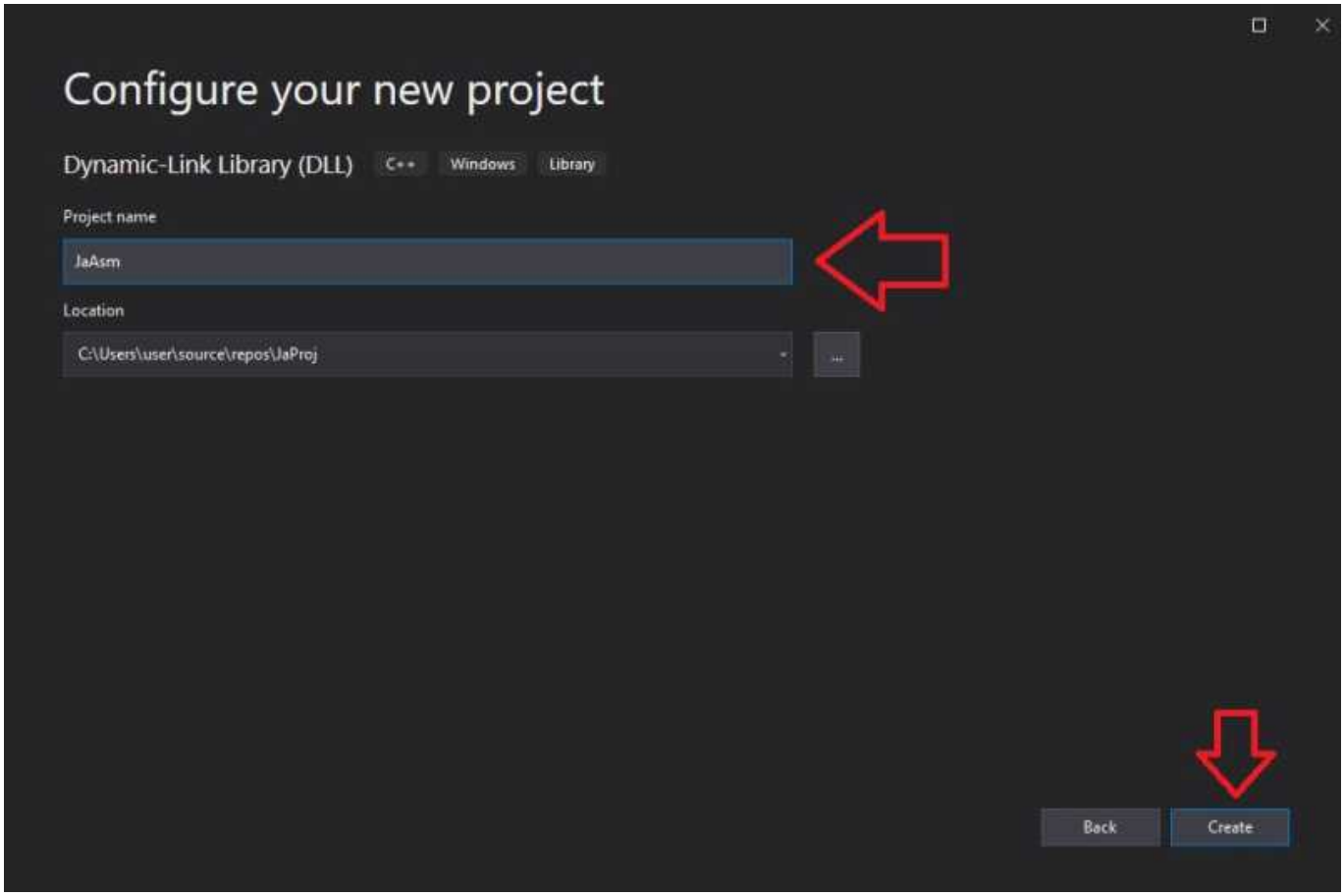




We create our new assembler project as a C++ compatible library proj
Windows operating system. From the list of available options, we sele
(DLL).



We name the project e.g. JaAsm and create it leaving the default pat

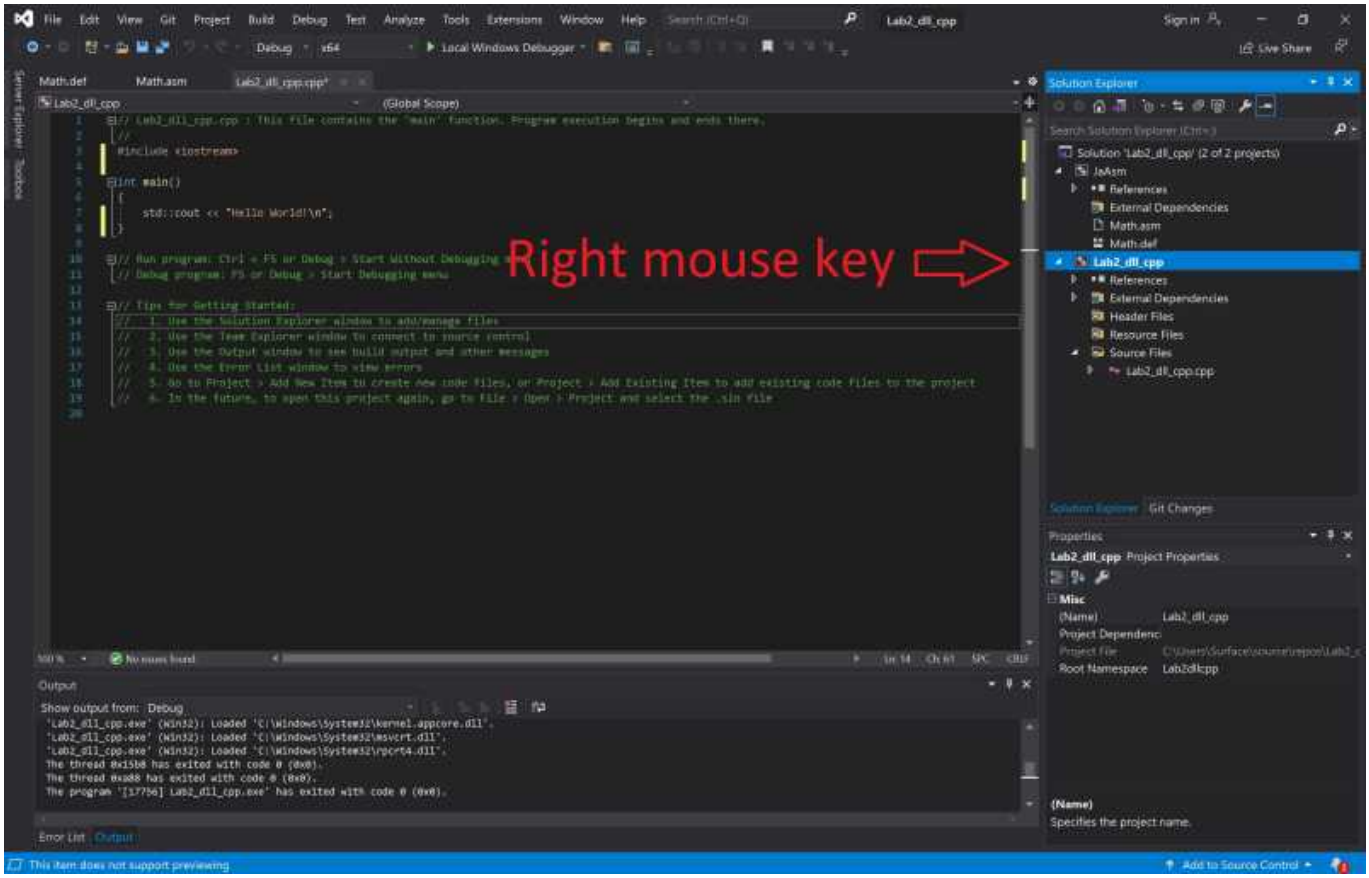


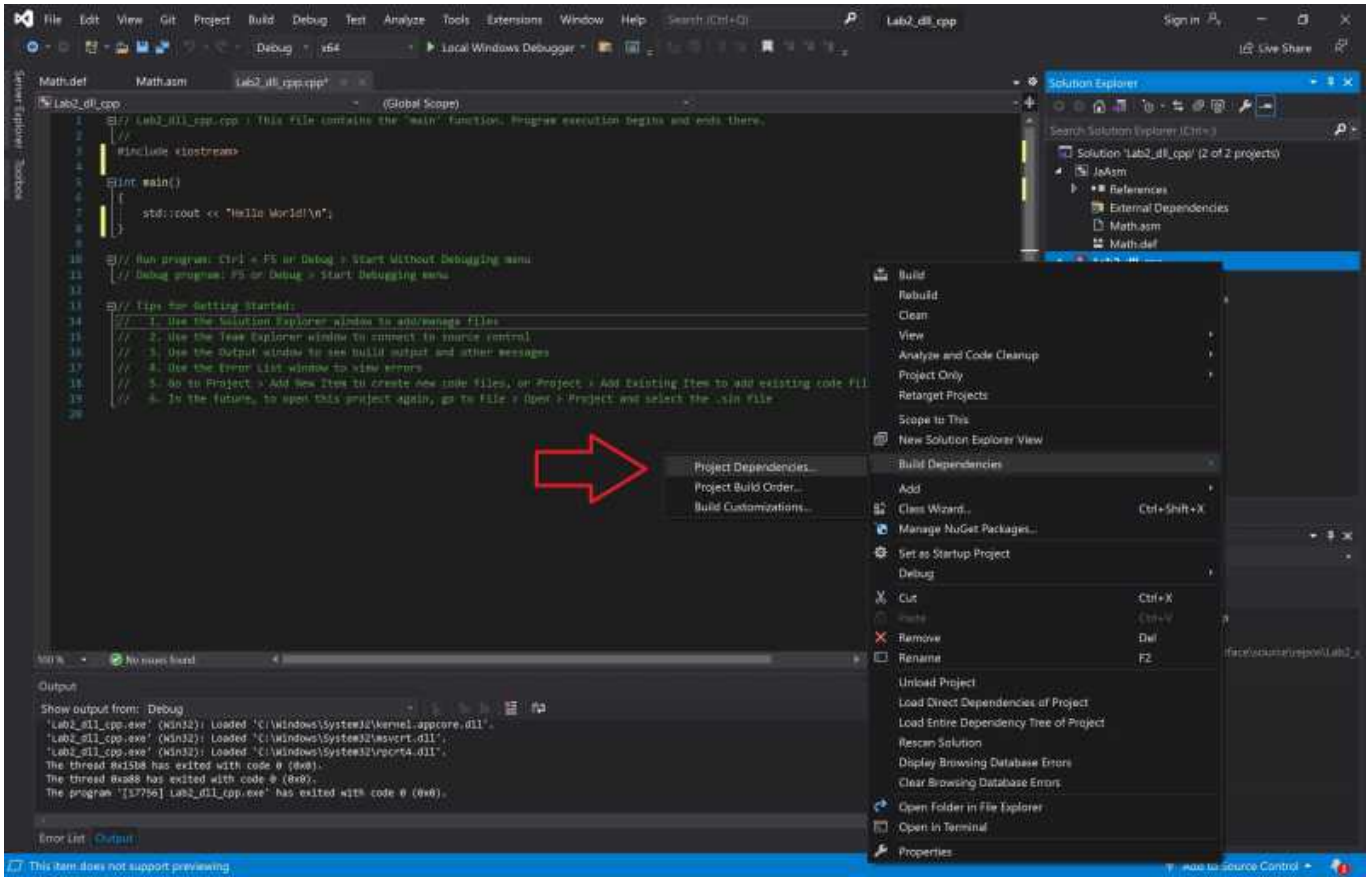


Both parts of our solution will be placed as subfolders in on to the name of the solution, in our case Lab2_dll_cpp . Eac take the name of the respective projects, i.e. Lab2_dll_cp C++ and JaAsm for the part written in assembler.

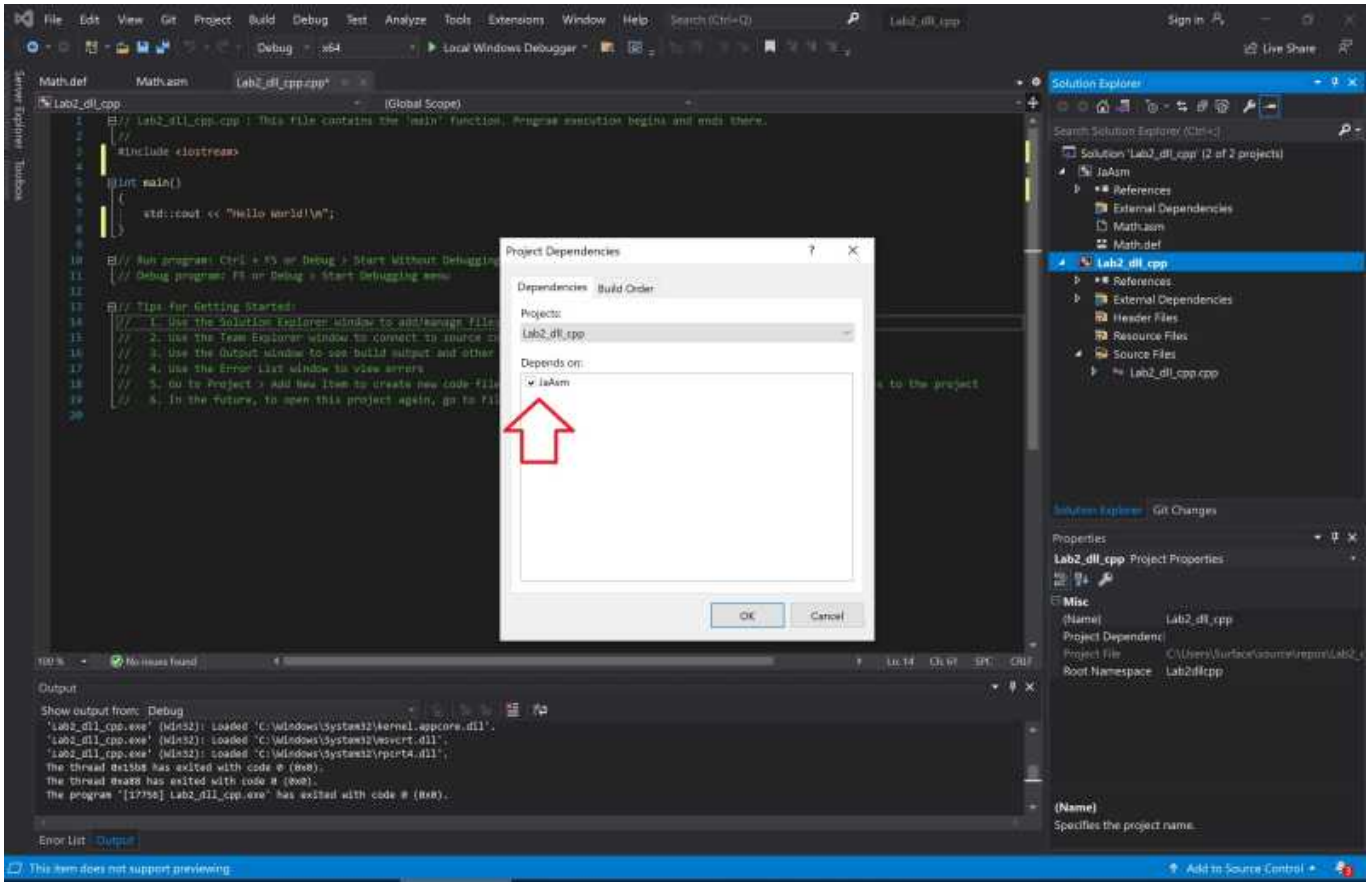
1.1.3 Setting dependencies in the solution

The next step is to set up the project dependencies so that compiling automatically forces the assembly library to be compiled. To do this, r project, select the Build dependencies menu, and then select Projec





In the newly opened window, select our JaAsm library and click OK

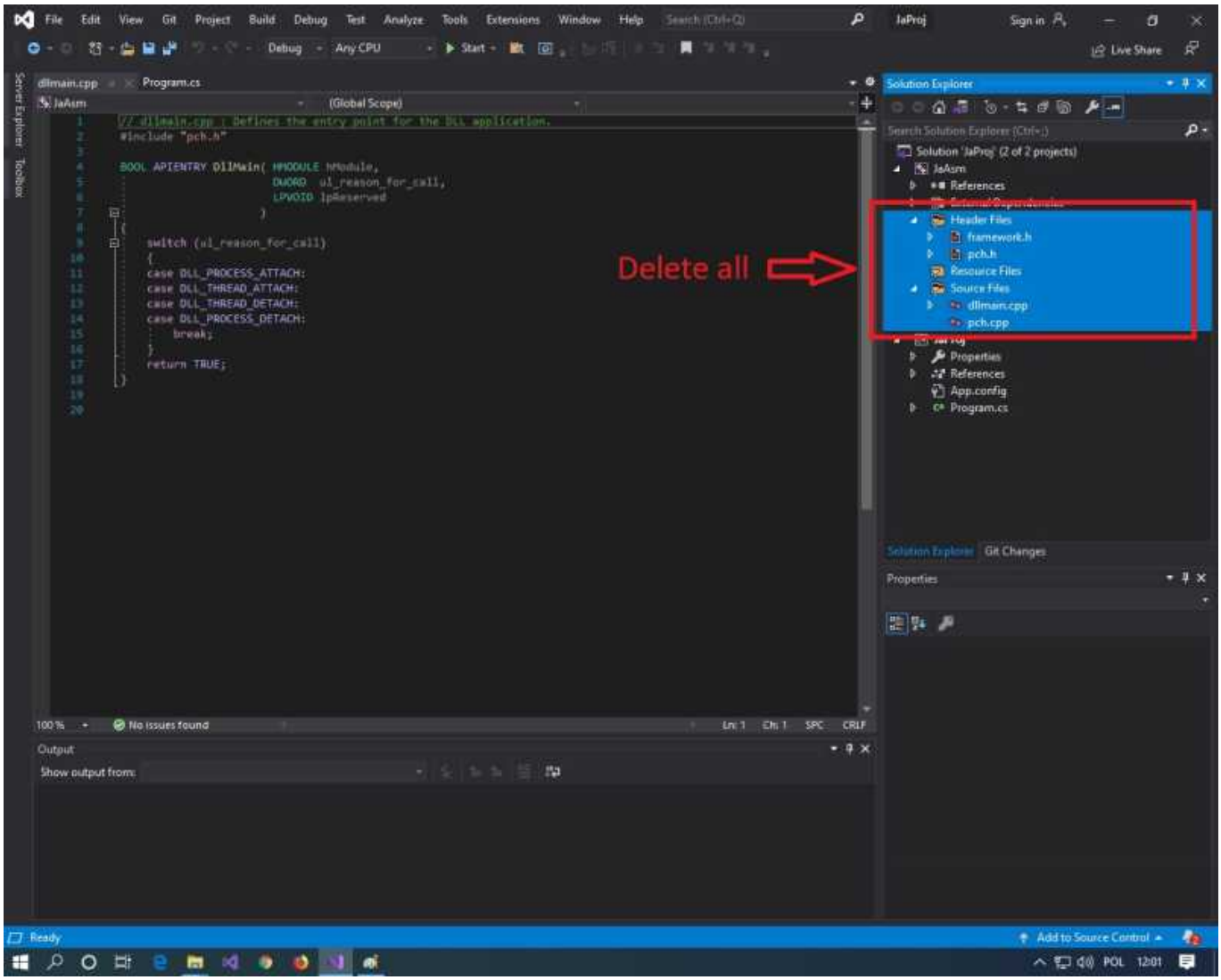


Once this dependency is set, when you compile a project or

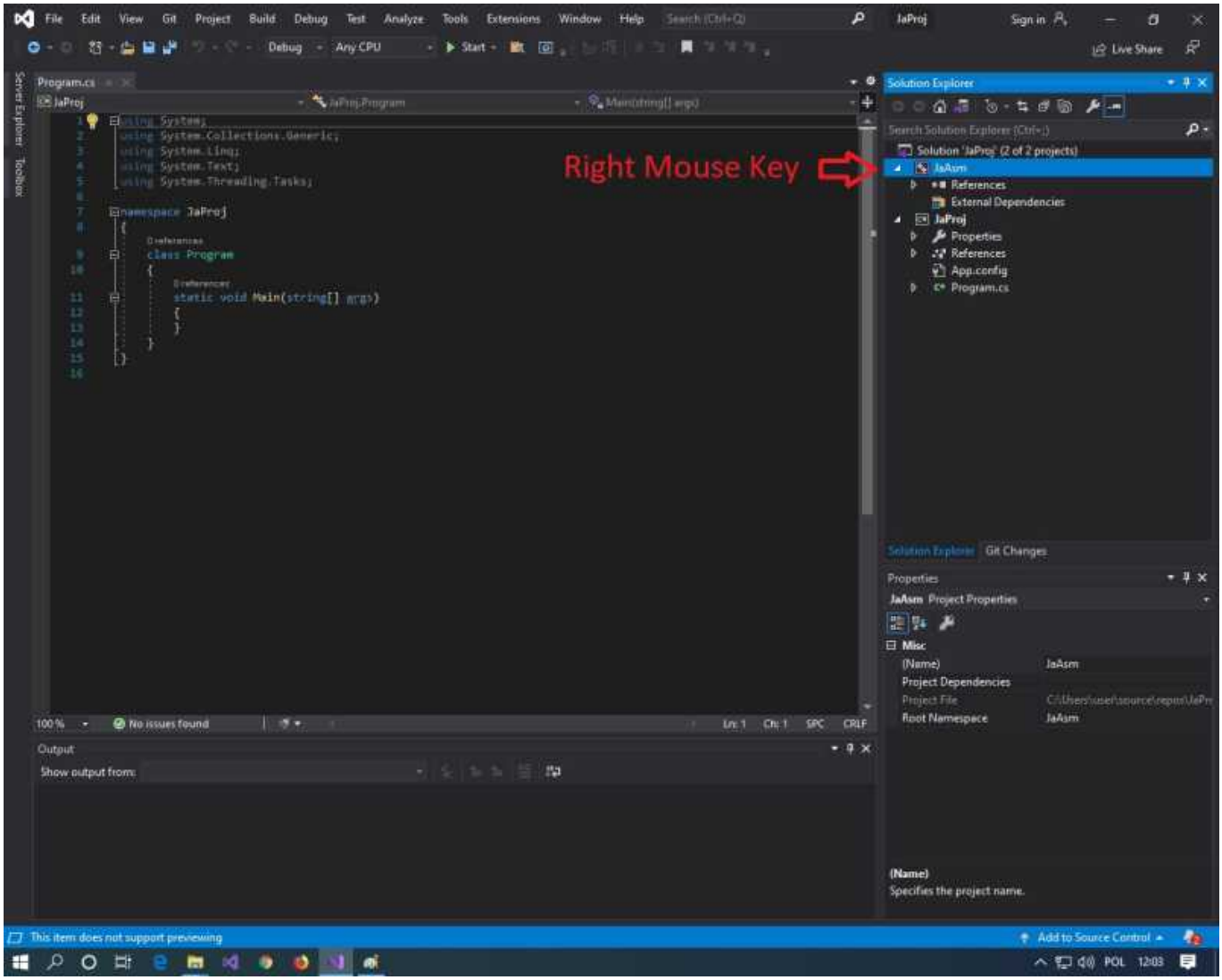
the assembler file will also be recompiled automatically.

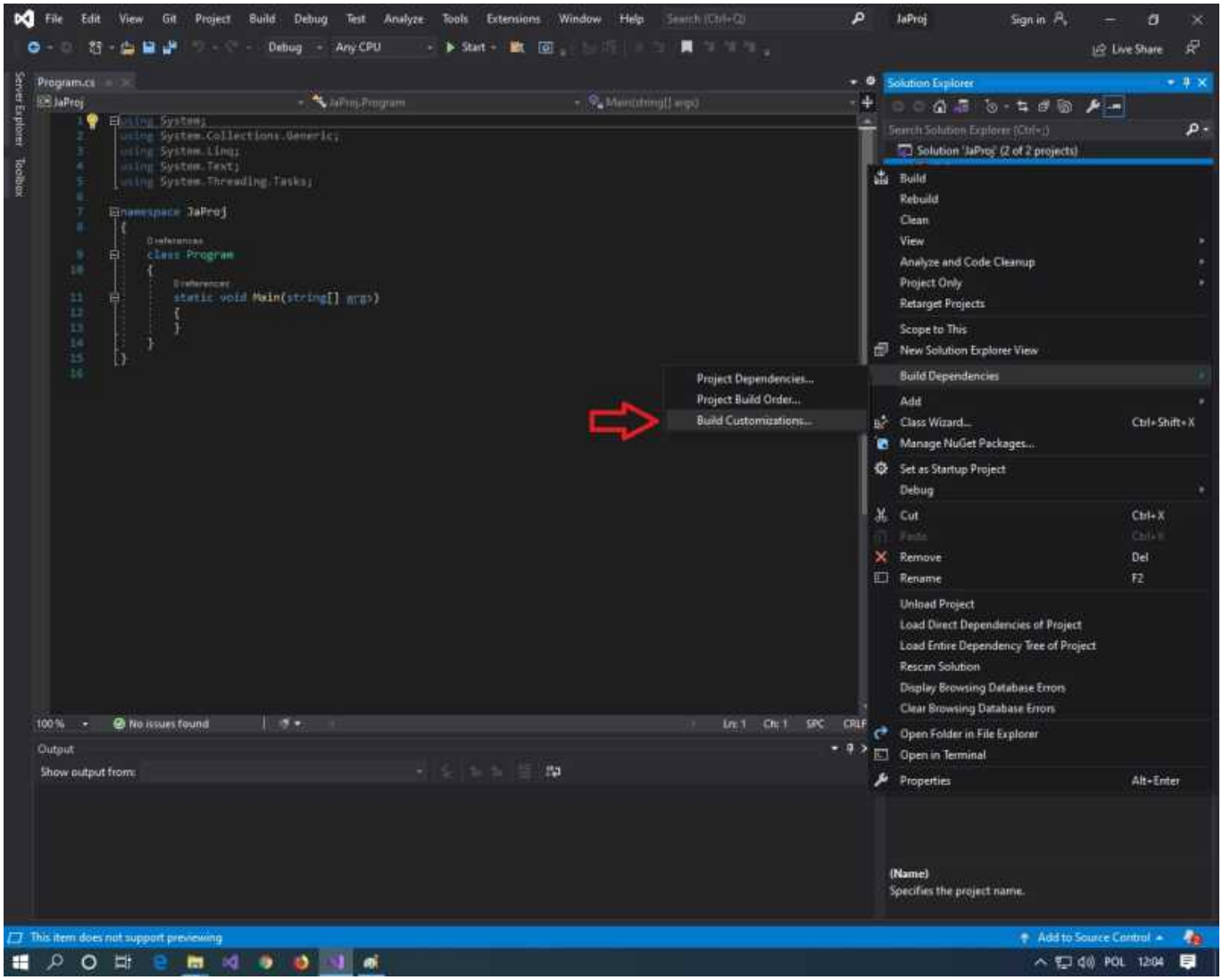
1.1.4 Zamiana biblioteki z C++ na asemblerow

Najpierw należy usunąć automatycznie utworzone pliki projektu zgodneg wystarczy zaznaczyć wskazane pliki nagłówkowe i źródłowe, a następ wybrać Usuń bądź też kliknąć Delete.

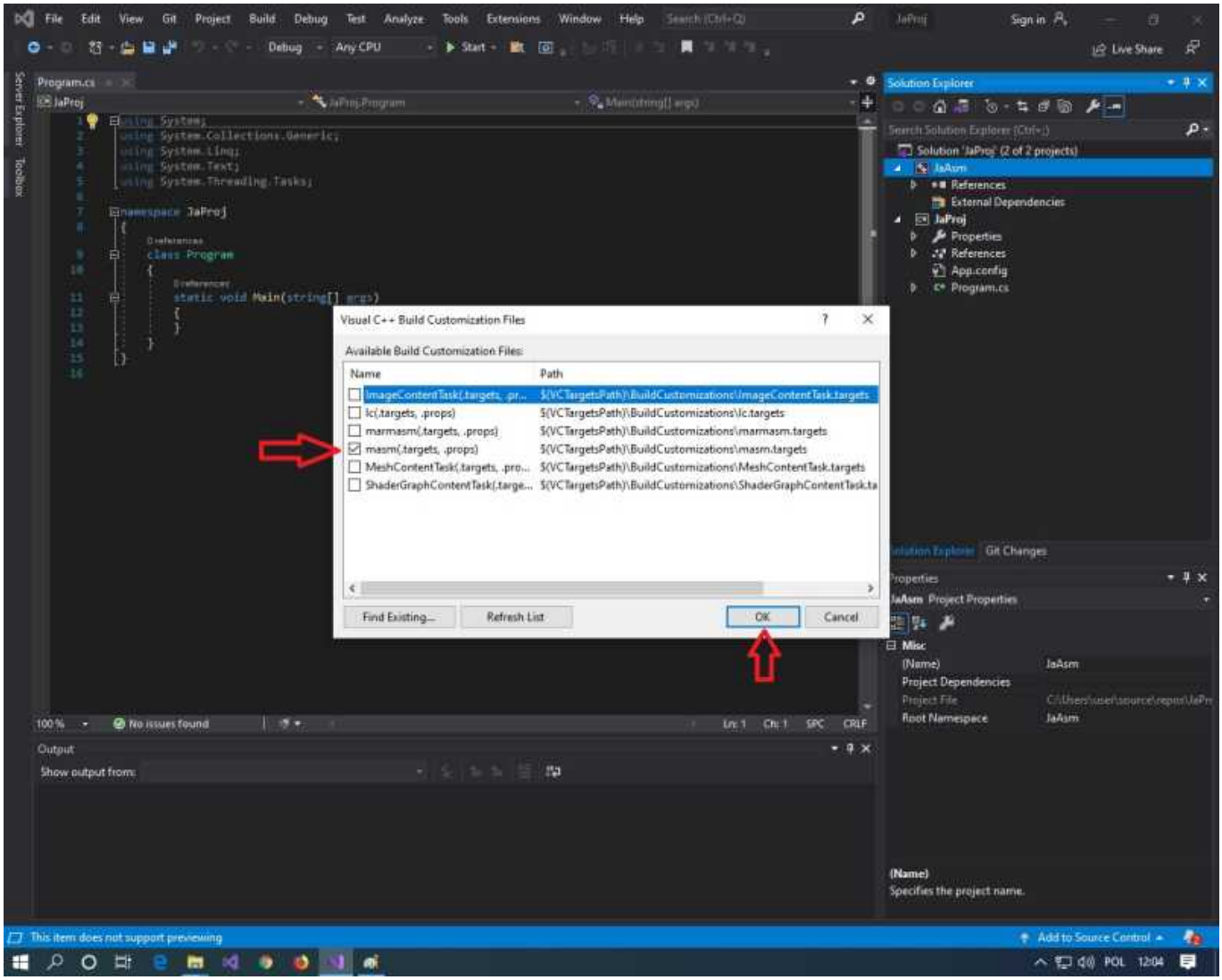


Dla biblioteki konieczne jest dostosowanie kompilacji jako plik asembl klikamy projekt biblioteki prawym klawiszem myszy i z listy wybieramy dependencies), a następnie Dostosowanie kompilacji (Build customiza

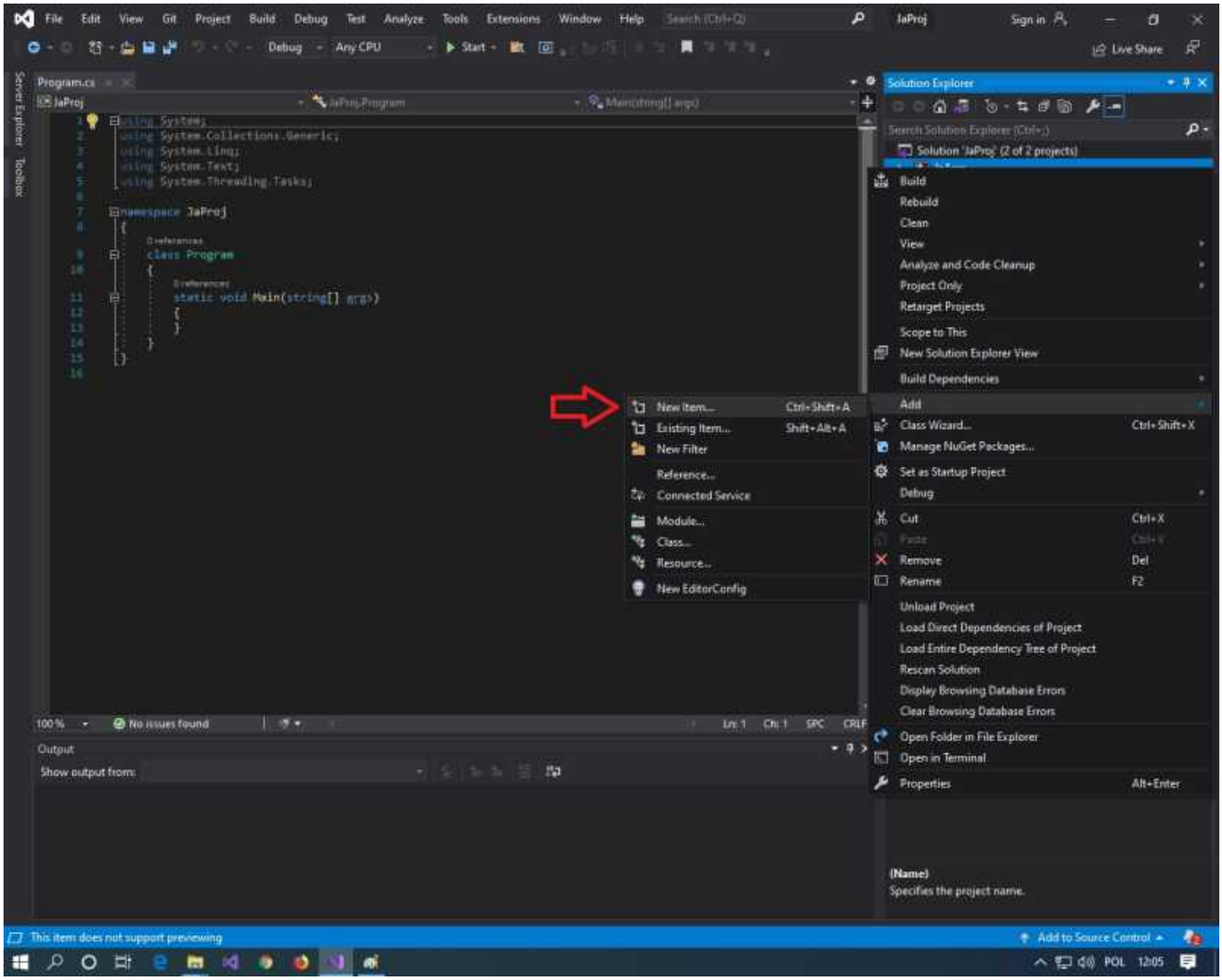

Right Mouse Key



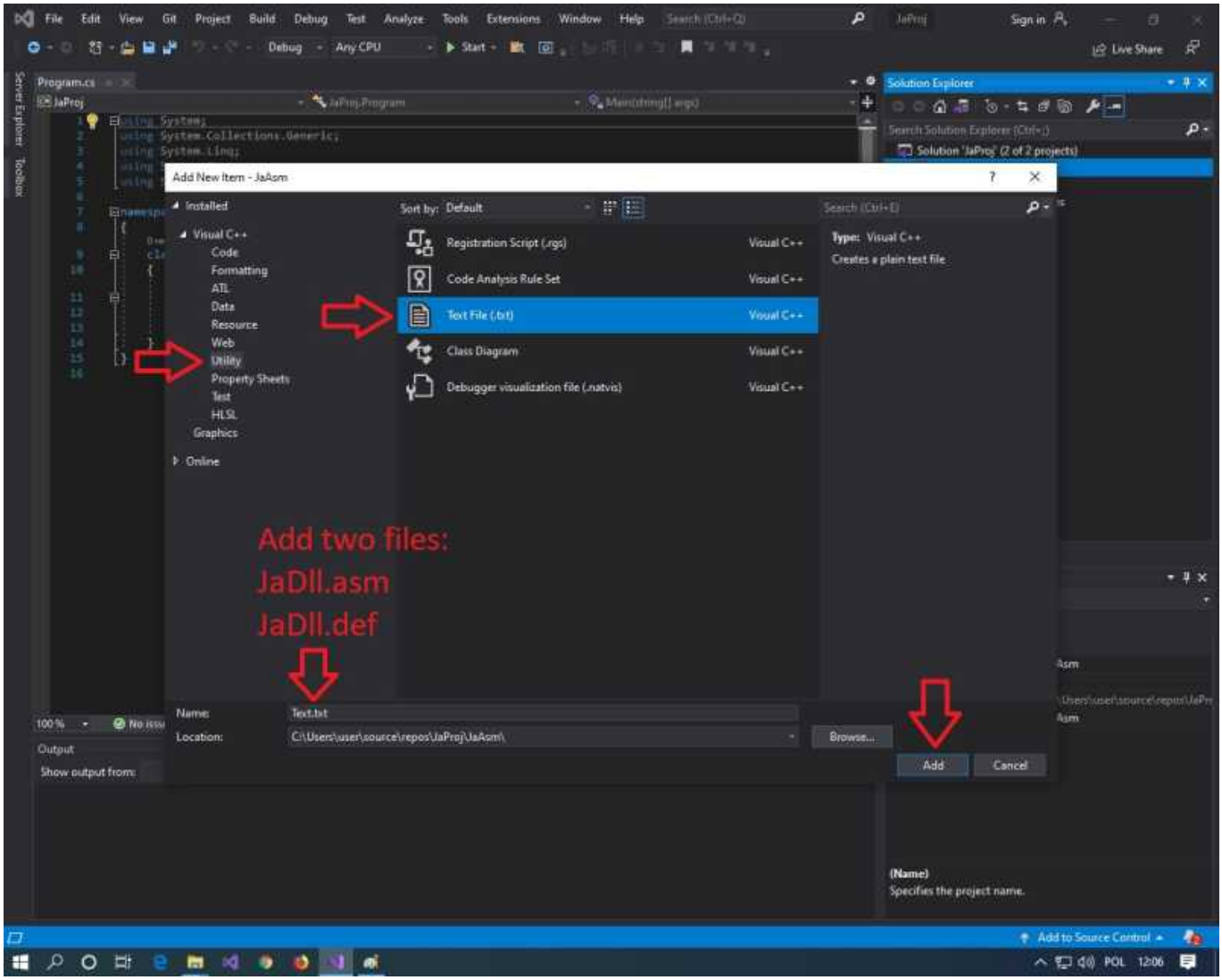
W nowo otwartym oknie zaznaczamy masm i potwierdzamy nasz wybór



Kolejnym krokiem jest dodanie do biblioteki plików zródBowego w asem to zrobi ponownie klikamy prawym przyciskiem myszy na bibliotek Ja Dodaj (Add) oraz Nowy element (New item).

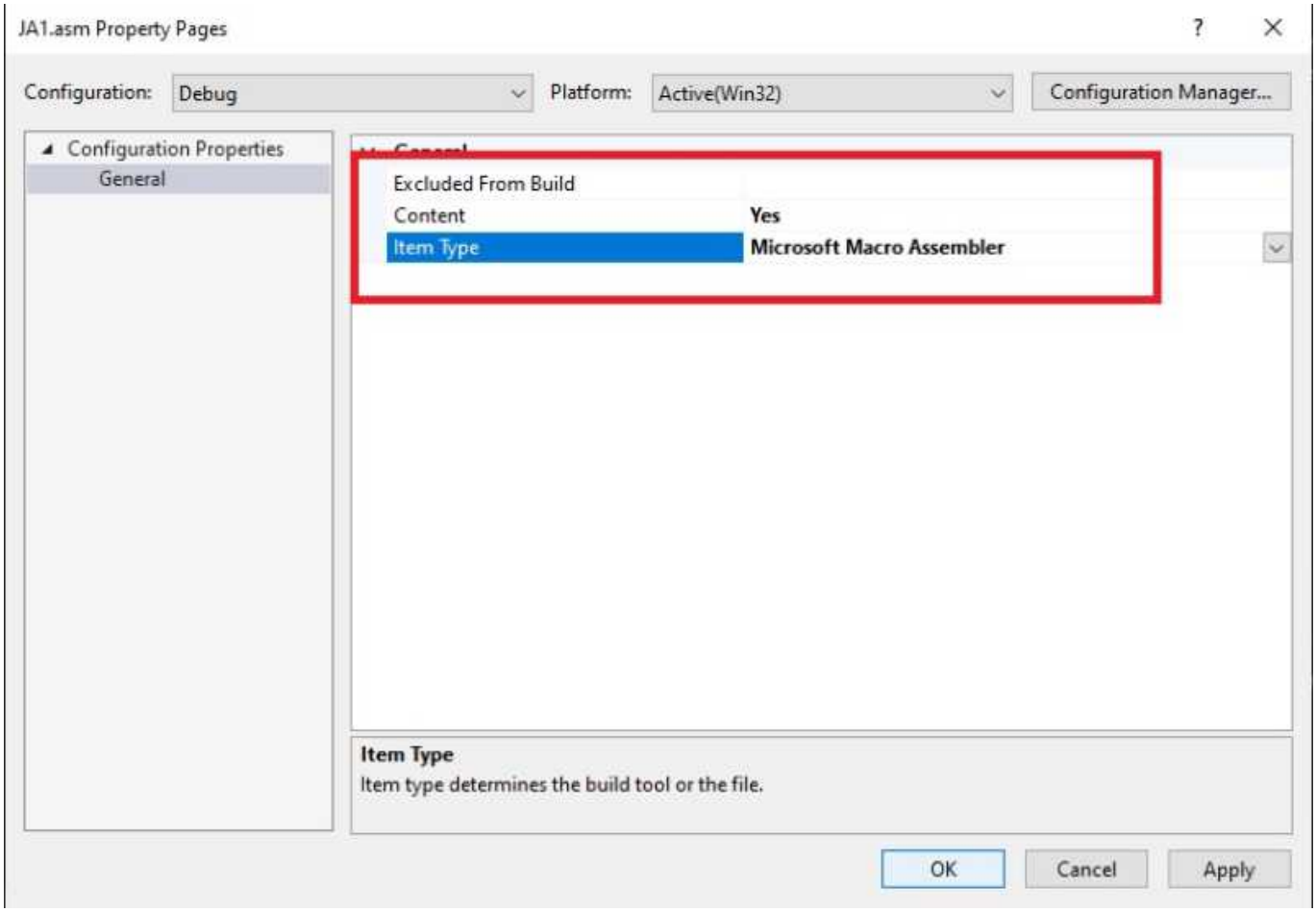


Czynno[t powtarzamy dwukrotnie wybieraj c za ka|dym razem Narz c tekstowy, podaj c nazwy: JaDll.asm oraz JaDll.def .



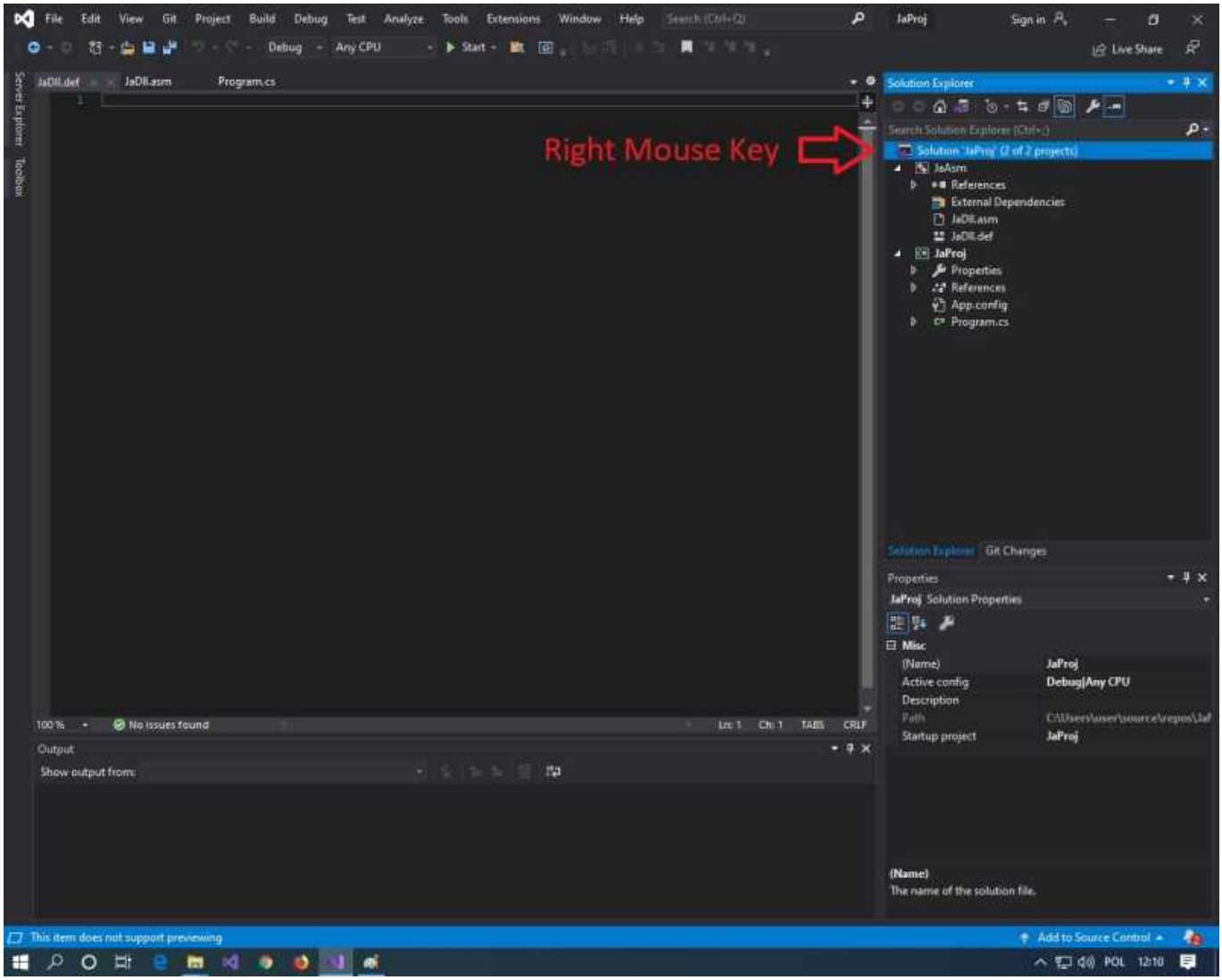
Przy wykonaniu czynno[ci zgodnie z podan uprzedni prawdopodobieDstwem krok przedstawiony na kolejnym rysunku

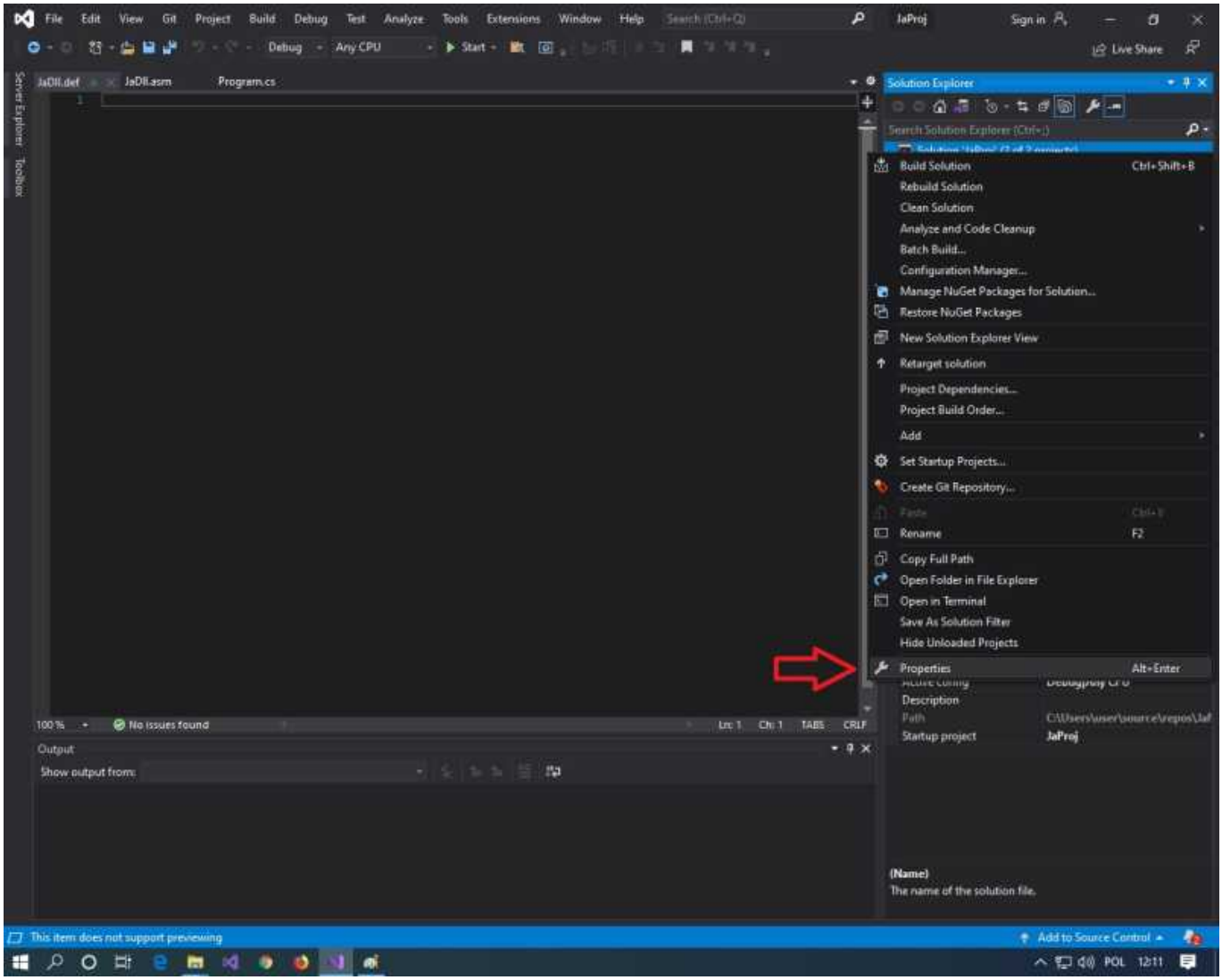
Aby kompilator wiedziaB, |e stworzony plik asemblerowy jest skojarzon typ. Krok ten jest niezb dny gdy wykorzystujemy plik zródBowy C++ i z rozszerzenie na *.asm. Klikaj c pracym klawiszem na pliku *.asm wybie typ pliku na Microsoft Macro Assembler.



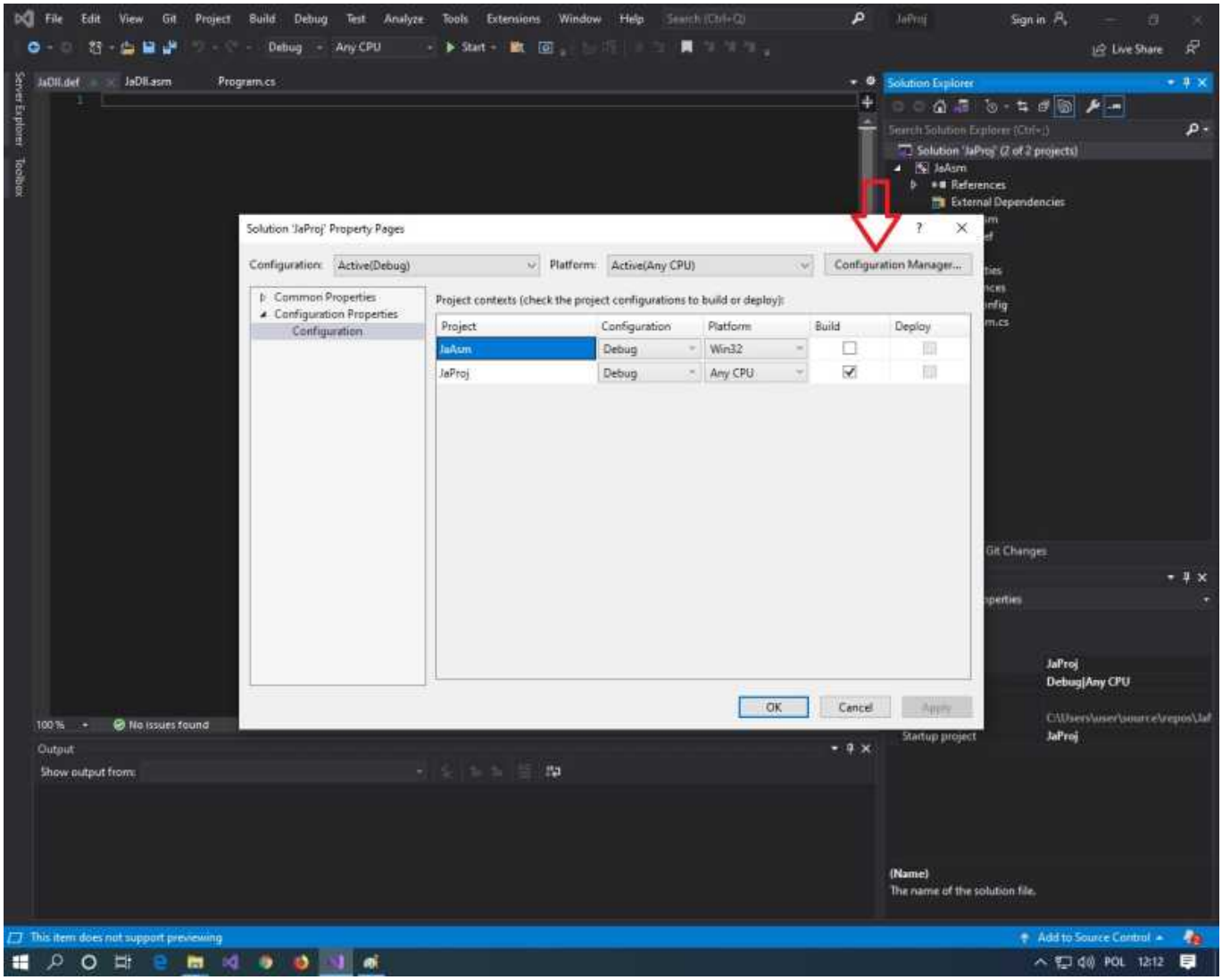
1.1.5 Ustawienie trybu kompilacji

Kompilacja powinna przebiega w trybie 64-bitowym (X64). W tym celu opcjach rozwi zania. Klikamy prawym przyciskiem myszy na nasze rozw WBa[ciwo[ci (Properties).

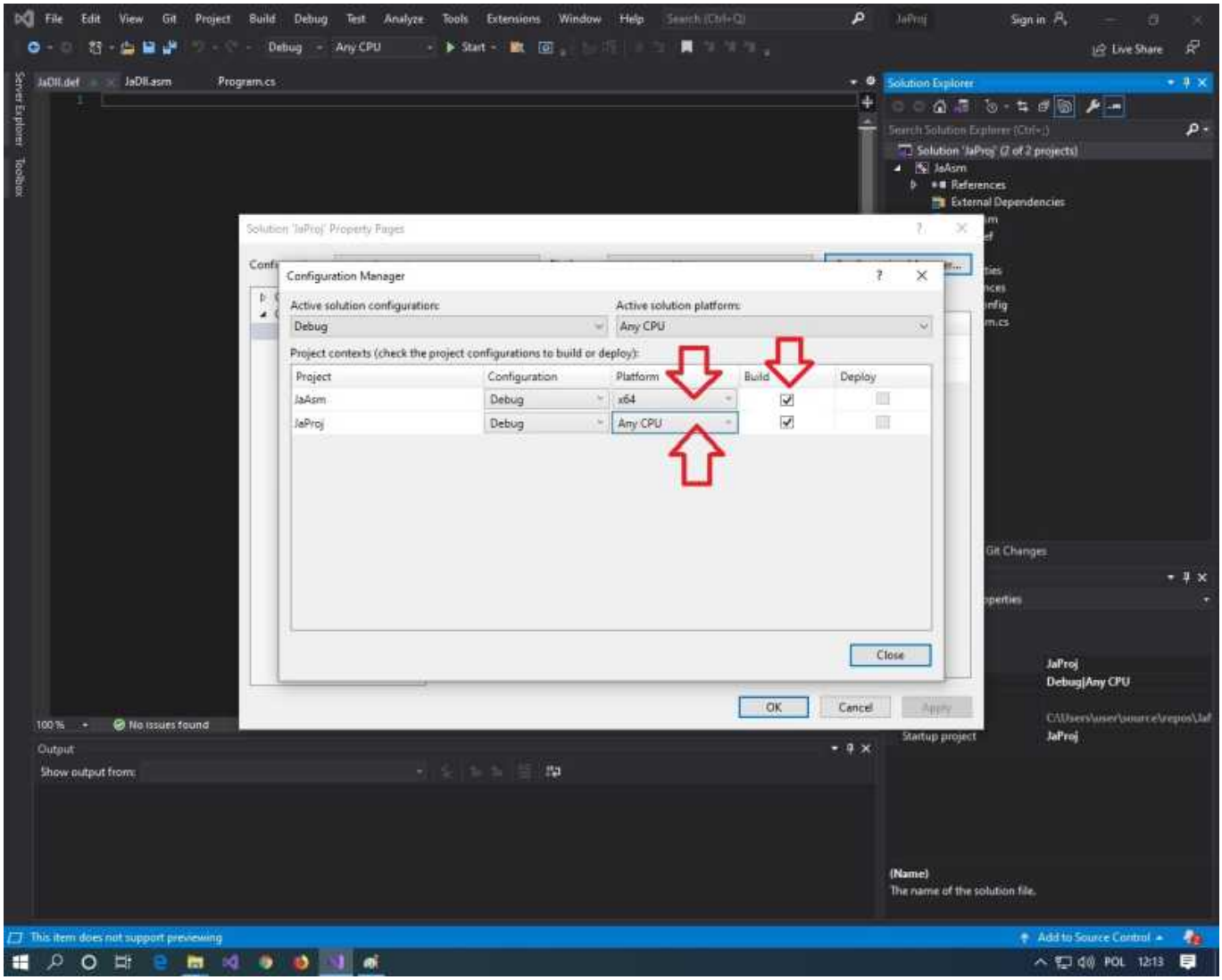

Right Mouse Key

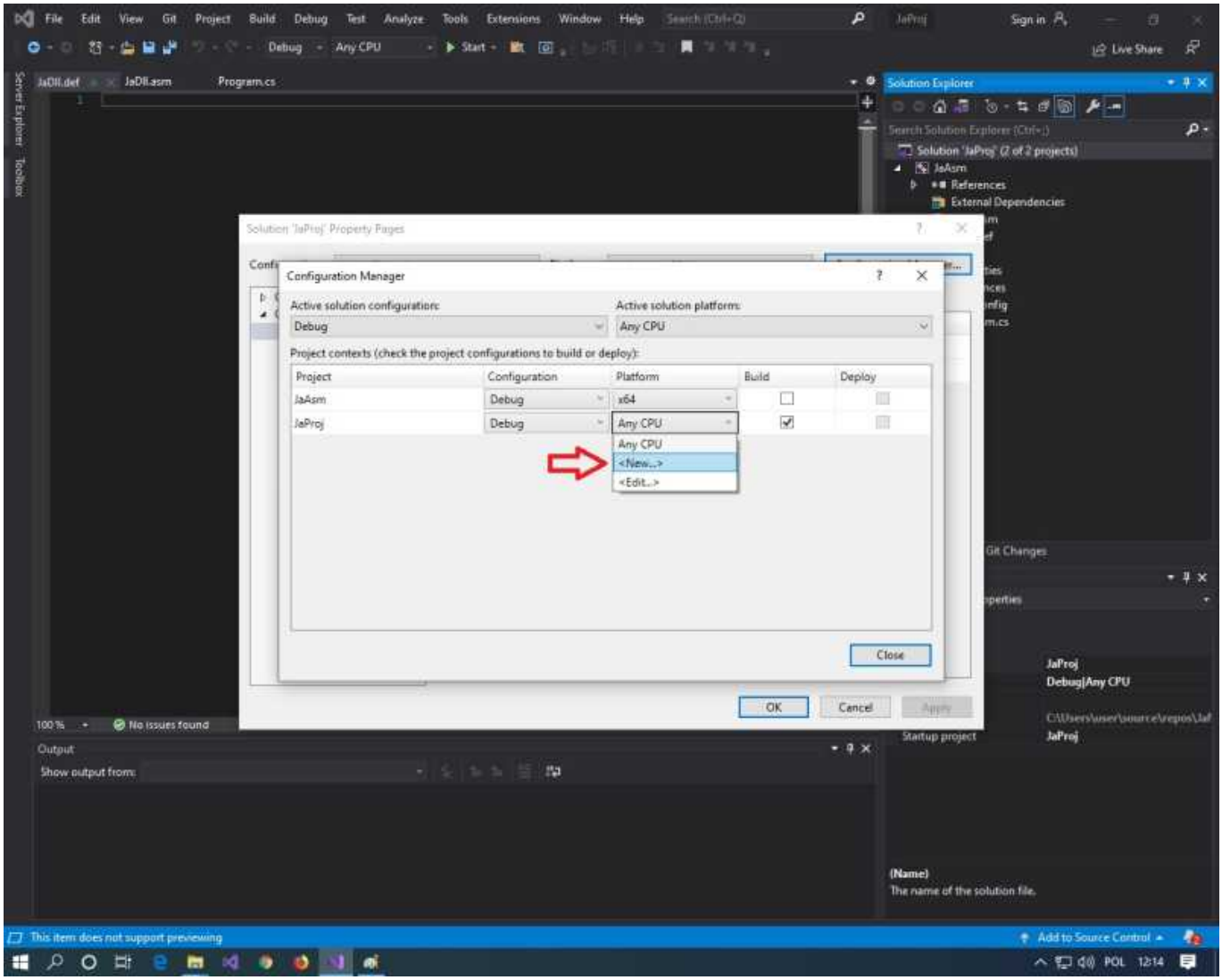


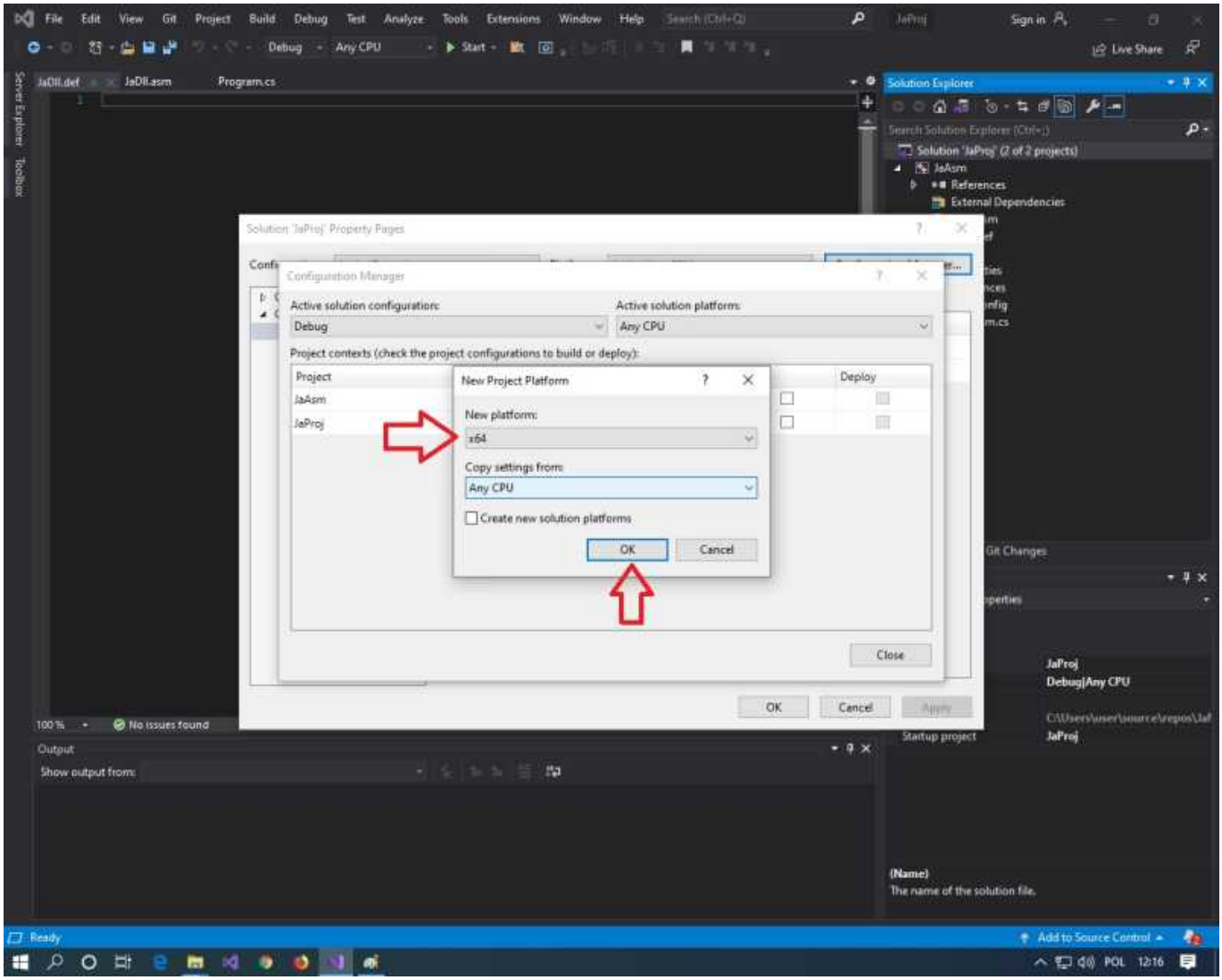
W nowo otwartym oknie przechodzimy na zakBadk WBa[ciwo[ci konû g
(Conû guration properties !' Conû guration) na nast pnie klikamy w Men
(Conû guration manager).



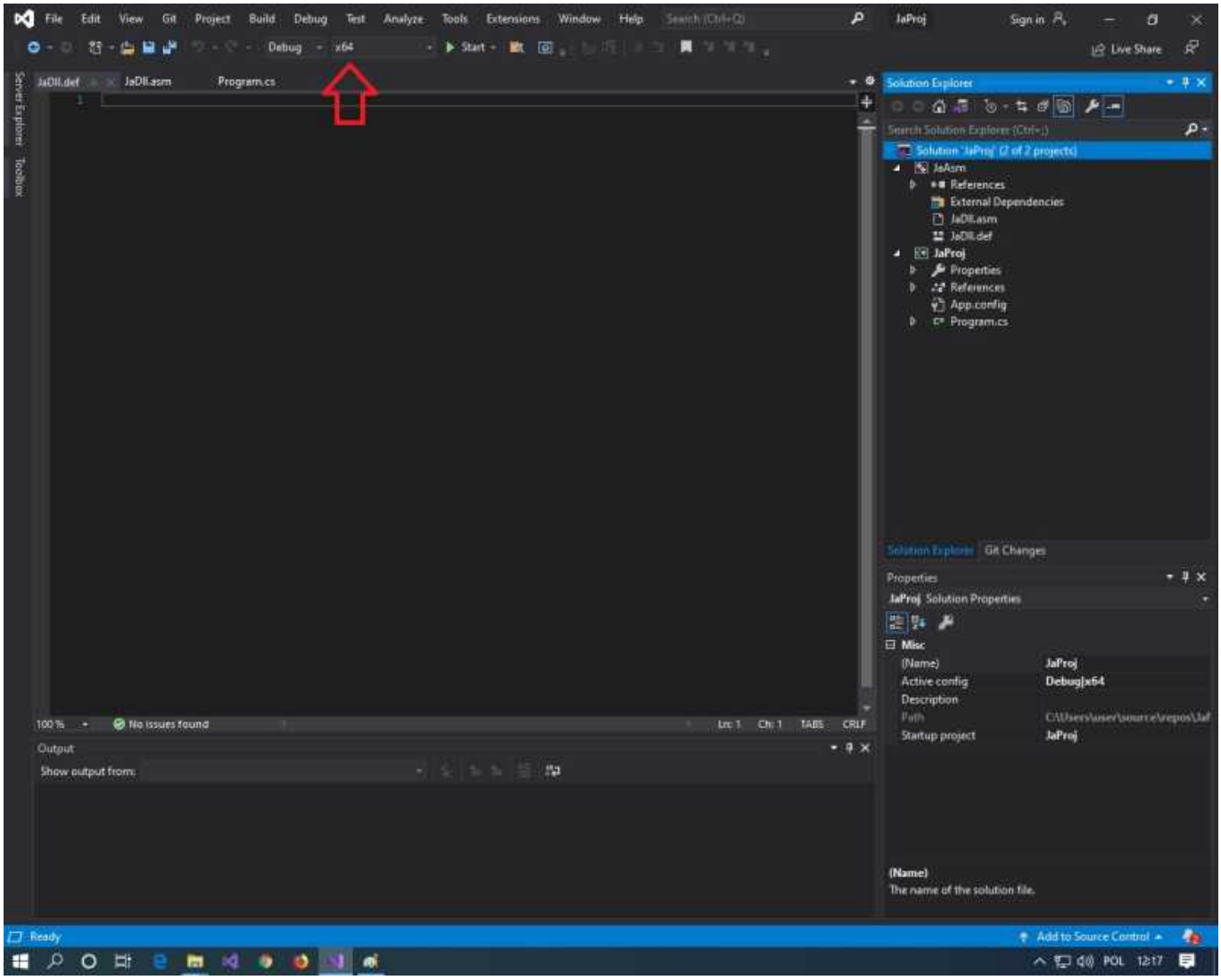
W nowym oknie wskazujemy wsz dzie platform x64 (gdyby dana opcja listy wybieramy pozycj Nowy (New), a nast pnie tworzymy platform ustawieD z Any CPU). Nale|y tak|e przy obu cz [ciach solucji wB czy


Configuration Manager
Active solution configuration:
Debug
Active solution platform:
Any CPU
Project contexts (check the project configurations to build or deploy):
Project
Configuration
Platform
Build
Deploy
JaAsm
Debug
x64
JaProj
Debug
Any CPU
Close
OK
Cancel
Apply


Configuration Manager
Active solution configuration:
Debug
Active solution platform:
Any CPU
Project contexts (check the project configurations to build or deploy):
Project
Configuration
Platform
Build
Deploy
JaAsm
Debug
x64
JaProj
Debug
Any CPU
Any CPU
<New...>
<Edit...>
Close
OK
Cancel
Apply
Solution Explorer
Output
Show output from:
No issues found

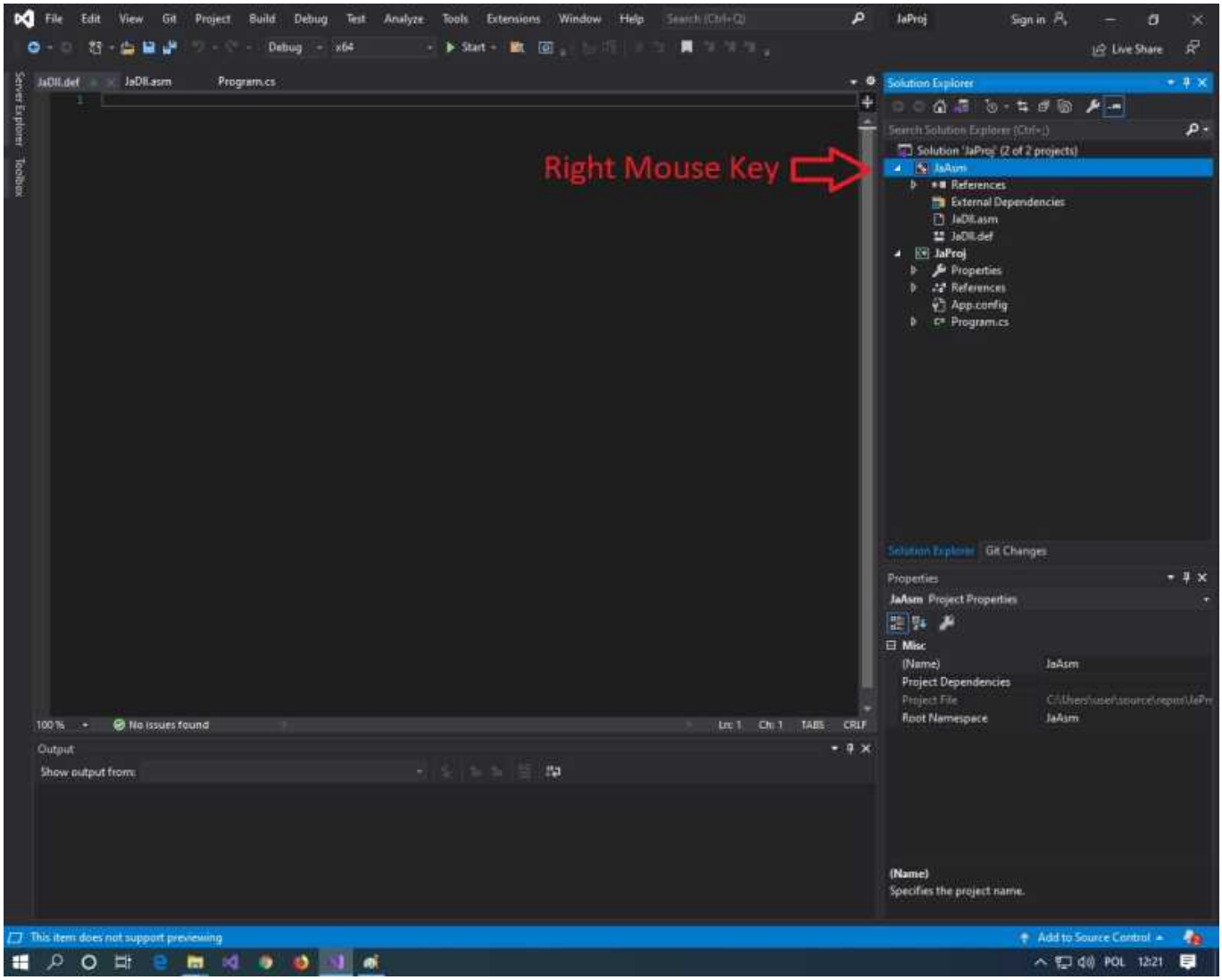


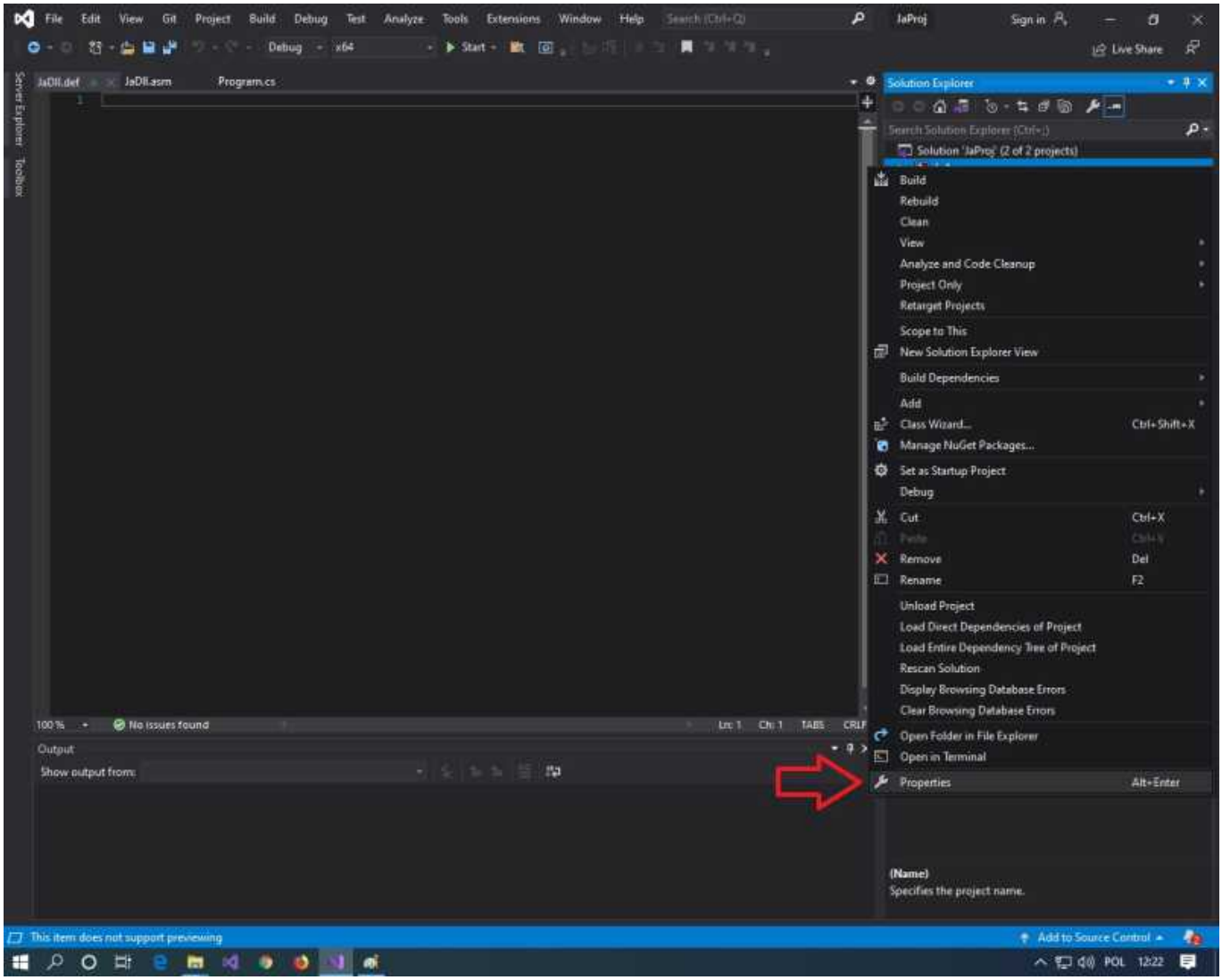
Po wszystkim upewniamy si raz jeszcze, |e wsz dzie wskazana platfor W gBównym pasku menu Visual Studio powinna automatycznie ustawi s warto to sprawdzi i w razie potrzeby zmodyû kowa .



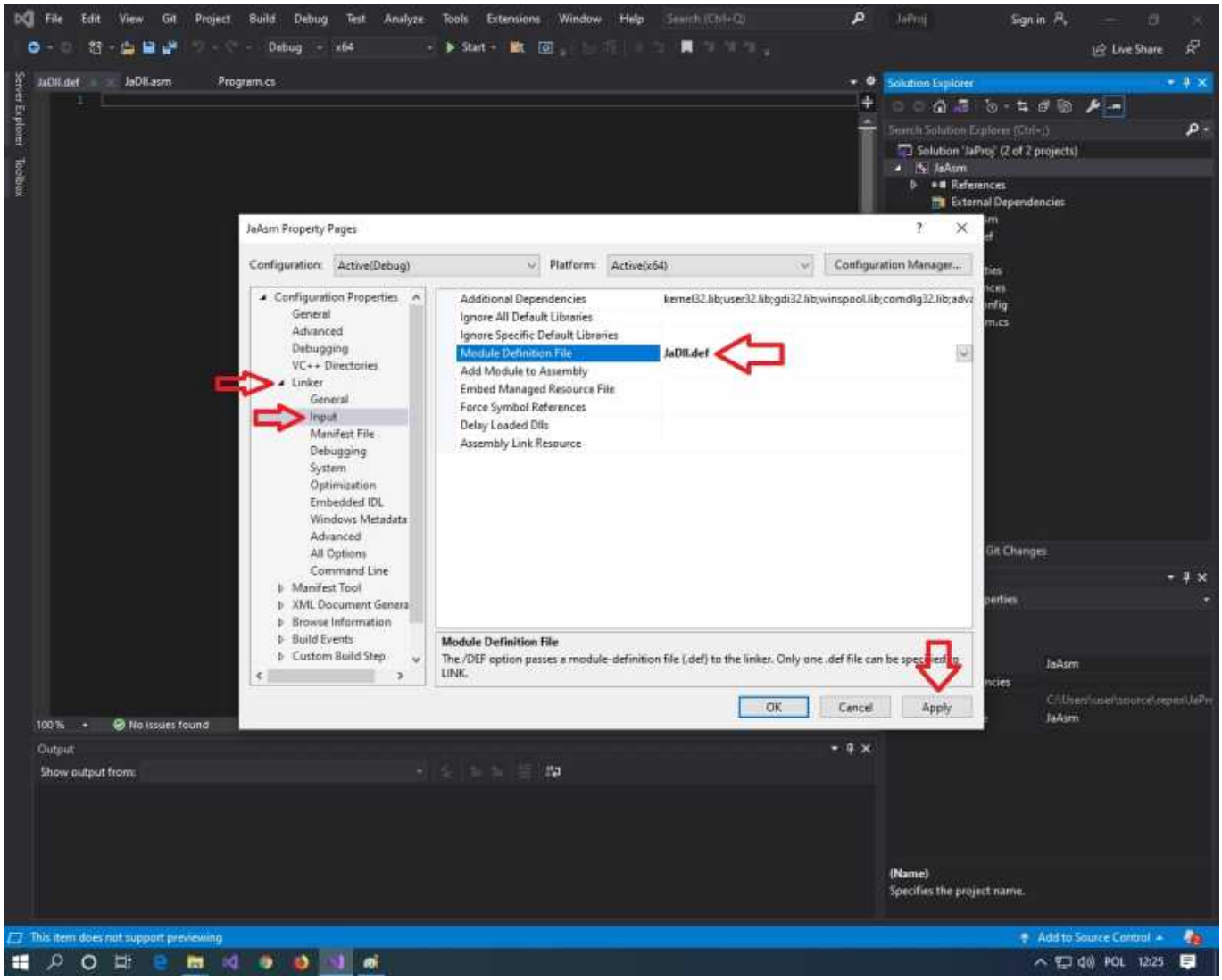
1.1.6 Końcowa konfiguracja biblioteki asemblerowej

Przystępujemy do konfiguracji biblioteki. W tym celu klikamy prawym przyciskiem myszy na projekt i wybieramy Właściwości (Properties).

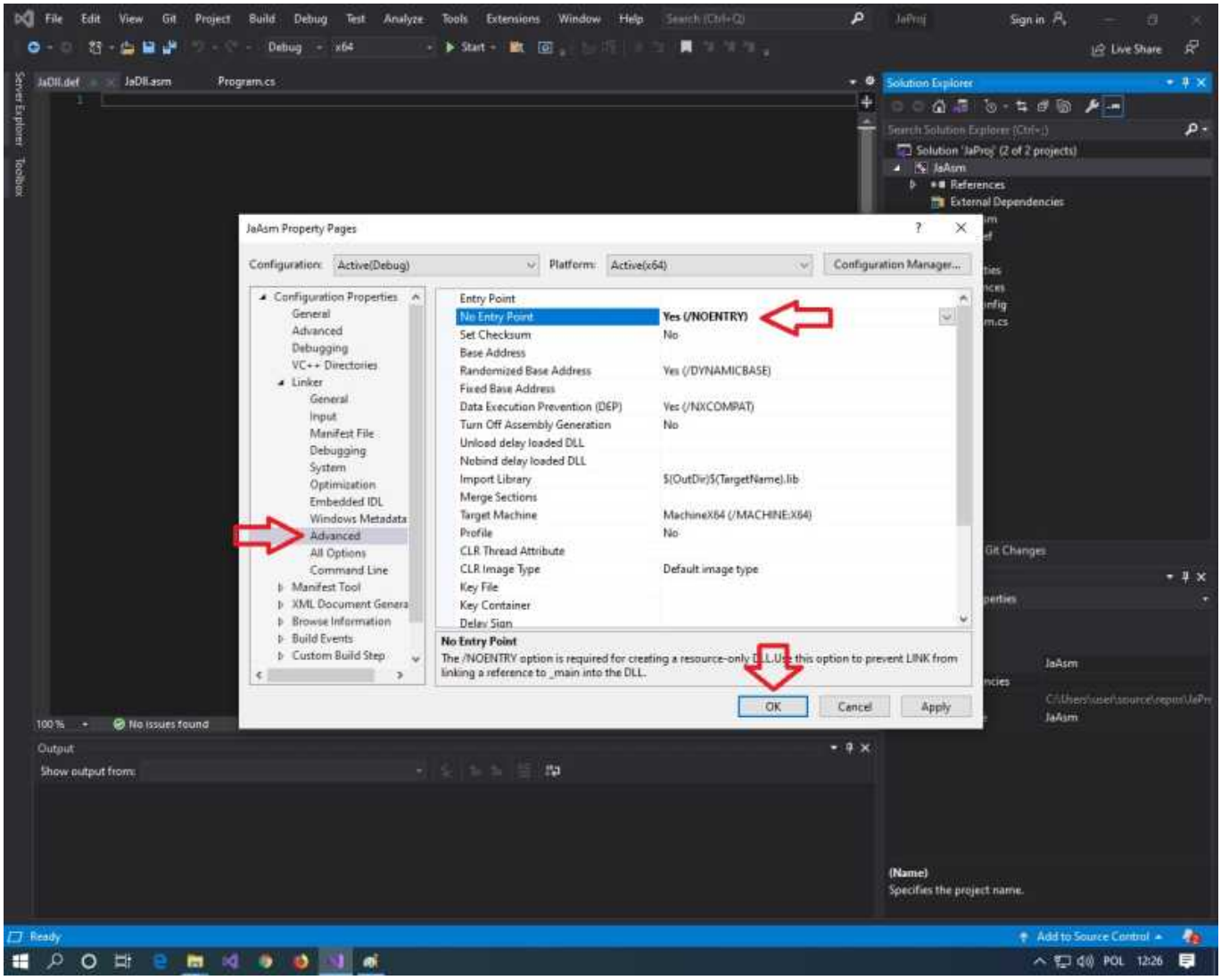

Right Mouse Key



W otwartym nowo oknie przechodzimy do Konsolidator !' Dane wej[cio polu Plik deû nicji moduBu (Module deû nition û le) wpisujemy nazw na przypadku JaDll.def . Wpis potwierdzamy.



KoDcowym krokiem konû guracji jest wyB czenie punktu wej[cia do bibli Konsolidator (Linker) przechodzimy do opcji Zaawansowane (Advan wej[cia (No entry point) wybieramy opcj Tak (/NOENTRY). Zapisujer zamykamy okno.



Biblioteka dll, w przeciwieDstwie do programu gBównego, startowego, który wykorzystuje system operacyjny do u Udost pnia jedynie zestaw funkcji dla innych programów.

1.2 Wykorzystanie biblioteki dll napisanej w ase

1.2.1 Biblioteka asemblerowa dll

Biblioteka dll zawiera funkcje, które mo|na u|y w innych programach. biblioteki b dzie udost pniaBa jedn funkcj , zwracaj c staB warto[. odczytamy w pliku C++. PrzykBadowa zawarto[pliku asemblerowego (. nast puj co.

```
.code
MyProc proc
mov RAX, 2021
ret
MyProc endp
```

```
end
```

Z plikiem kodu programu nierozerwalnie zwi zany jest plik deû nicji (Ja
nazw biblioteki DLL jako agrument dyrektywy LIBRARY, oraz list nazw
moduBów poprzedzonych dyrektyw EXPORTS. Aby nasza funkcja MyPro
C++ musimy j zadeklarowa jako eksportowan . Plik JaDll.def w naszy
nast puj co.

```
LIBRARY JaAsm
EXPORTS MyProc
```

Cz sto chcemy wykorzysta przekazane do funkcji parametry. Przypomn
konwencji wywoBaD pierwsze cztery parametry s umieszczane bezpo[r
SzczegóBy s opisane w instrukcji do pierwszego wiczenia, dwa parametry typu caBko
umieszczone w rejestrach RCX i RDX. Funkcja, któa zwraca sum dwóc
wygl daBaby w nast puj cy sposób.

```
...
MyProc proc
mov RAX, RCX
add RAX, RDX
ret
MyProc endp
...
```

Argumenty typu zmiennoprzecinkowego nale|y przekaza przez rejestry
zwróci przez XMM0. Oczywi[cie nale|y u|y instrukcji rozszerzeD MMX

```
...
FloatProc PROC
addsd xmm0, xmm1
ret
FloatProc ENDP
...
```

1.2.2OdwoBanie do biblioteki dll z programu C++

Aby program napisany w C++ mógB wykorzysta funkcj z biblioteki nal

1. Przygotowa typ funkcji zgodny z jej pózniejszym wywoBaniem,
2. ZaBadowa bibliotek funkcj LoadLibrary uzyskuj c jednocze[nie je
3. Odczyta aktualny adres interesuj cej nas funkcji wywoBuj c GetPr
4. U|y funkcji odwoBuj c si do jej adresu.

PrzykBadowy kod mógBby wygl da tak:

```
#include <windows.h>
#include <iostream>
typedef int(_stdcall* MyProc)();
```

```
HINSTANCE dllHandle = NULL;

int main()
{
    dllHandle = LoadLibrary(TEXT("JaAsm.dll"));
    MyProc procedura = (MyProc)GetProcAddress(dllHandle, "MyProc")
    std::cout << procedura();
}
```

U|ycie funkcji z parametrami wymaga nieco zmodyû kowanych deû nicji t
uwag na typ warto[ci zwracanej przez funkcj . Pami ta nale|y, |e w pr
zmiennoprzecinkowych, rozmiar rejestrów XMM0, XMM1 to 64 bity. Odp
zmiennoprzecinkowej podwójnej precyzji, czyli w j zyku C++ typowi do

```
...
typedef int(_stdcall* MyProc)(int, int);
typedef double(_stdcall* FloatProc)(double, double);
...
```

1.2.3 Zadanie do zrealizowania w ramach laboratorium

Nale|y napisa bibliotek DLL w asemblerze i program w C++ demonstru
Biblioteka DLL udost pnia funkcje wykonuj ce i zwracaj ce wynik podst
arytmetycznych (dodawanie, odejmowanie, mno|enie i dzielenie) na pa
parametry ich wywoBania. Funkcje powinny mie wersj staBo- i zmienn
funkcji).
Program w C++ wywoBuje wszystkie funkcje przekazuj c przykBadowe
wy[wietli wyniki na ekranie. Nie jest wymagany interfejs u|ytkownika
by umieszczone w kodzie programu.

From:
https://dokuwiki.zmitac.aei.polsl.pl/ - RAU6 Dokuwiki

Permanent link:
https://dokuwiki.zmitac.aei.polsl.pl/en:edu:zmitac:s1:apl:lab=17325

Last update: 2024/11/25 11:51