Instrukcja zakładania projektu asm + cpp w Visual Studio 2017

Wersja 15.6.2

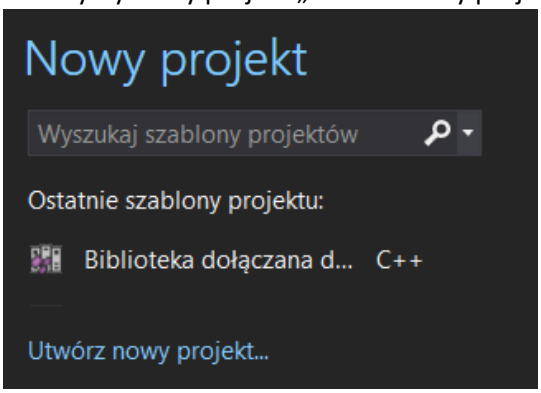
15.03.2018
Krzysztof Hanzel

Spis treści

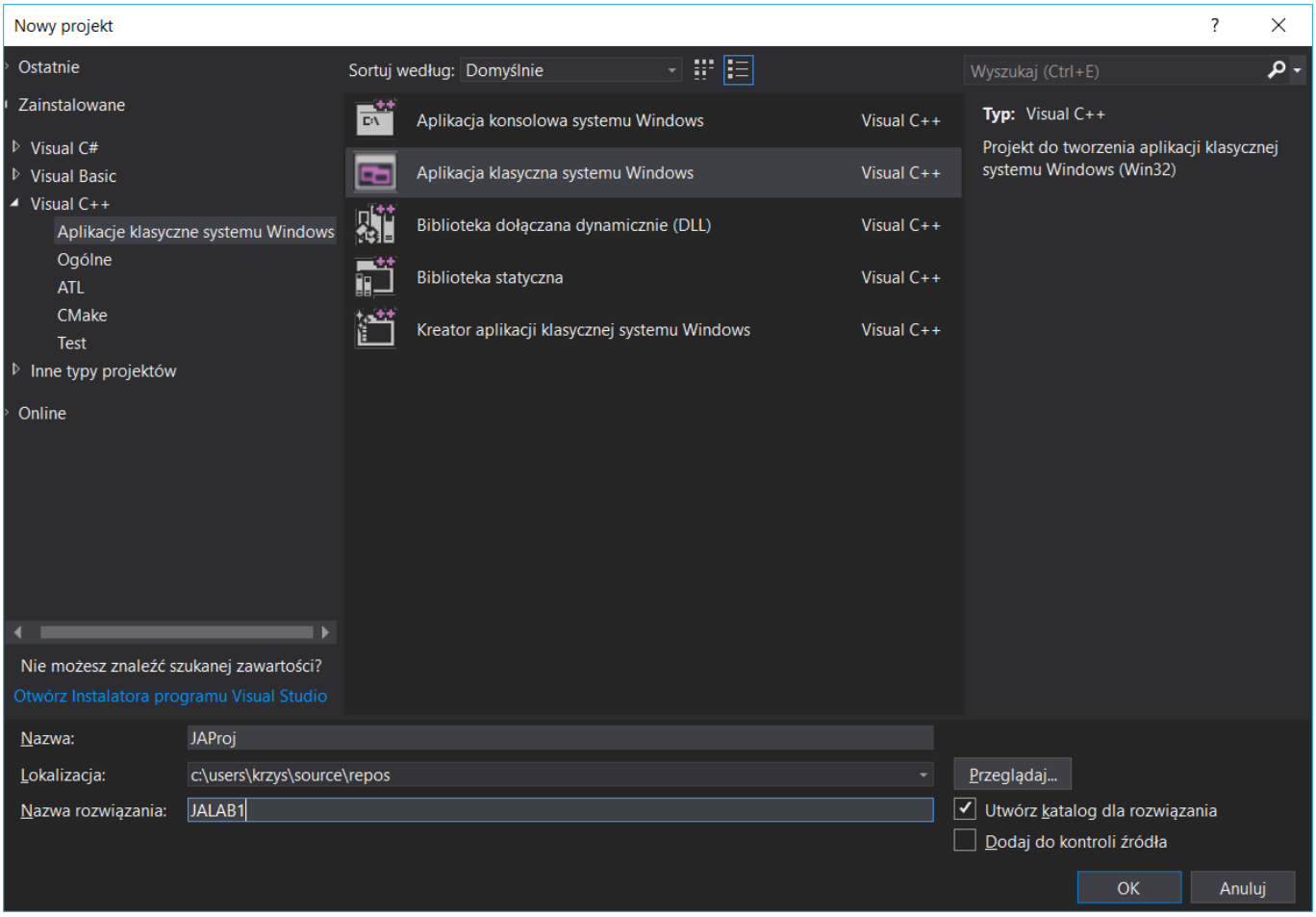
Aplikacja x64

Tworzenie projektów

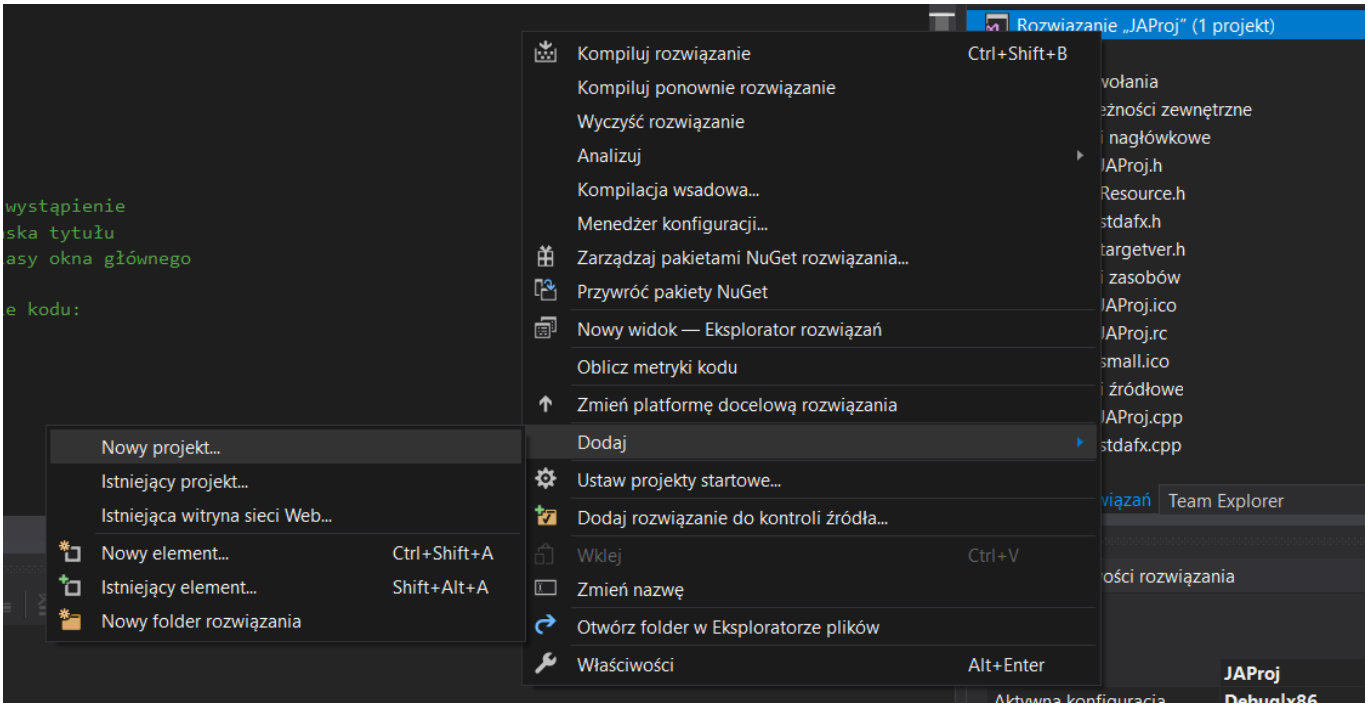
1. Tworzymy nowy projekt „Utwórz nowy projekt…”



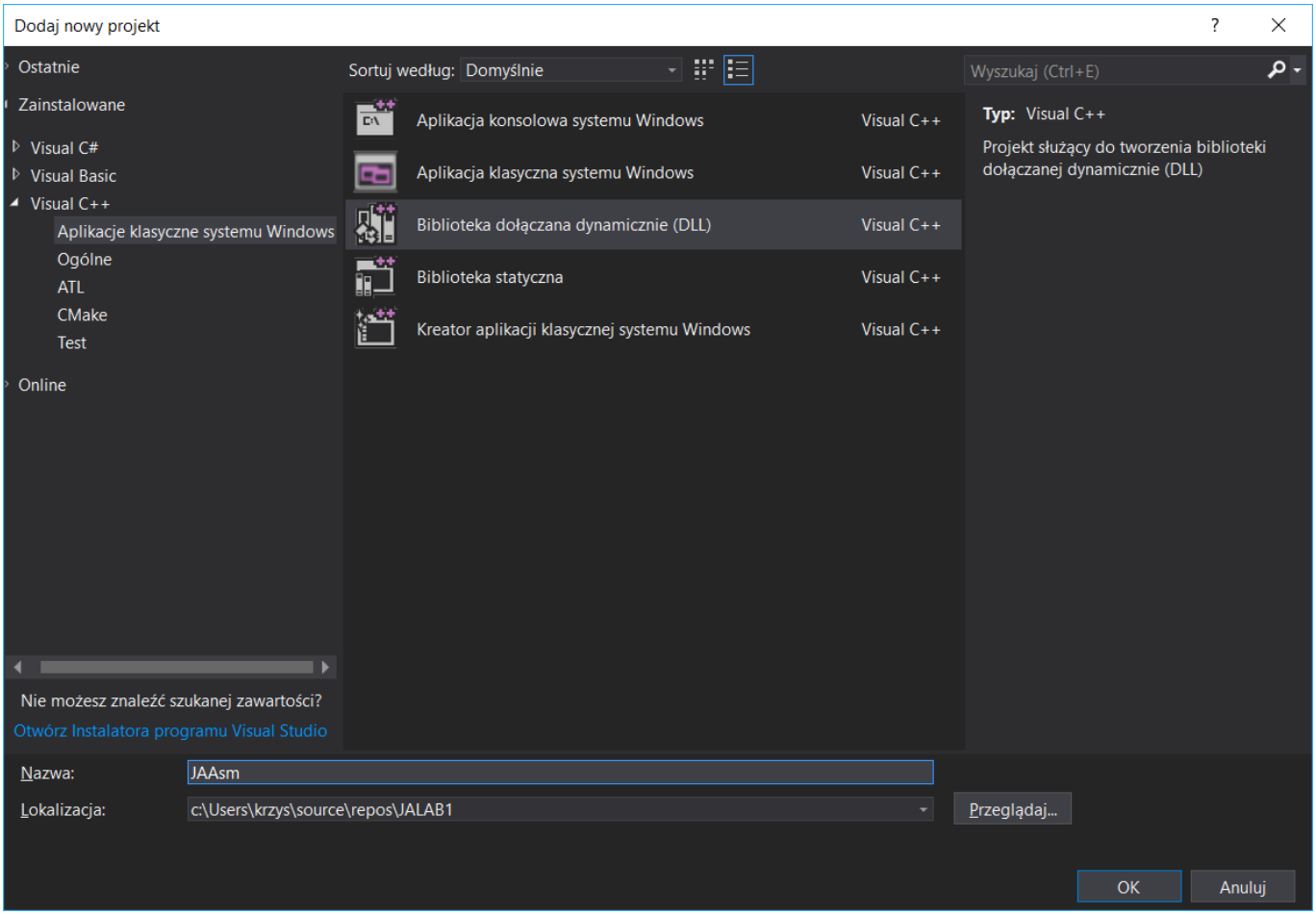
2. Konfigurujemy projekt jako Aplikacja klasyczna systemu Windows. Nazwa to JAProj a Nazwa rozwiązania to JALAB1



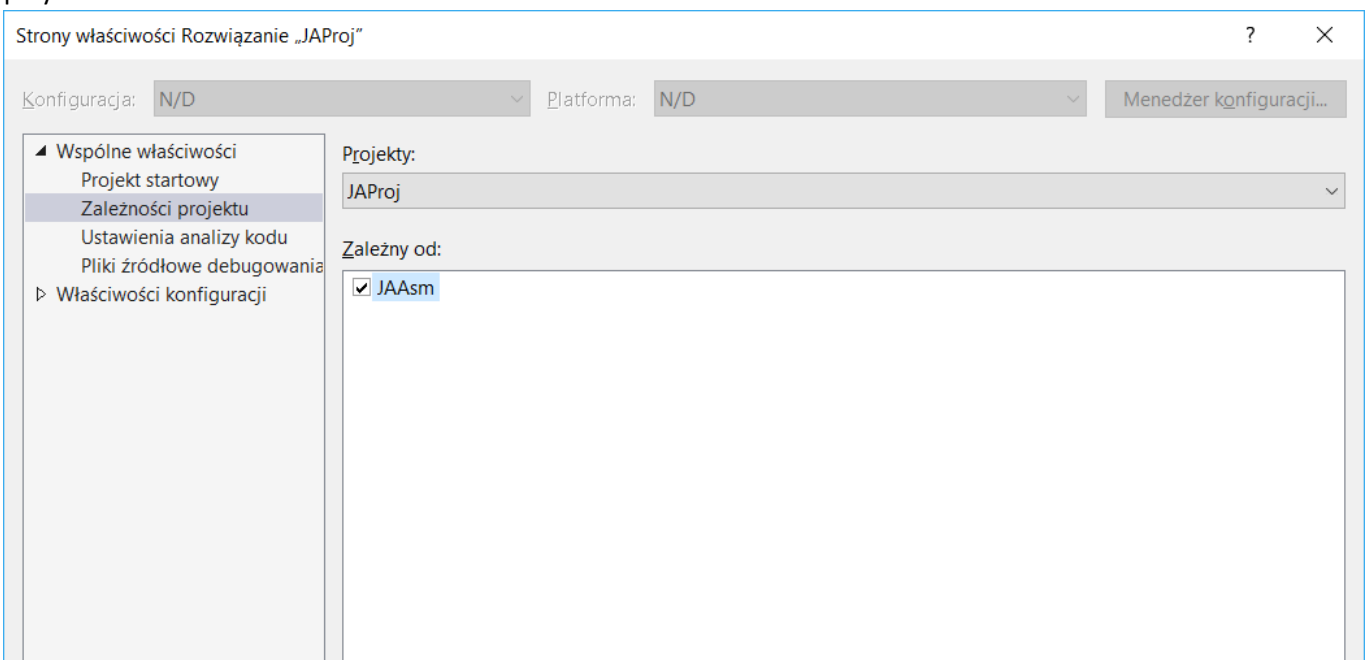
3. Dodajemy nowy projekt (PPM na projekt, następnie Dodaj oraz Nowy projekt)



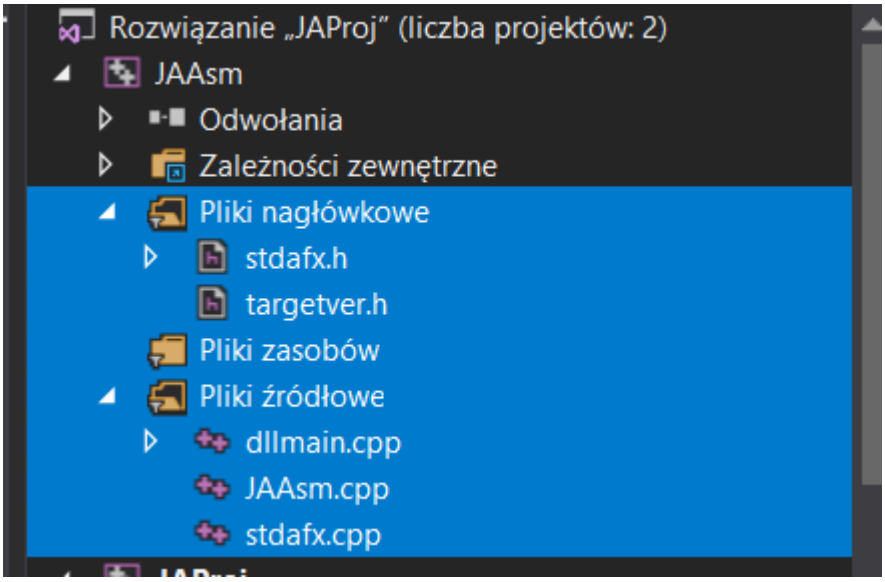
4. Wybieramy bibliotekę dynamiczną (DLL) oraz jako nazwę podajemy JAAsm:



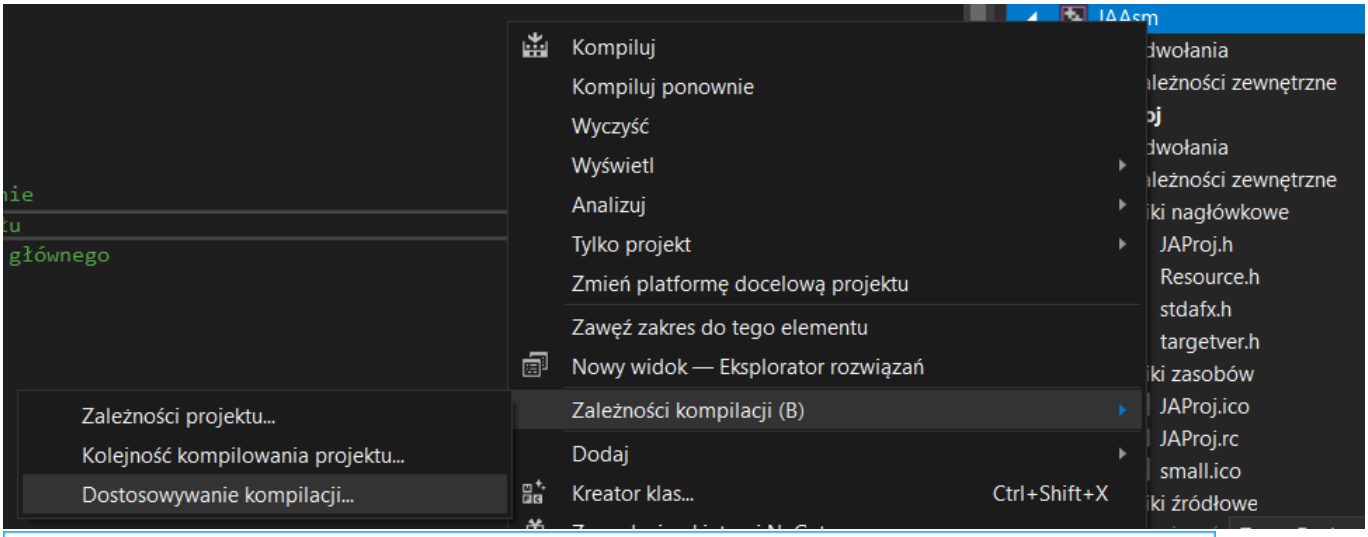
5. Po dołączeniu biblioteki JAAsm, należy ustawić jej zależność kompilacji względem projektu w języku wysokiego poziomu. W tym celu klikamy PPM na solucję i wybieramy właściwości. Następnie przechodzimy na zakładkę wspólne właściwości, zależności projektu i zaznaczamy pole wyboru Zależny od przy JAAsm:

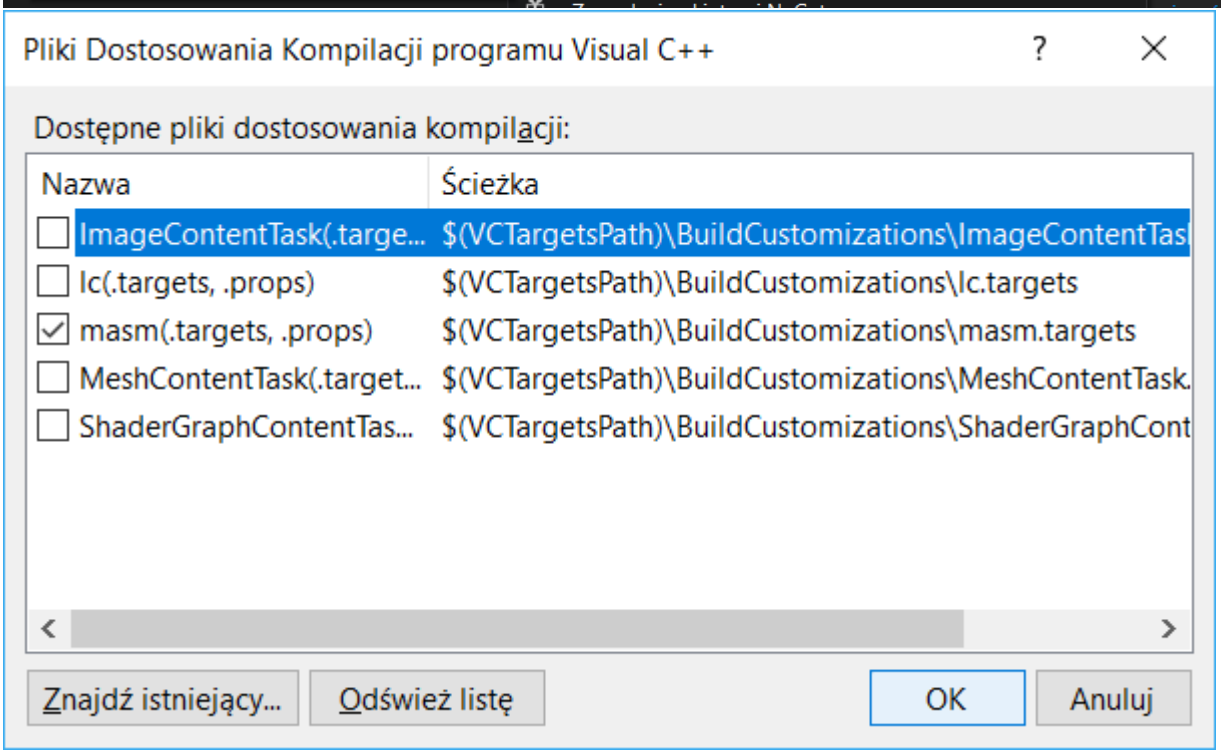


6. Można skasować powstałe w bibliotece pliki:

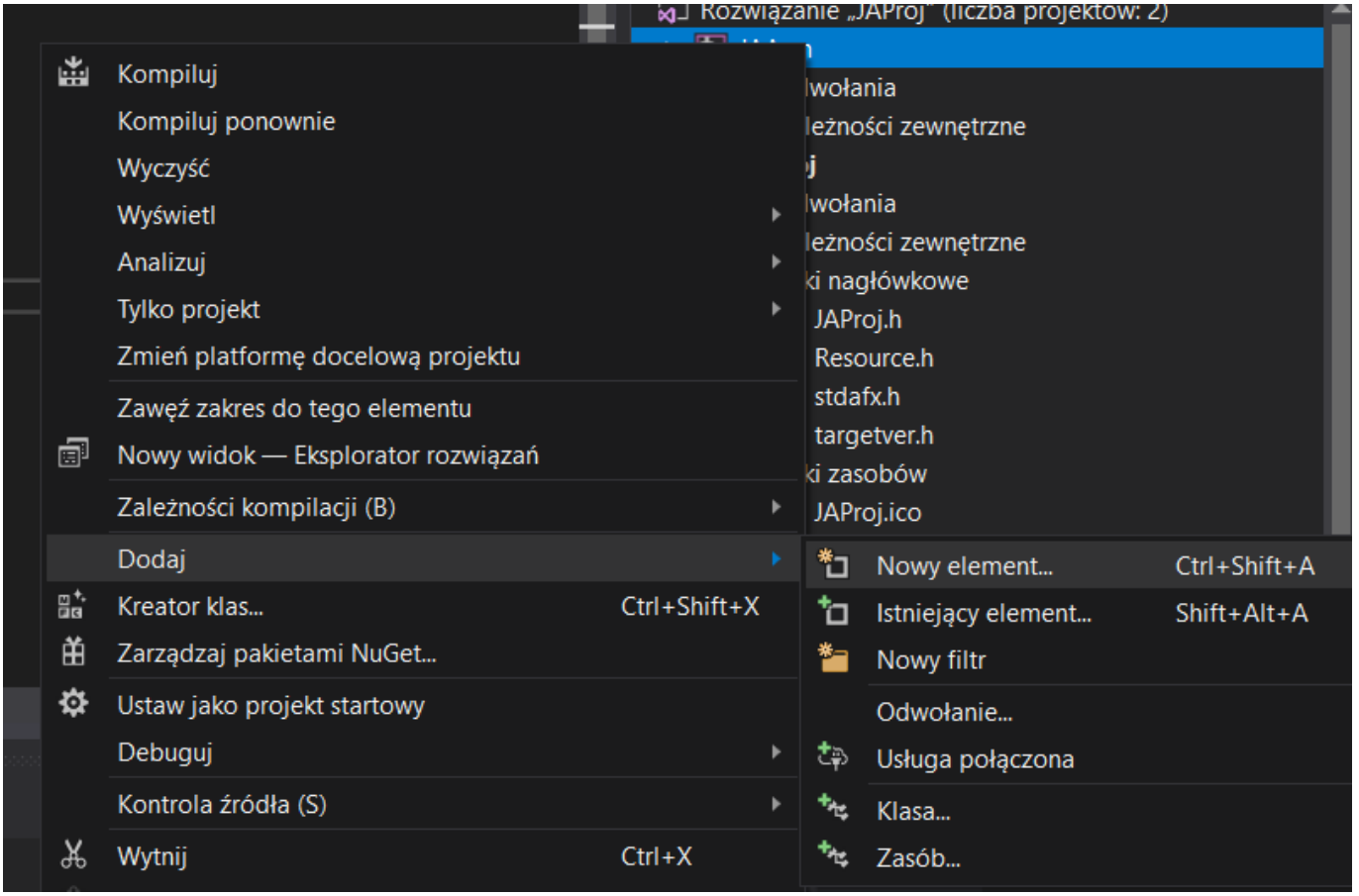


7. W zależnościach kompilacji biblioteki zmieniamy dostosowanie kompilacji do masm:

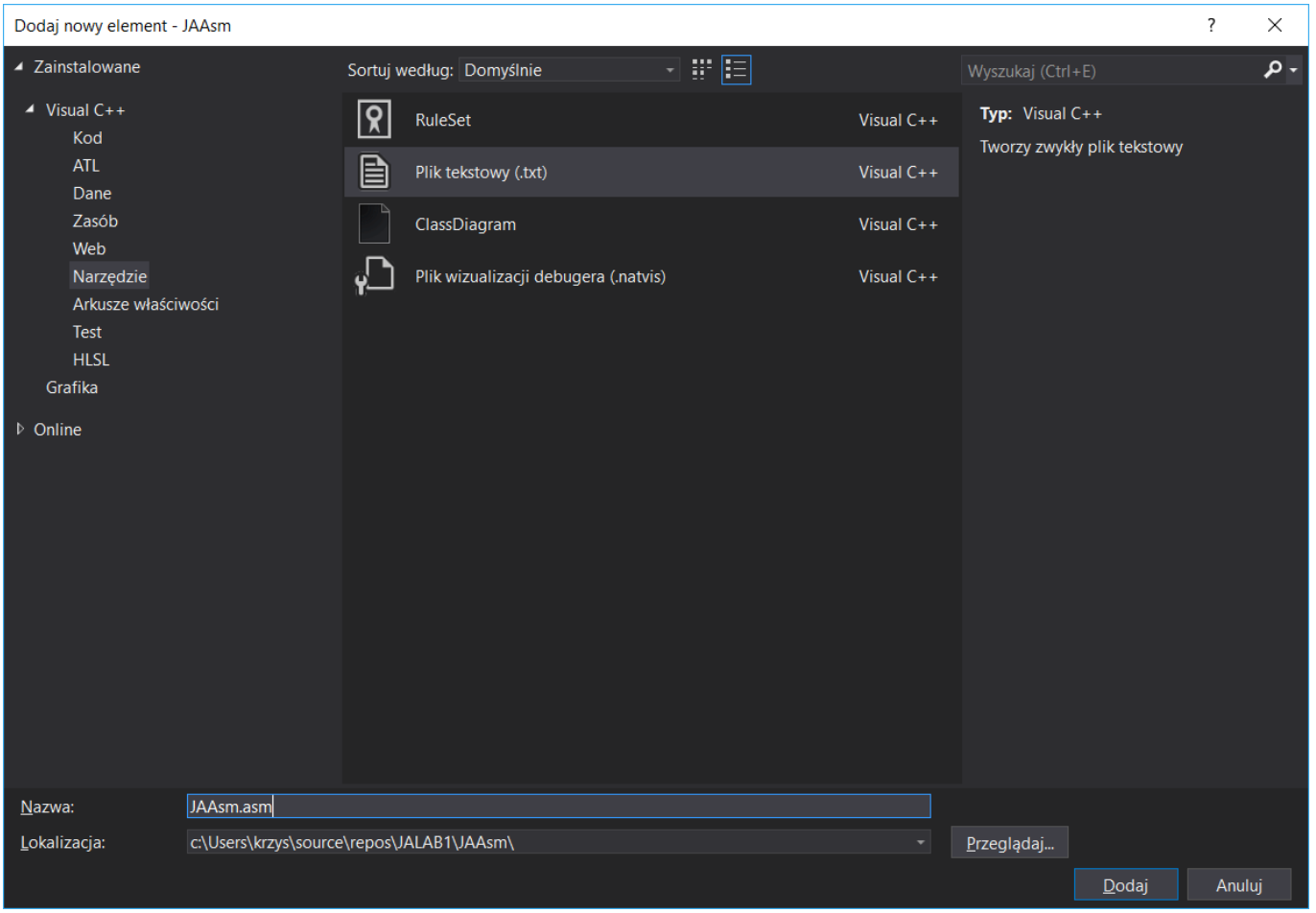




8. Dodajemy do biblioteki nowy element w postaci pliku asm. W tym celu wybieramy:

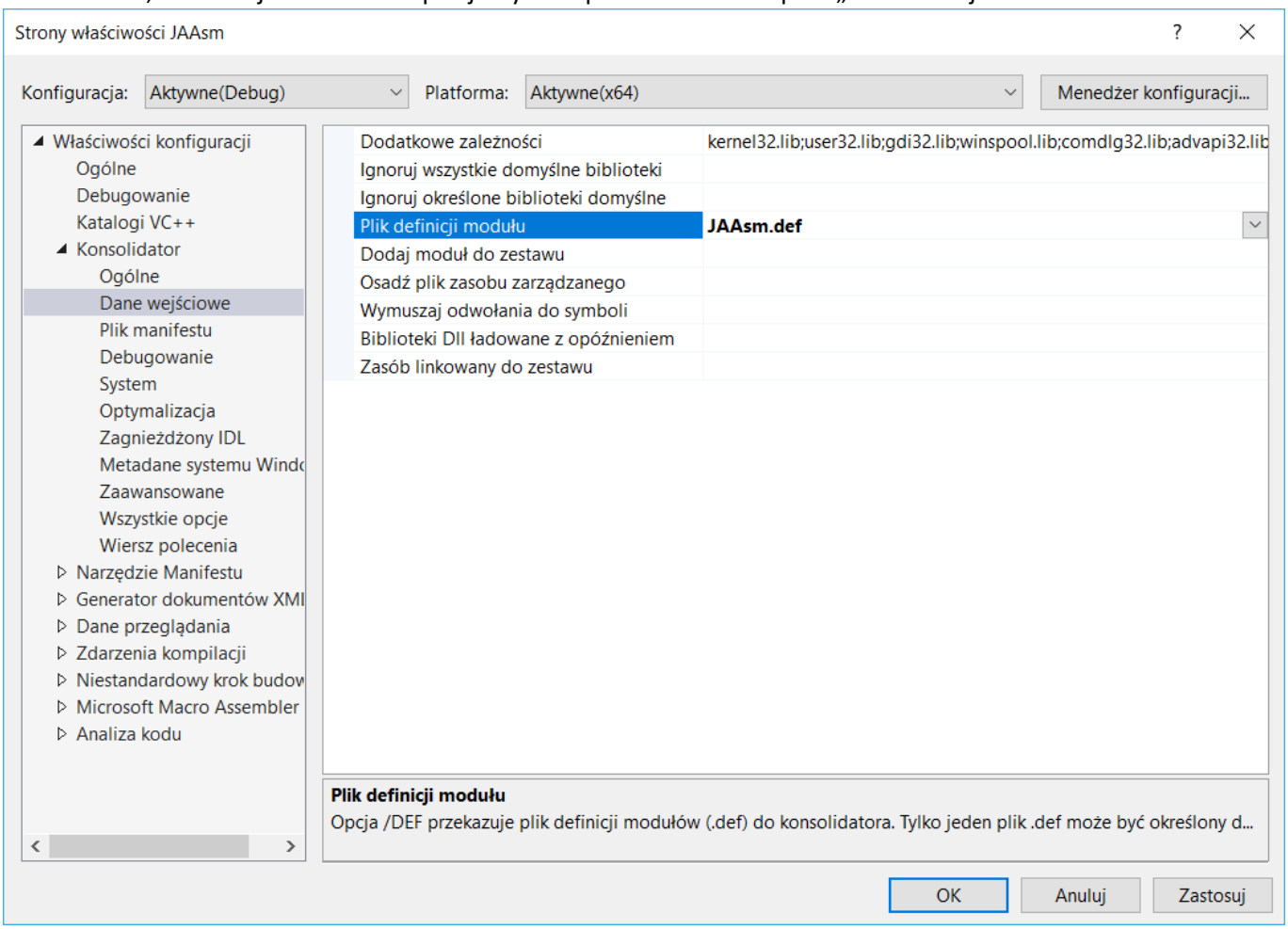


9. Następnie należy wybrać plik tekstowy (Visual C++, Narzędzie) oraz zmienić jego nazwę na JAAsm.asm:

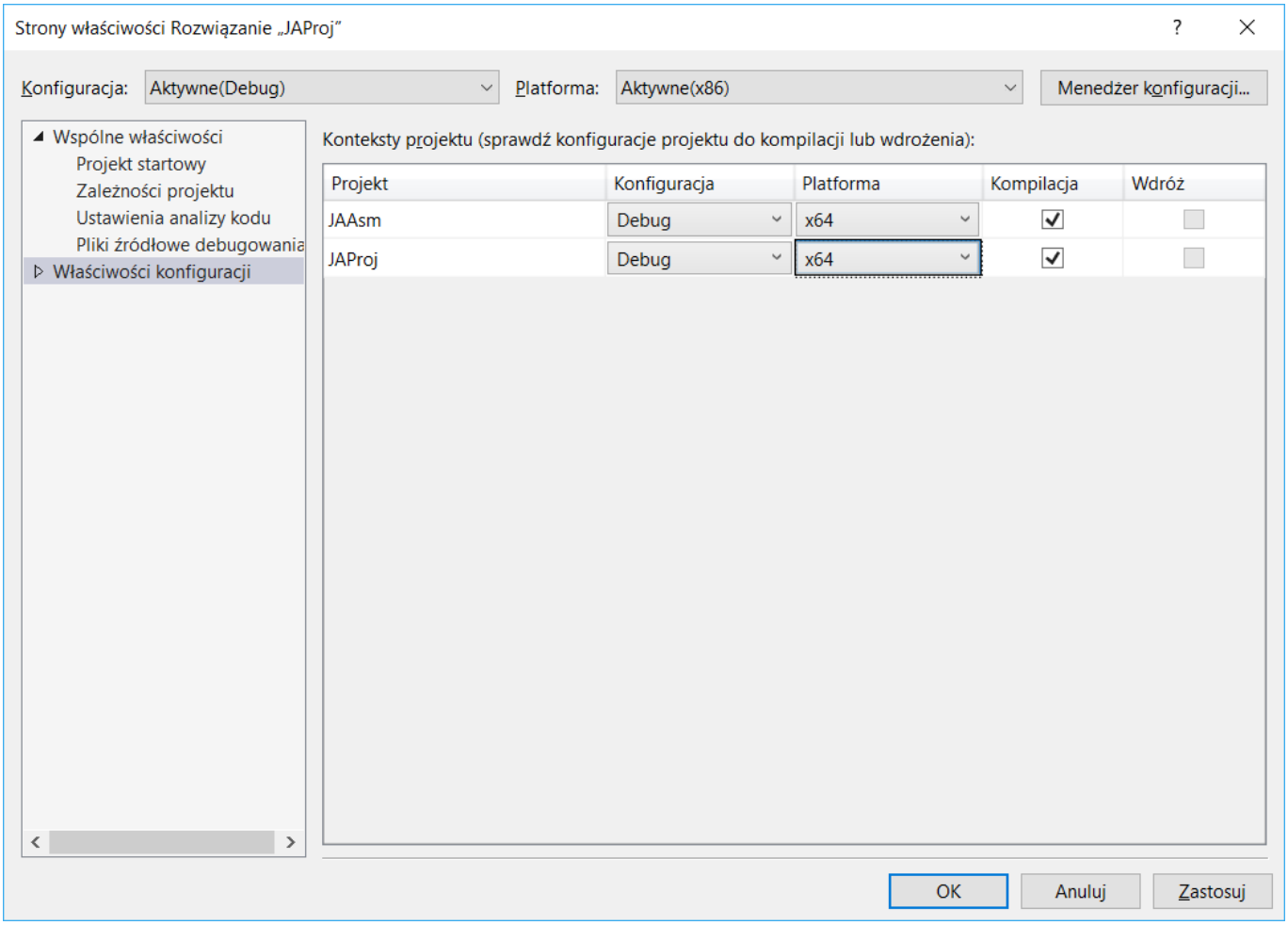


10. Analogicznie dodajemy plik JAAsm.def

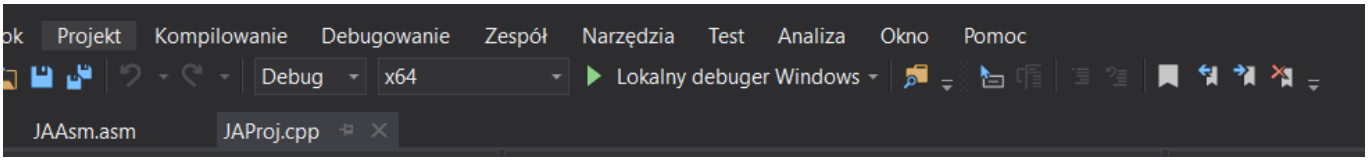
11. Po dodaniu pliku definicji modułu, należy podpiąć go pod linkera. W tym celu przechodzimy do właściwości biblioteki (PPM, Właściwości), następnie przechodzimy do zakładki Właściwości konfiguracji, Konsolidator, Dane wejściowe oraz wpisujemy nasz plik JAAsm.def w polu „Plik definicji modułu”:



12. Zmieniamy właściwości kompilacji na x64. W tym celu należy kliknąć PPM na solucję oraz wybrać właściwości. Następnie przechodzimy na zakładkę Właściwości Konfiguracji, oraz zmieniamy dostępne tam parametry platformy na x64:

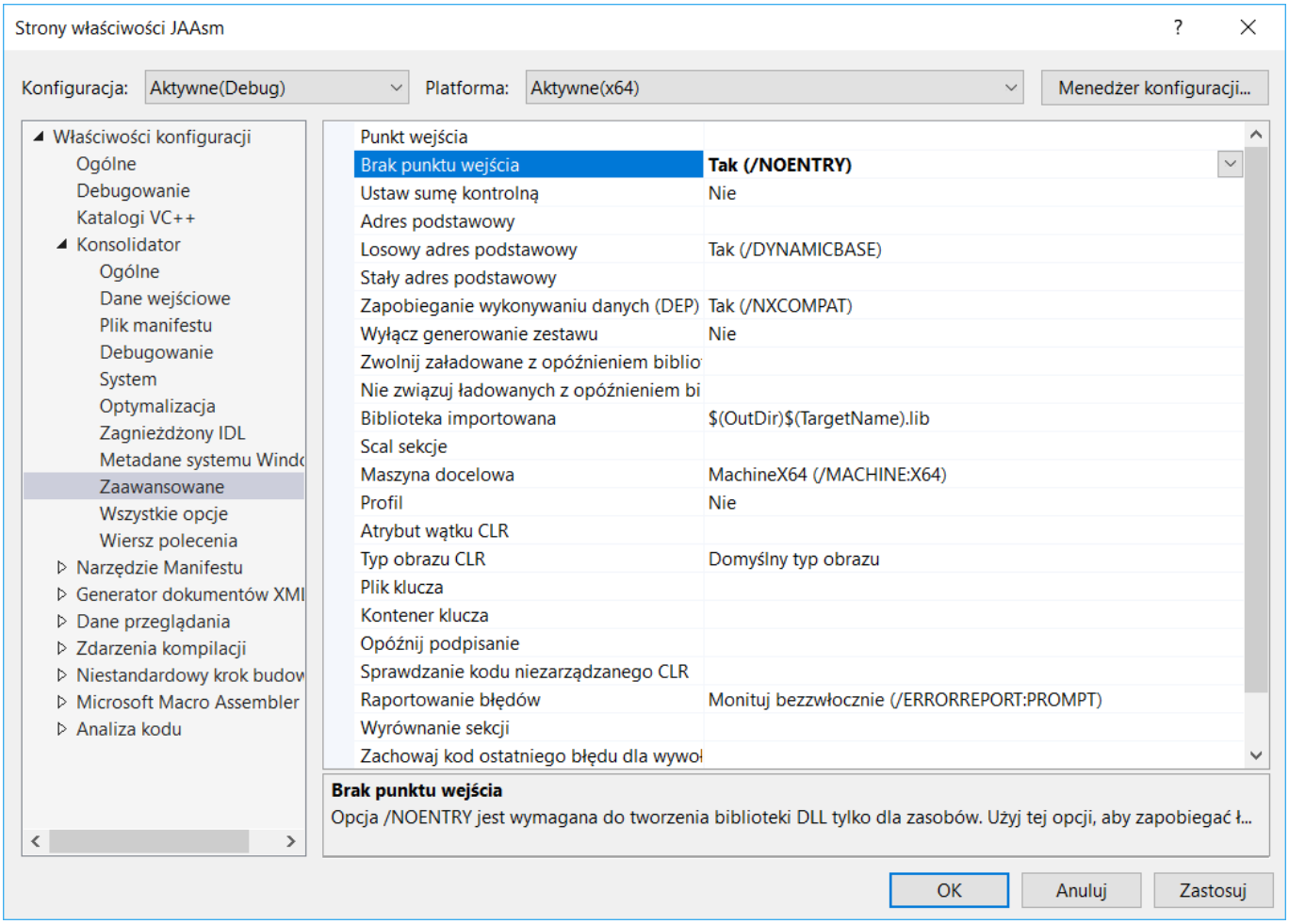


Należy też zmienić bitowość kompilacji projektu z x86 na x64:



13. Należy również zmienić parametr punktu wejścia do biblioteki. W tym celu należy wejść do właściwości biblioteki (PPM, Właściwości), przejść na zakładkę Właściwości konfiguracji, Konsolidator, Zaawansowane

oraz zmienić parametr Brak punktu wejścia z Nie na Tak:



Realizacja kodu projektu

1. W pliku JAAsm.asm zawieramy następujący, PRZYKŁADOWY kod:

```
.code

MyProc1 proc

add RCX, RDX
mov RAX, RCX

ret
MyProc1 endp
end
```

2. W pliku JAAsm.def deklarujemy powstałą procedurę:

```
LIBRARY JAAsm

EXPORTS MyProc1
```

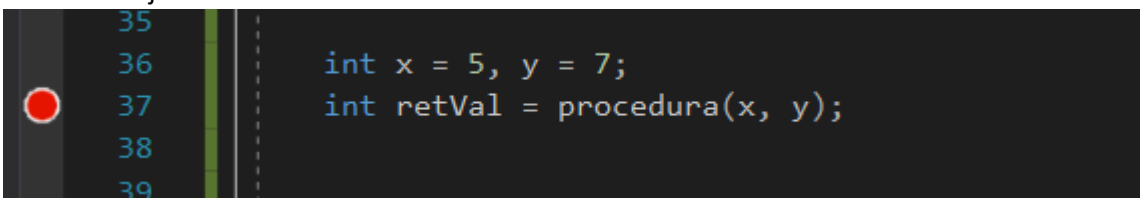
3. Po tych operacjach, powinno się udać poprawnie skompilować bibliotekę
4. Dla projektu w języku wysokiego poziomu, należy zadeklarować typ procedury:
   ```
   typedef int(_fastcall *MyProc1)(int, int);
   ```
5. Następnie utworzyć uchwyt do biblioteki:
   ```
   HINSTANCE dllHandle = NULL;
   ```
6. Oraz załadować samą bibliotekę:
   ```
   dllHandle = LoadLibrary(L"JAAsm.dll");
   ```
7. Następnie należy pobrać procedurę z biblioteki. W tym celu wywołujemy następujący kod:
   ```
   MyProc1 procedura = (MyProc1)GetProcAddress(dllHandle, "MyProc1");
   ```

8. W punkcie 4 zadeklarowaliśmy że nasza procedura będzie komunikowała się z programem wysokiego poziomu za pomocą fastcall, będzie przyjmowała dwa parametry typu int, oraz zwracała int. Należy w tym miejscu zapoznać się z działaniem fastcall'a, szczególnie odnośnie rejestrów w których przekazywane i zwracane są wartości! Możemy teraz zadeklarować sobie zmienne oraz wywołać naszą procedurę:

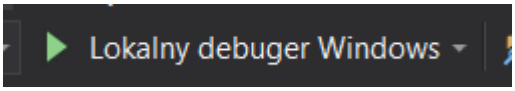
```
int x = 5, y = 7;
int retVal = procedura(x, y);
```

Debugowanie i analiza kodu

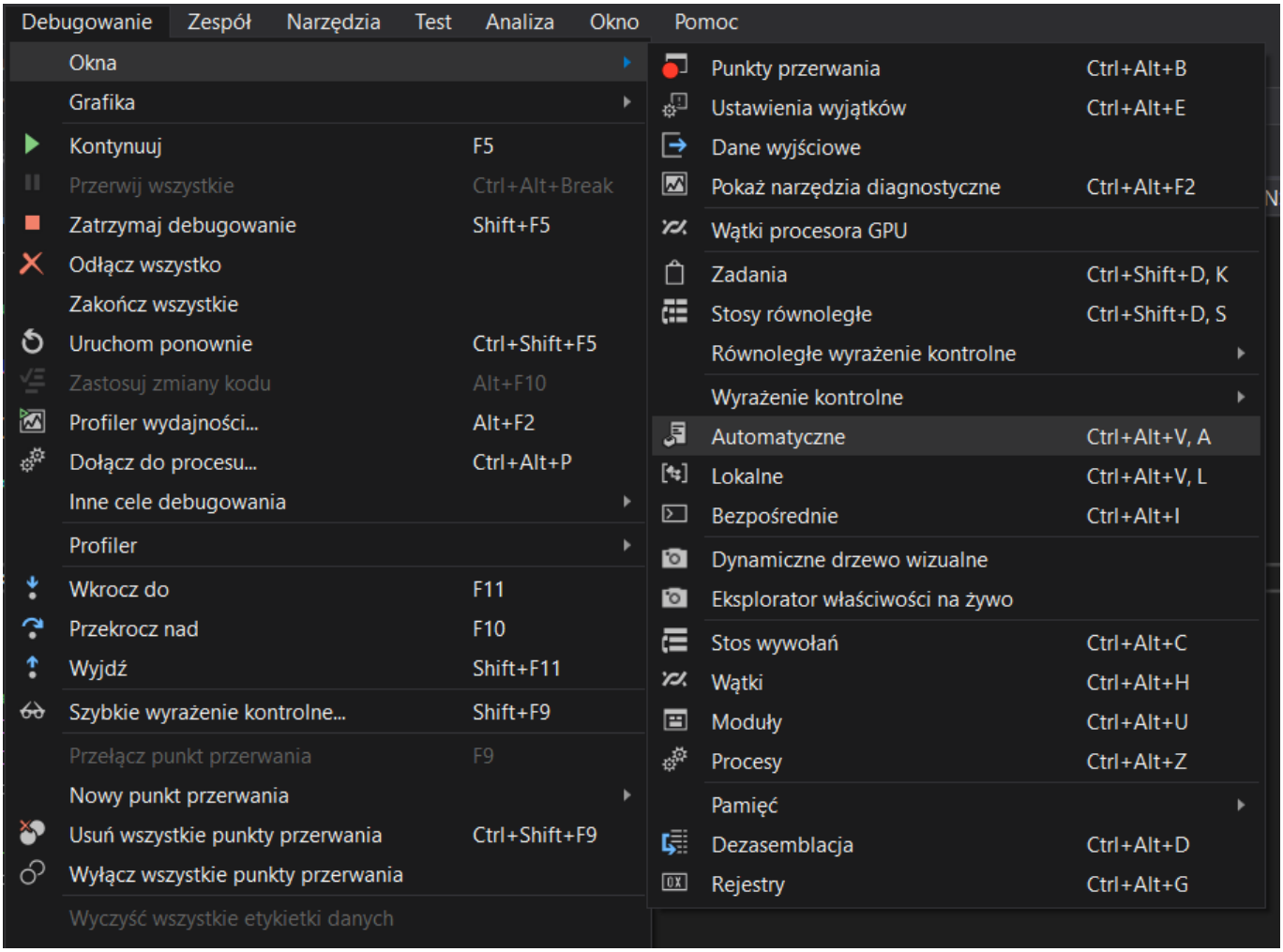
9. Po poprawnym skompilowaniu projektu stawiamy breakpoint na instrukcji realizacji procedury bibliotecznej:



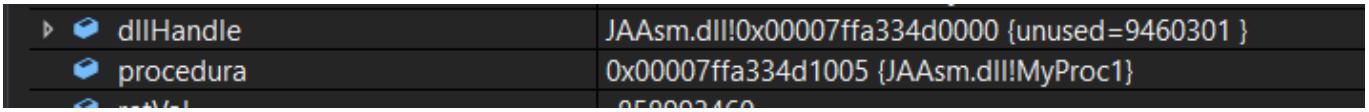
Oraz uruchamiamy debugowanie:



10. Po zatrzymaniu się kompilatora na instrukcji, należy włączyć okna umożliwiające podgląd warunków działania kodu. W tym celu w menu Debugowanie, Okna wybieramy
    a. Automatycznie (podgląd aktualnych zmiennych)
    b. Rejestry (podgląd rejestrów i flag)

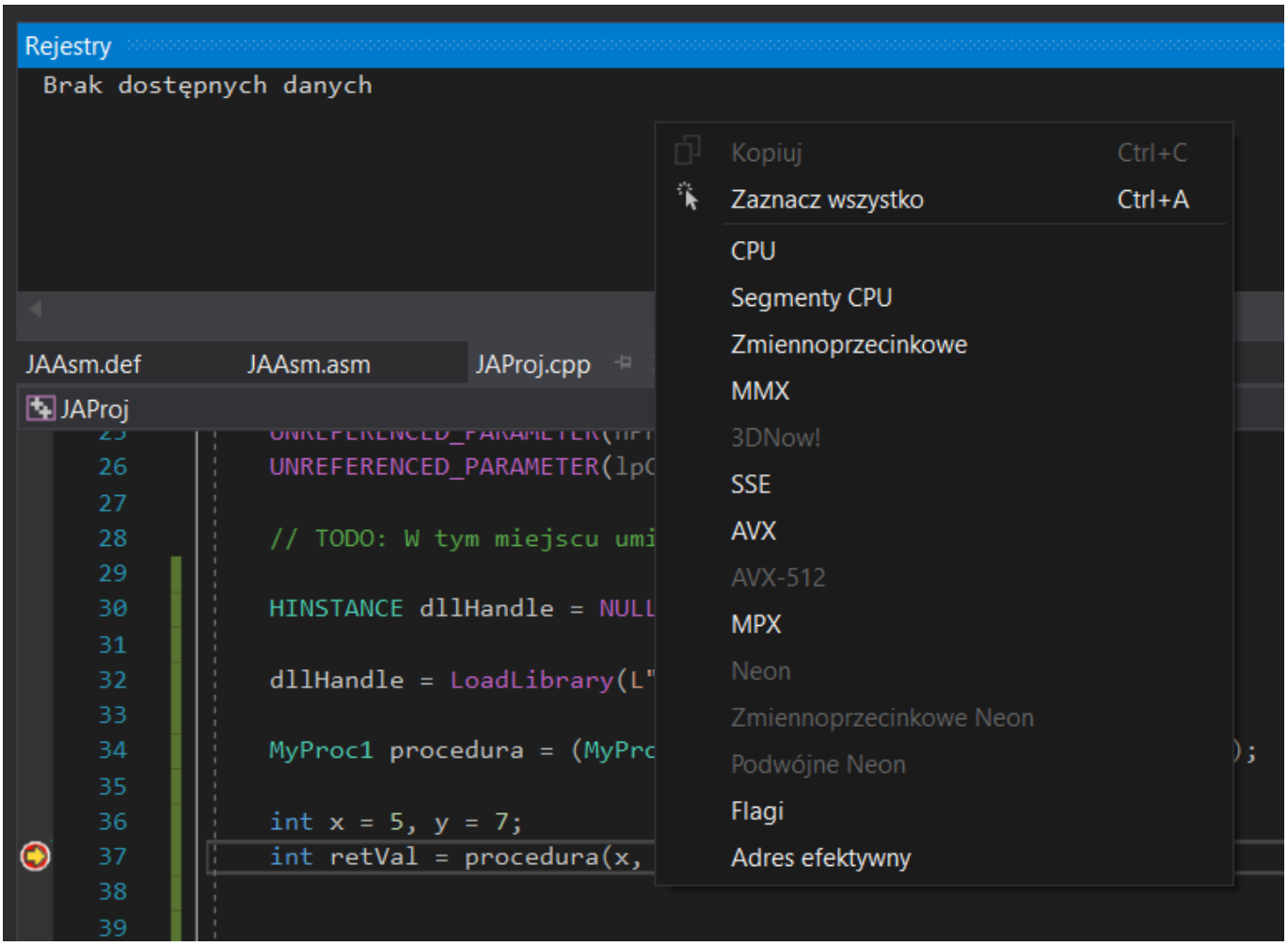


11. Informacje zawarte w podglądzie aktualnych zmiennych mówią nam o ich wartości oraz np. o poprawnej inicjalizacji biblioteki:

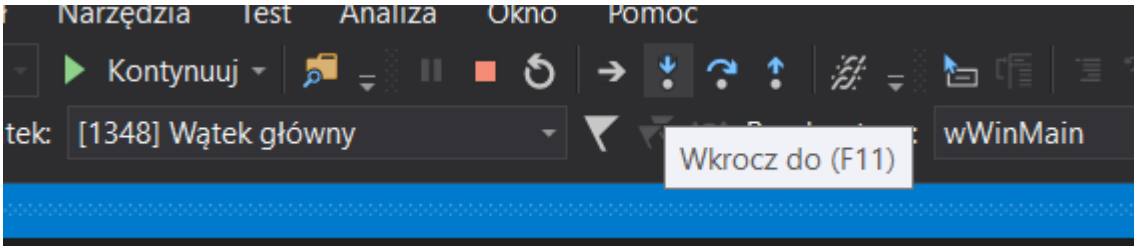


Gdzie dllHandle przechowuje uchwyt do biblioteki, a procedura jest wskaźnikiem na adres w bibliotece w której zaczyna się nasza procedura (Jak widać, obie załadowane są poprawnie)

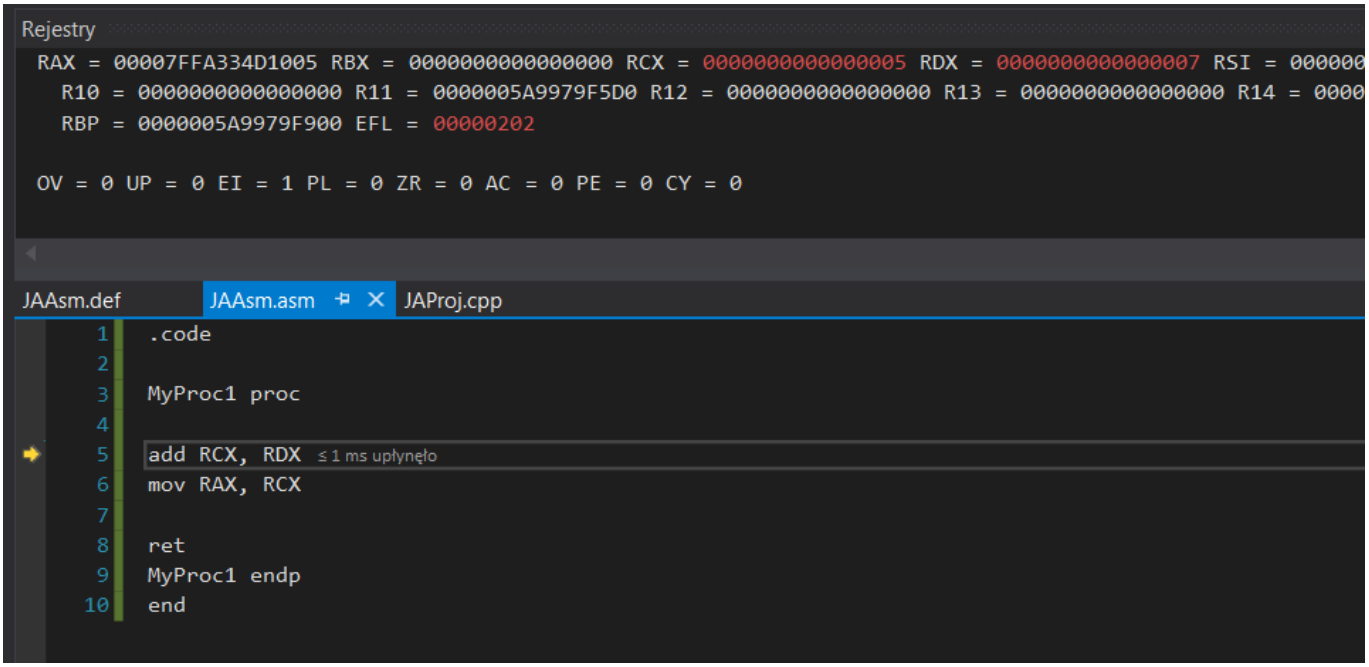
12. W polu Rejestry należy uwidocznić Rejestry procesora, oraz Flagi. W tym celu należy kliknąć PPM w dowolnym miejscu okna Rejestry i zaznaczyć żądane opcje:



13. Analiza poszczególnych instrukcji jest możliwa po przejściu do biblioteki z wykorzystaniem przycisku „Wskocz do”



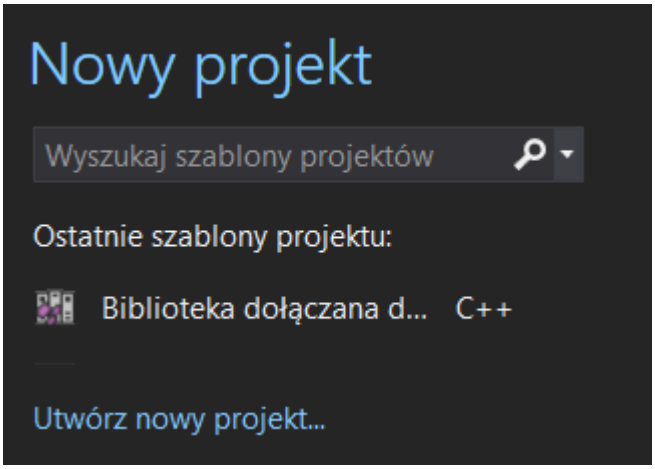
Widać teraz przykładowo przekazanie wartości zmiennych do rejestrów RCX oraz RDX:



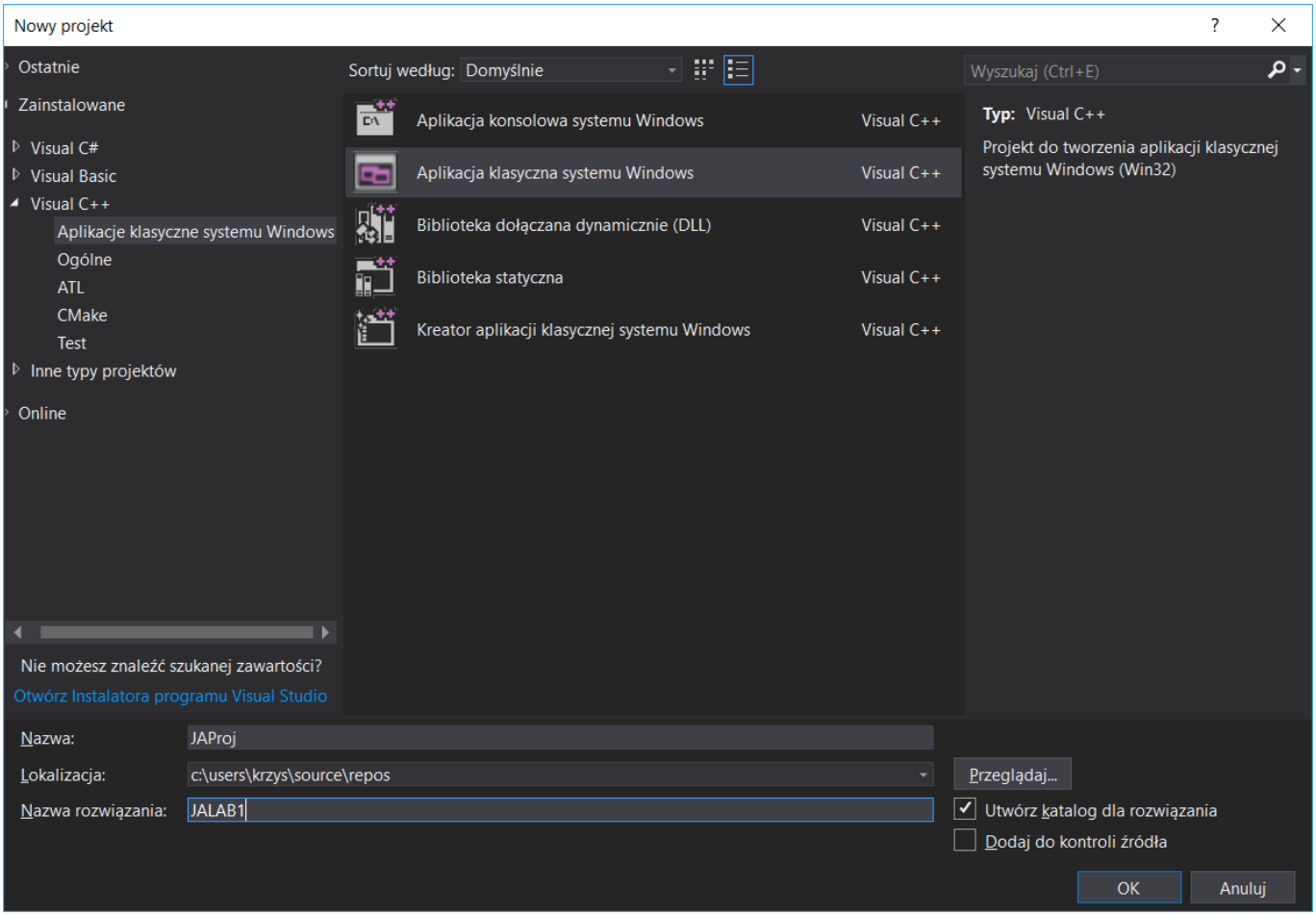
Aplikacja x32

Tworzenie projektów

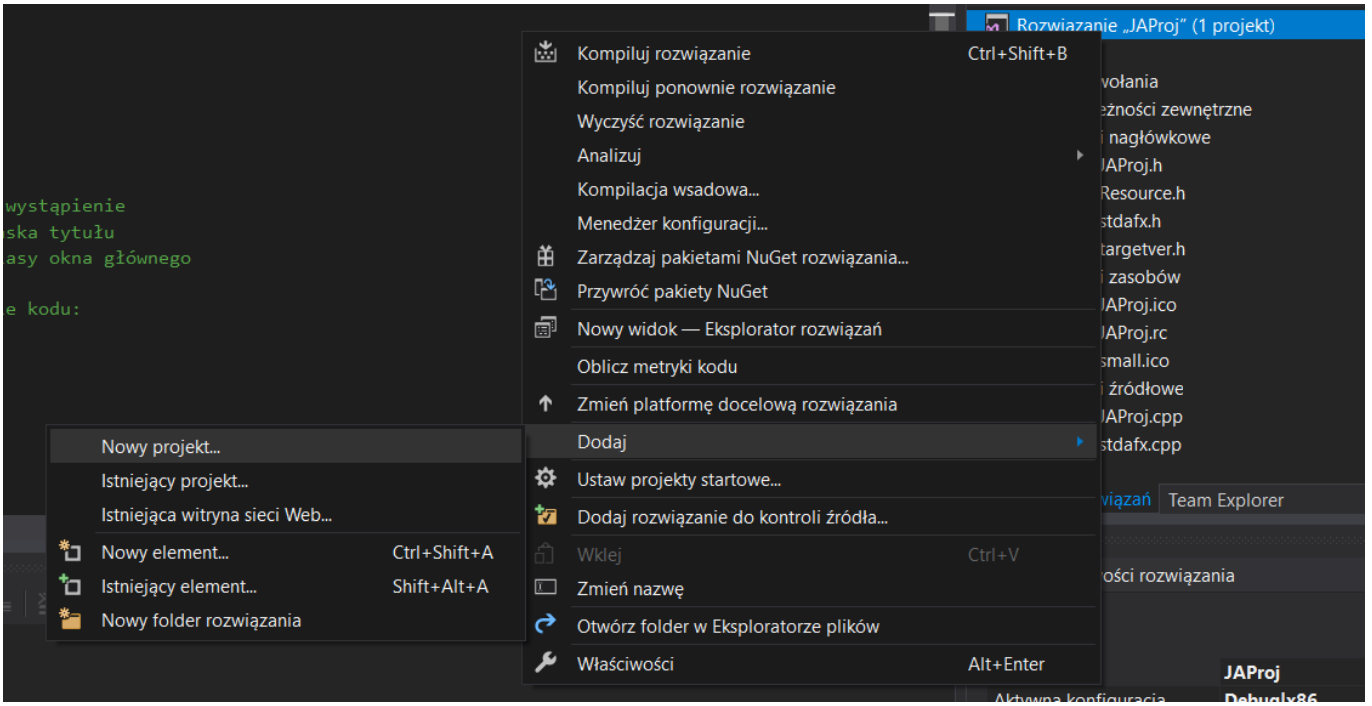
1. Tworzymy nowy projekt



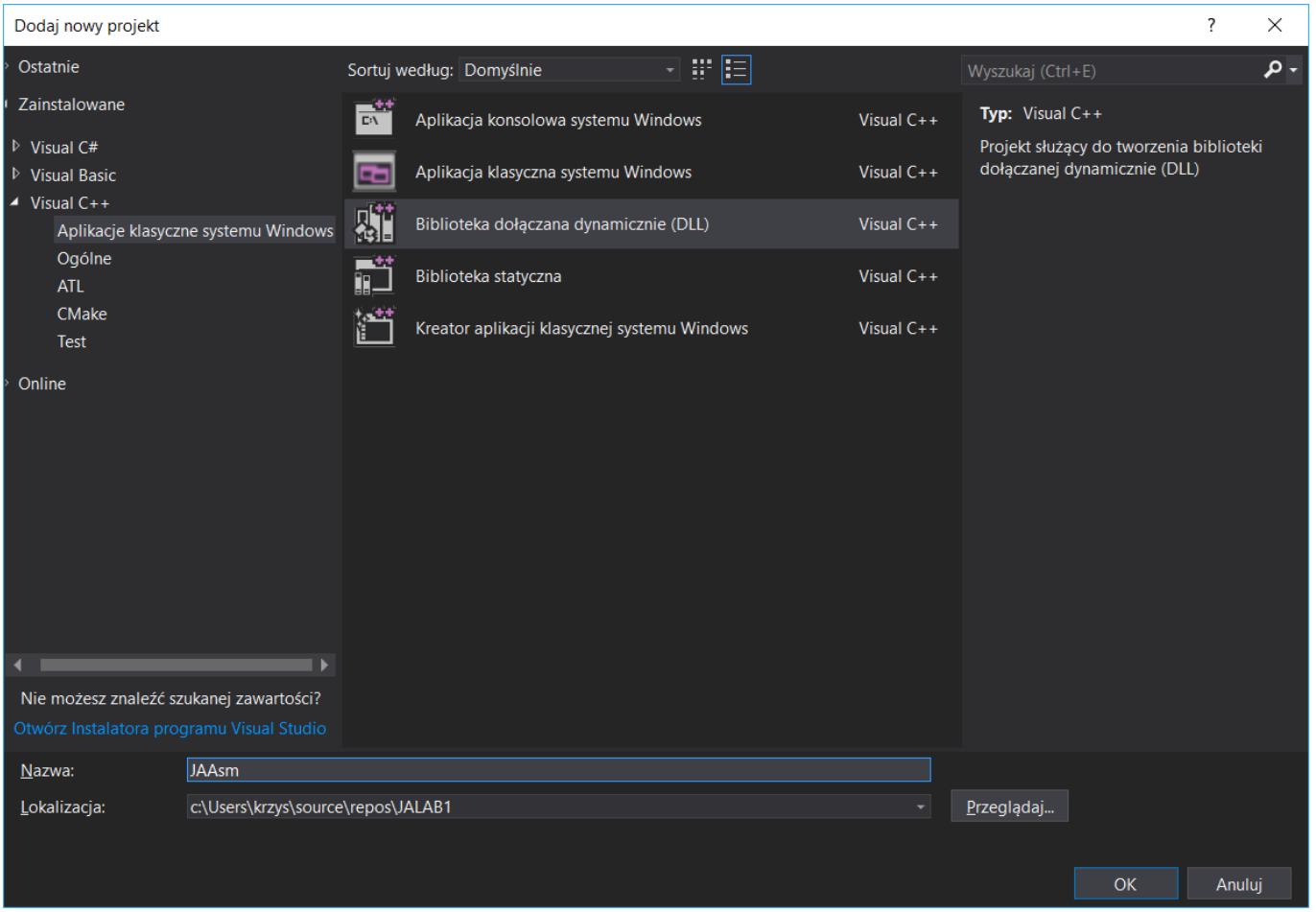
2. Konfigurujemy projekt jako Aplikacja klasyczna systemu Windows. Nazwa to JAProj a Nazwa rozwiązania to JALAB1



3. Dodajemy nowy projekt (PPM na projekt, następnie Dodaj oraz Nowy projekt)

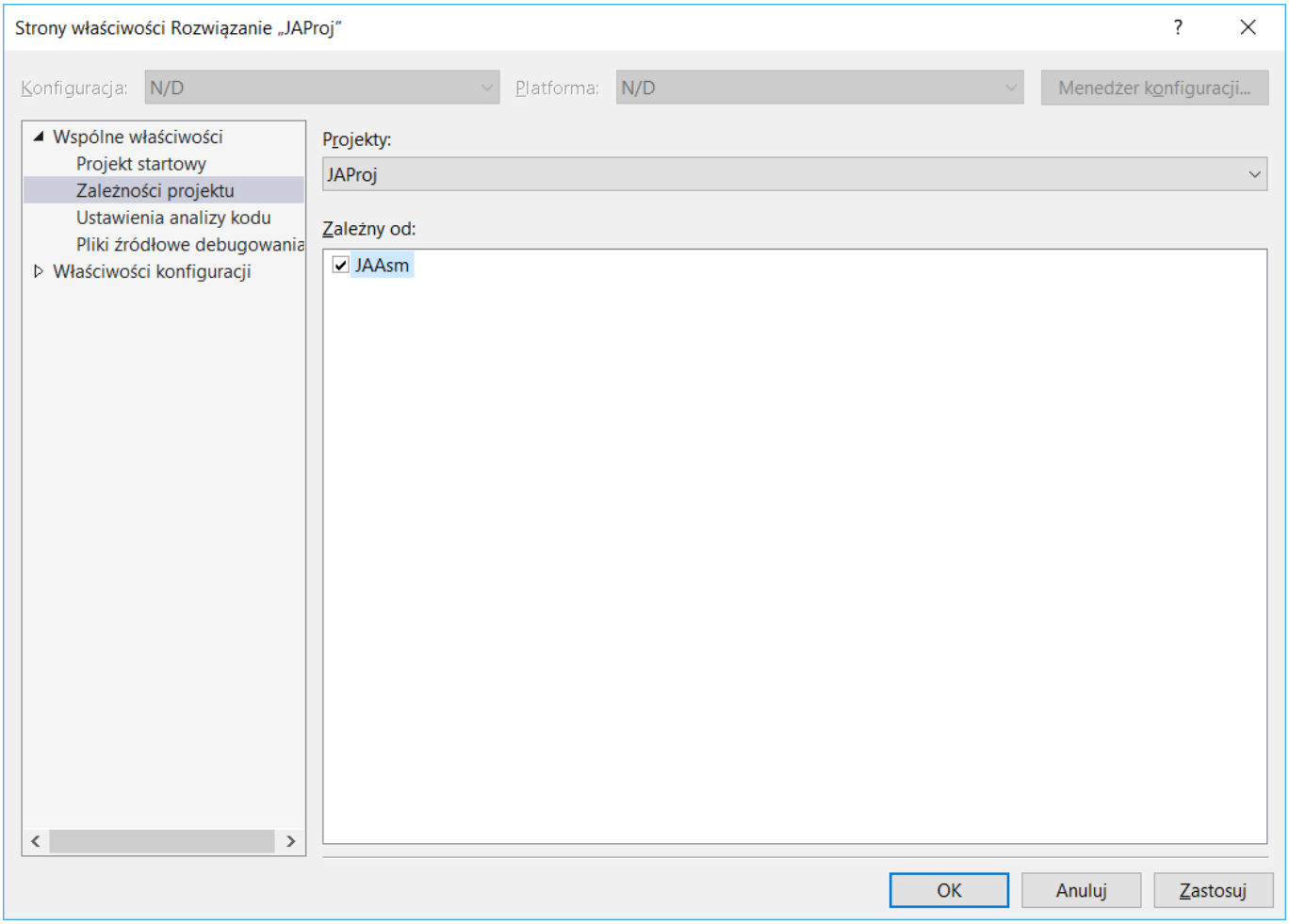


4. Wybieramy bibliotekę dynamiczną (DLL) oraz jako nazwę podajemy JAAsm:

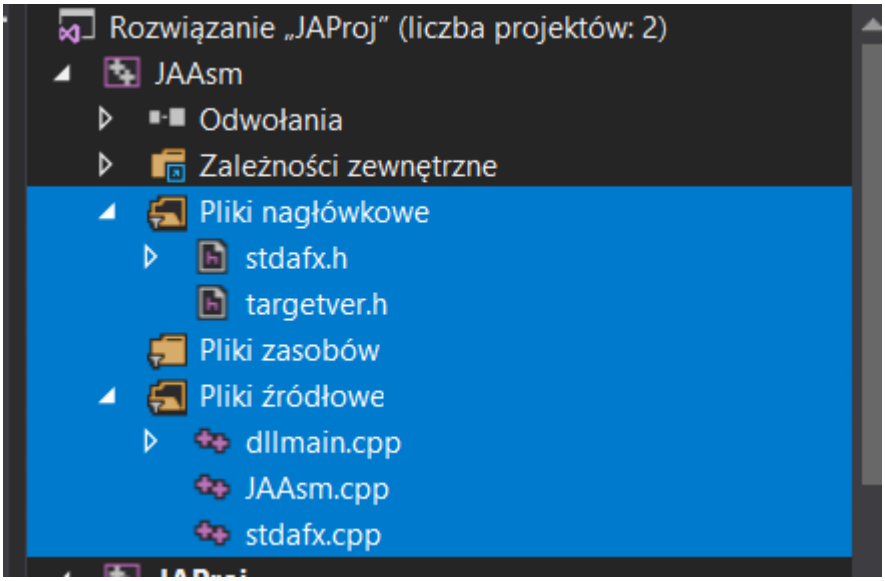


5. Po dołączeniu biblioteki JAAsm, należy ustawić jej zależność kompilacji względem projektu w języku wysokiego poziomu. W tym celu klikamy PPM na solucję i wybieramy właściwości. Następnie przechodzimy na zakładkę wspólne właściwości, zależności projektu i zaznaczamy pole wyboru Zależny od

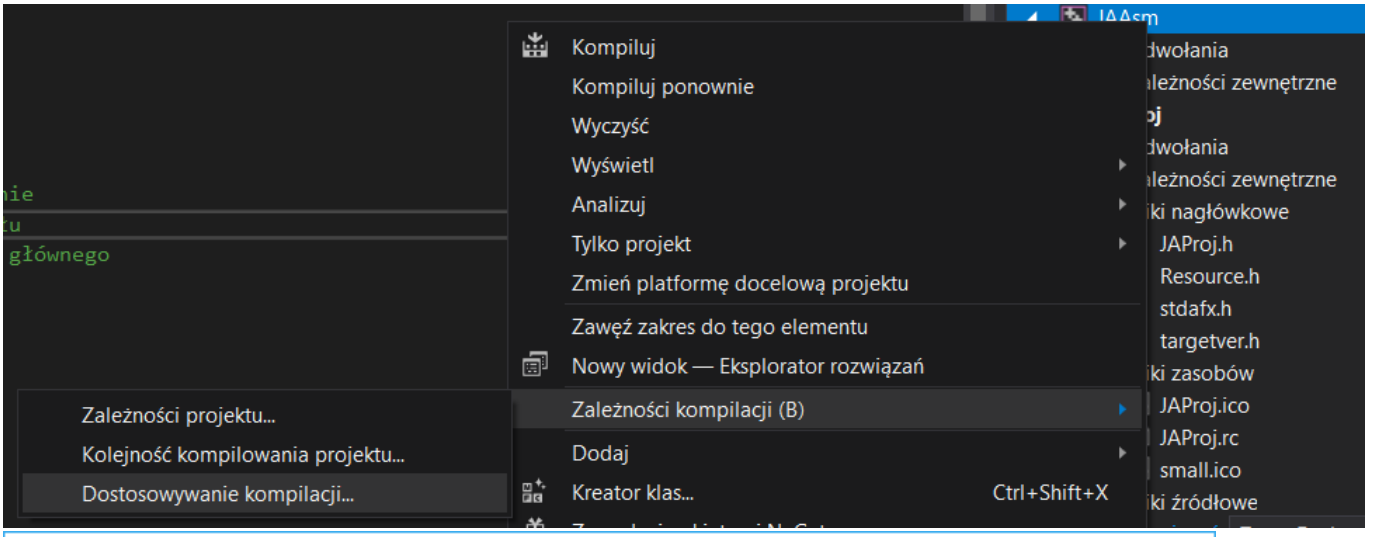
przy JAAsm:

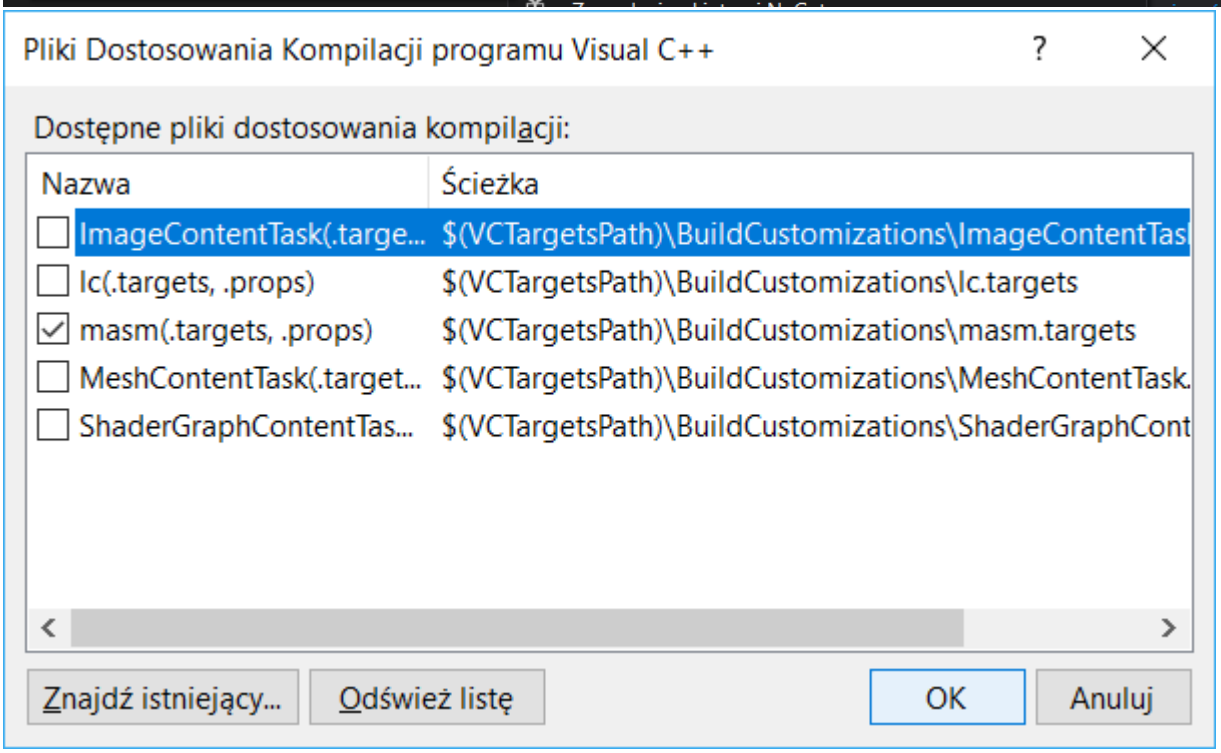


6. Można skasować powstałe w bibliotece pliki:

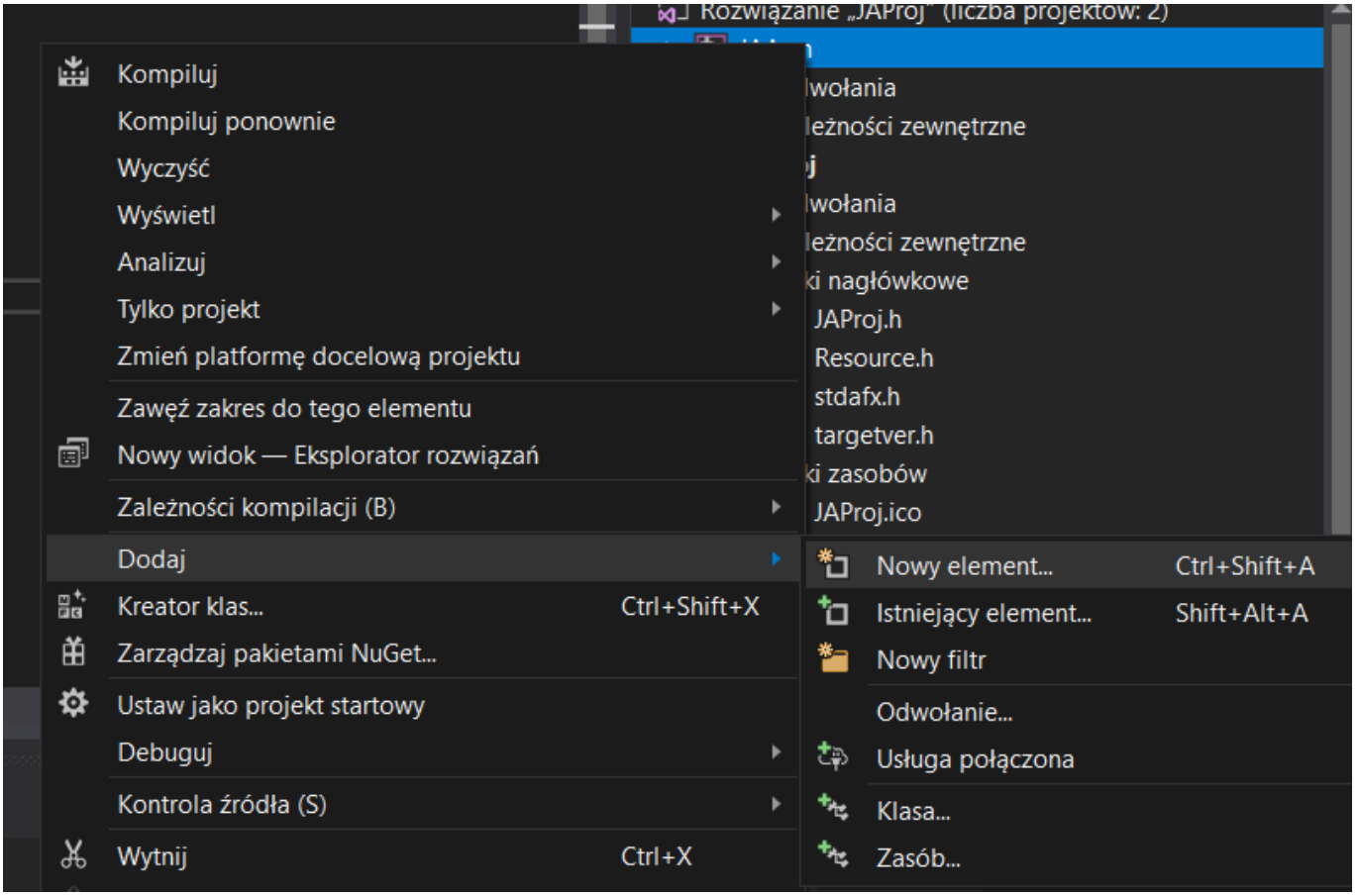


7. W zależnościach kompilacji biblioteki zmieniamy dostosowanie kompilacji do masm:

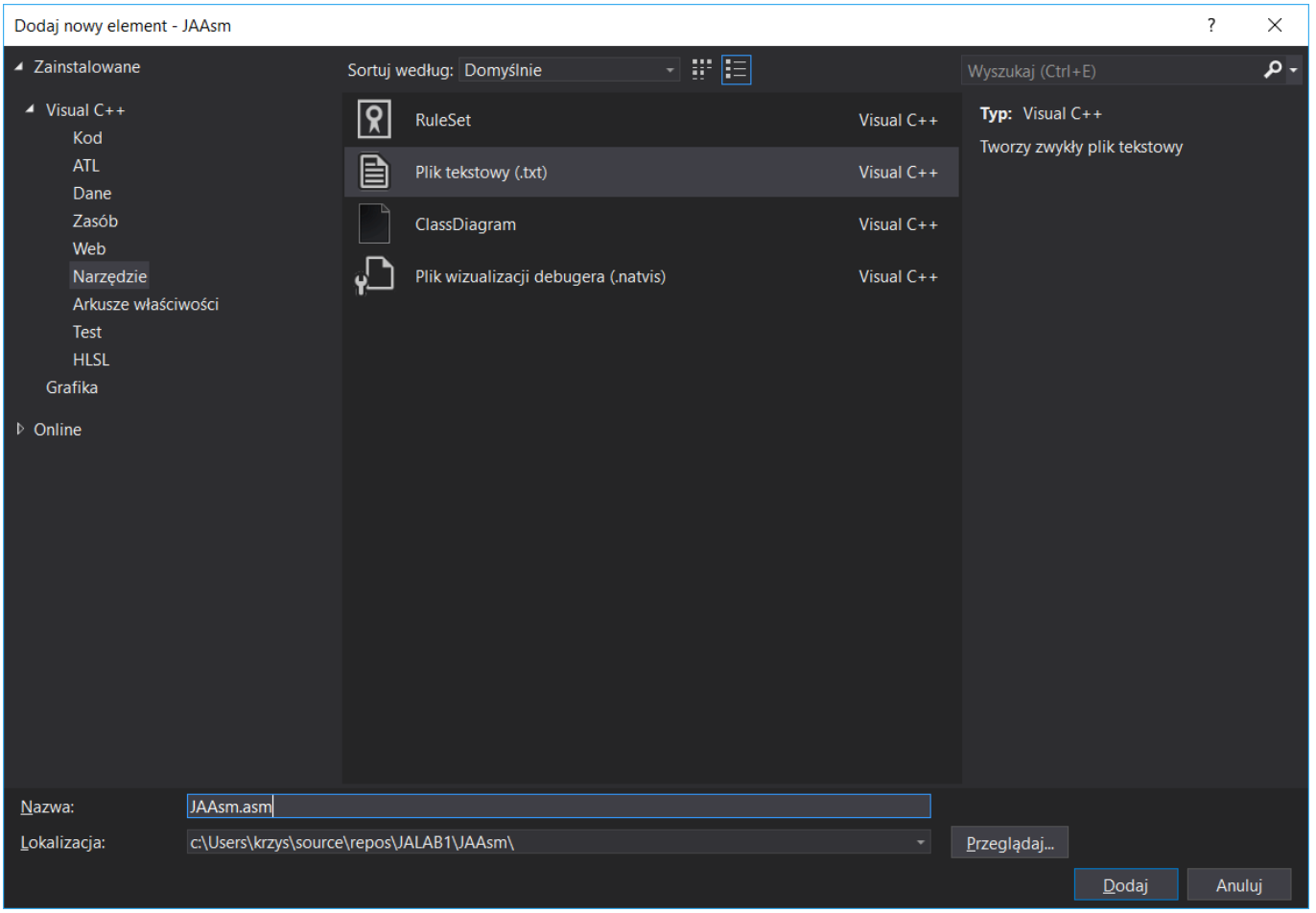




8. Dodajemy do biblioteki nowy element w postaci pliku asm. W tym celu wybieramy:

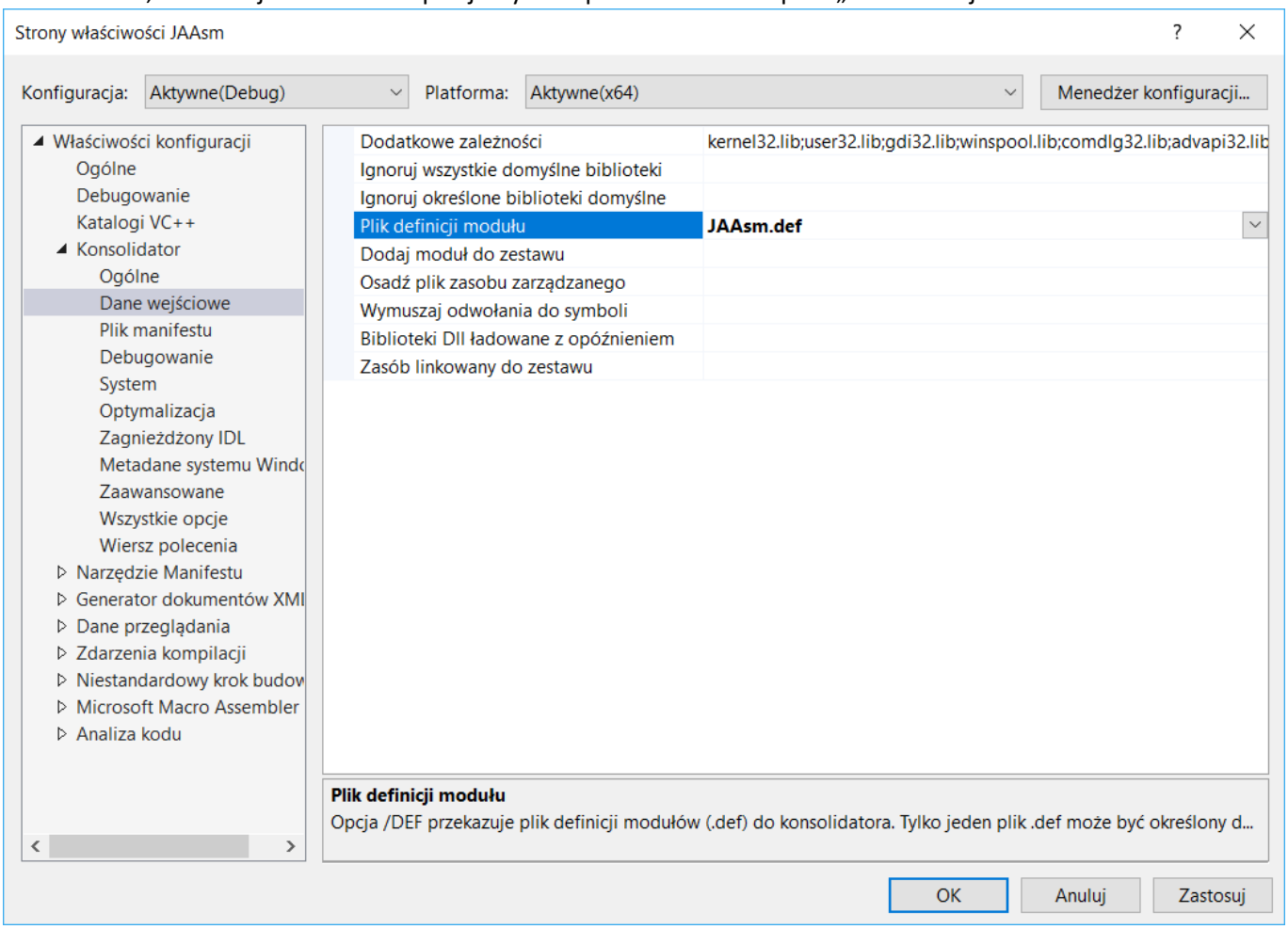


9. Następnie należy wybrać plik tekstowy (Visual C++, Narzędzie) oraz zmienić jego nazwę na JAAsm.asm:



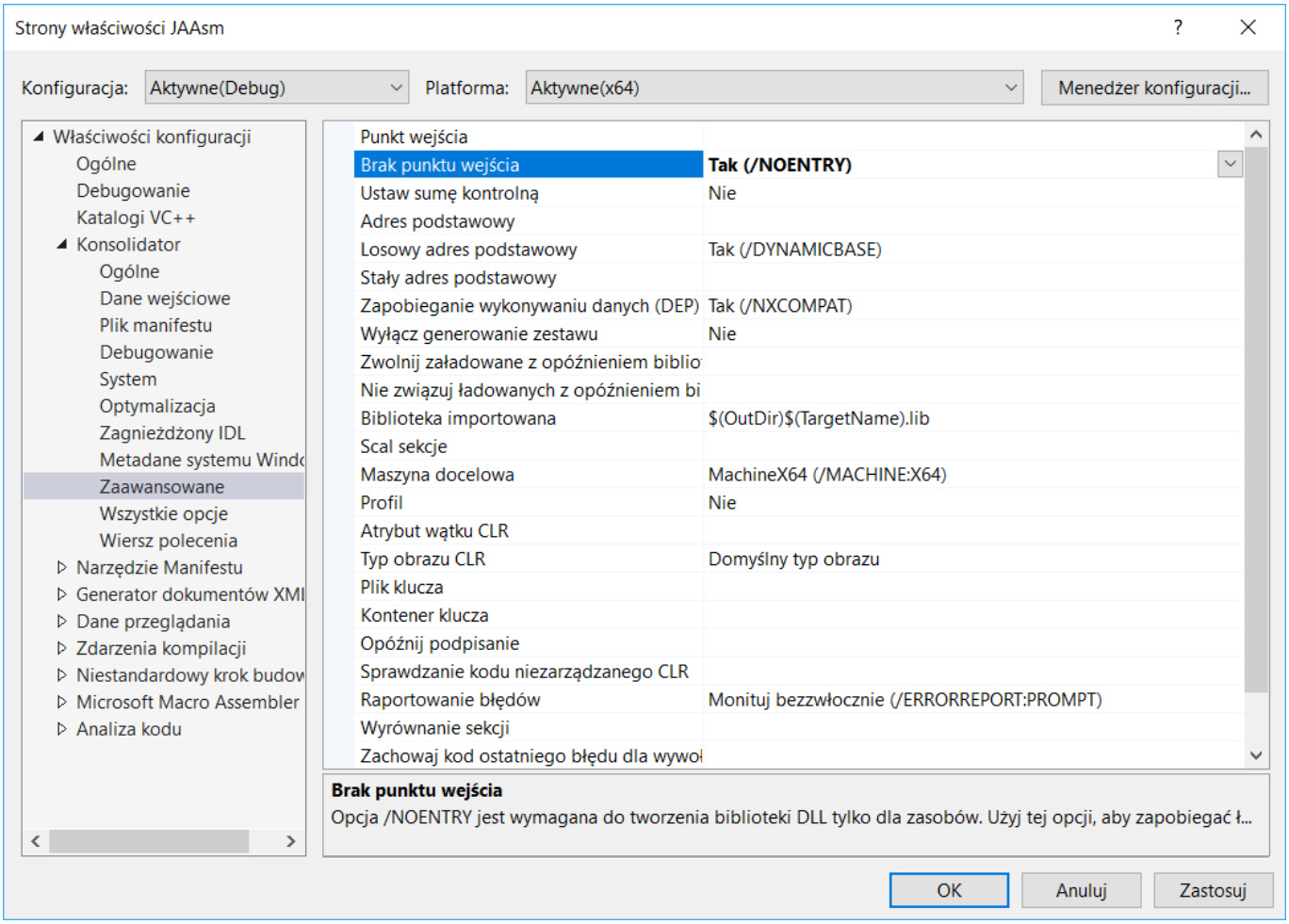
10. Analogicznie dodajemy plik JAAsm.def

11. Po dodaniu pliku definicji modułu, należy podpiąć go pod linkera. W tym celu przechodzimy do właściwości biblioteki (PPM, Właściwości), następnie przechodzimy do zakładki Właściwości konfiguracji, Konsolidator, Dane wejściowe oraz wpisujemy nasz plik JAAsm.def w polu „Plik definicji modułu”:



12. Należy również zmienić parametr punktu wejścia do biblioteki. W tym celu należy wejść do właściwości biblioteki (PPM, Właściwości), przejść na zakładkę Właściwości konfiguracji, Konsolidator, Zaawansowane

oraz zmienić parametr Brak punktu wejścia z Nie na Tak:



Realizacja kodu projektu

1. W pliku JAAsm.asm dodajemy następujący kod:

```
.386
.MODEL FLAT, STDCALL

.code

MyProc1 proc x:DWORD, y:DWORD

mov eax, x

ret
MyProc1 endp
end
```

2. W pliku JAAsm.def dodajemy następujący kod:

```
LIBRARY "JAAsm"
EXPORTS MyProc1
```

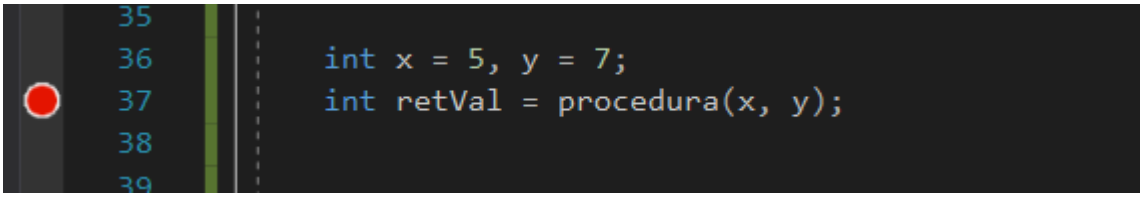
3. Dla projektu w języku wysokiego poziomu, należy zadeklarować typ procedury
   `typedef int(_stdcall *MyProc1)(int, int);`
4. Następnie utworzyć uchwyt do biblioteki:
   `HINSTANCE dllHandle = NULL;`
5. Oraz załadować samą bibliotekę:
   `dllHandle = LoadLibrary(L"JAAsm.dll");`
6. Następnie należy pobrać procedurę z biblioteki. W tym celu wywołujemy następujący kod:
   `MyProc1 procedura = (MyProc1)GetProcAddress(dllHandle, "MyProc1");`

7. Oraz dokonać wywołania funkcji z biblioteki:

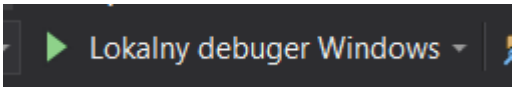
```
int x = 5, y = 7;
int retVal = procedura(x, y);
```

Debugowanie i analiza kodu

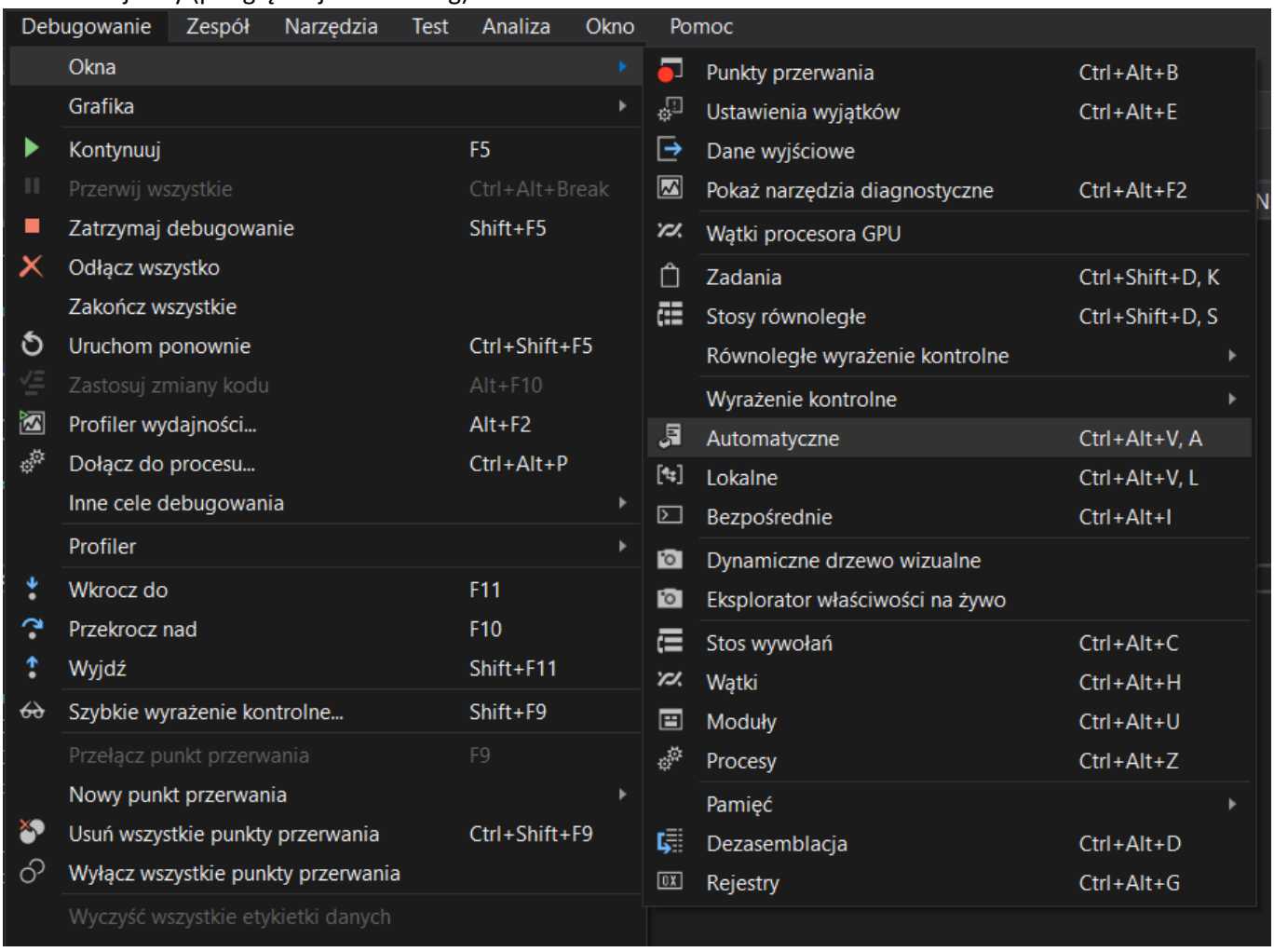
1. Po poprawnym skompilowaniu projektu stawiamy breakpoint na instrukcji realizacji procedury bibliotecznej:



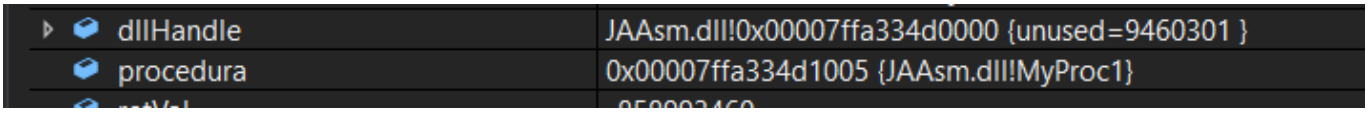
Oraz uruchamiamy debugowanie:



2. Po zatrzymaniu się kompilatora na instrukcji, należy włączyć okna umożliwiające podgląd warunków działania kodu. W tym celu w menu Debugowanie, Okna wybieramy
   a. Automatycznie (podgląd aktualnych zmiennych)
   b. Rejestry (podgląd rejestrów i flag)

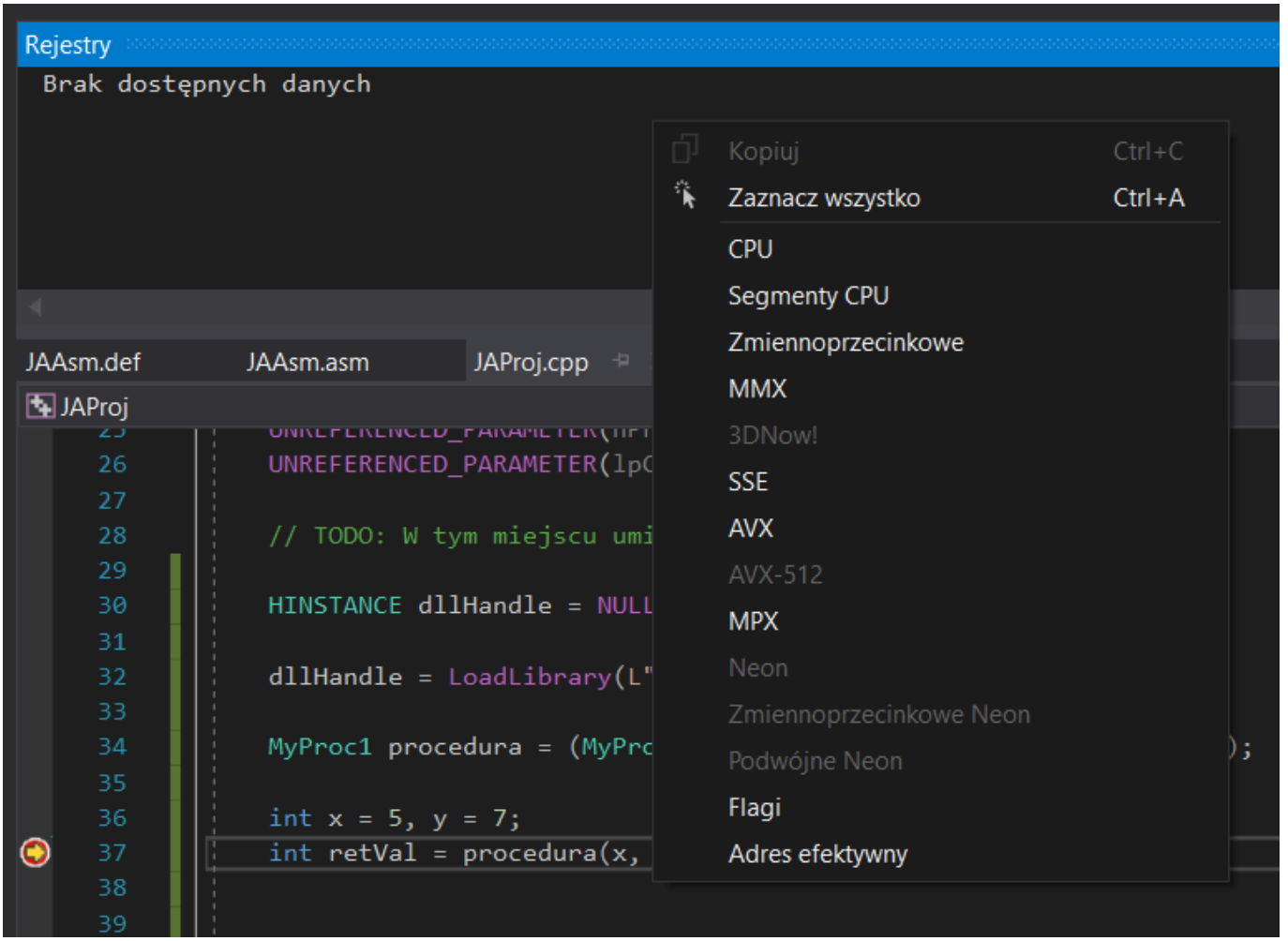


3. Informacje zawarte w podglądzie aktualnych zmiennych mówią nam o ich wartości oraz np. o poprawnej inicjalizacji biblioteki:

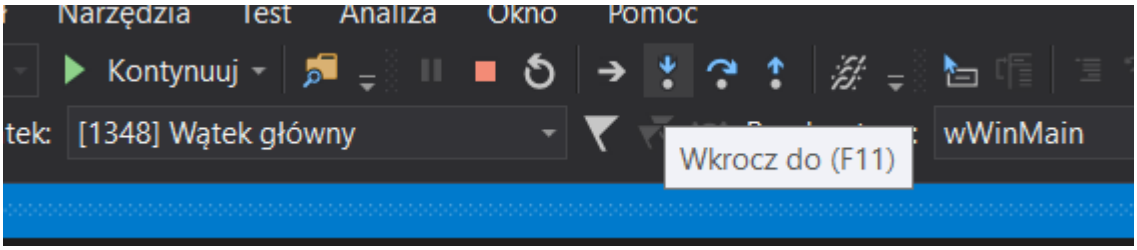


Gdzie dllHandle przechowuje uchwyt do biblioteki, a procedura jest wskaźnikiem na adres w bibliotece w której zaczyna się nasza procedura (Jak widać, obie załadowane są poprawnie)

4. W polu Rejestry należy uwidocznić Rejestry procesora, oraz Flagi. W tym celu należy kliknąć PPM w dowolnym miejscu okna Rejestry i zaznaczyć żądane opcje:



5. Analiza poszczególnych instrukcji jest możliwa po przejściu do biblioteki z wykorzystaniem przycisku „Wskocz do”



Widać teraz przykładowo wynik mnożenia rejestrów eax oraz ecx: