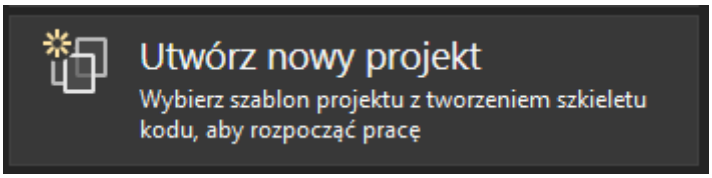
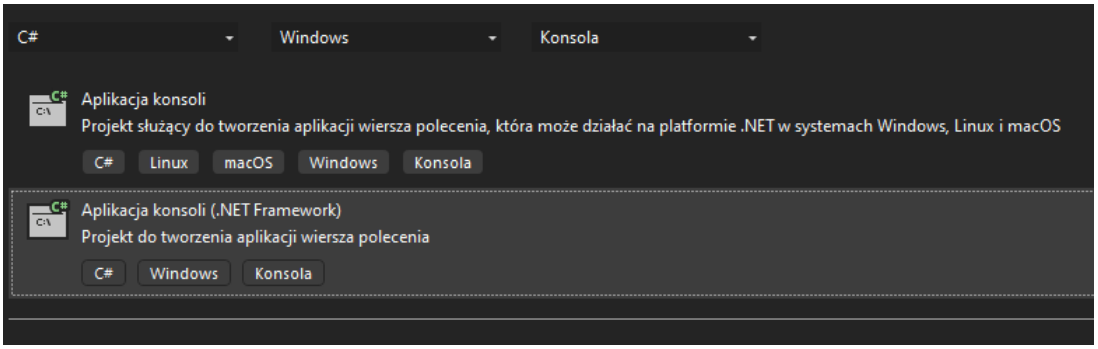
Instrukcja zakładania projektu x64 asm + C# w Visual Studio 2022

Krzysztof Hanzel & Ernest Antolak

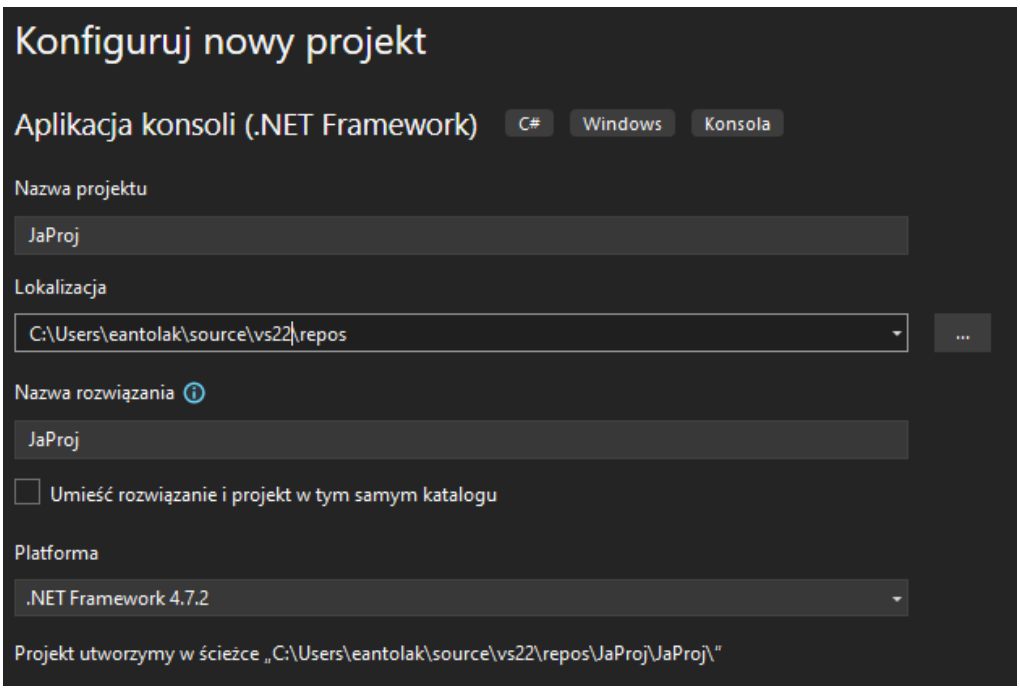
1. Zakładamy nowy projekt korzystając z przycisku „Utwórz nowy projekt”;



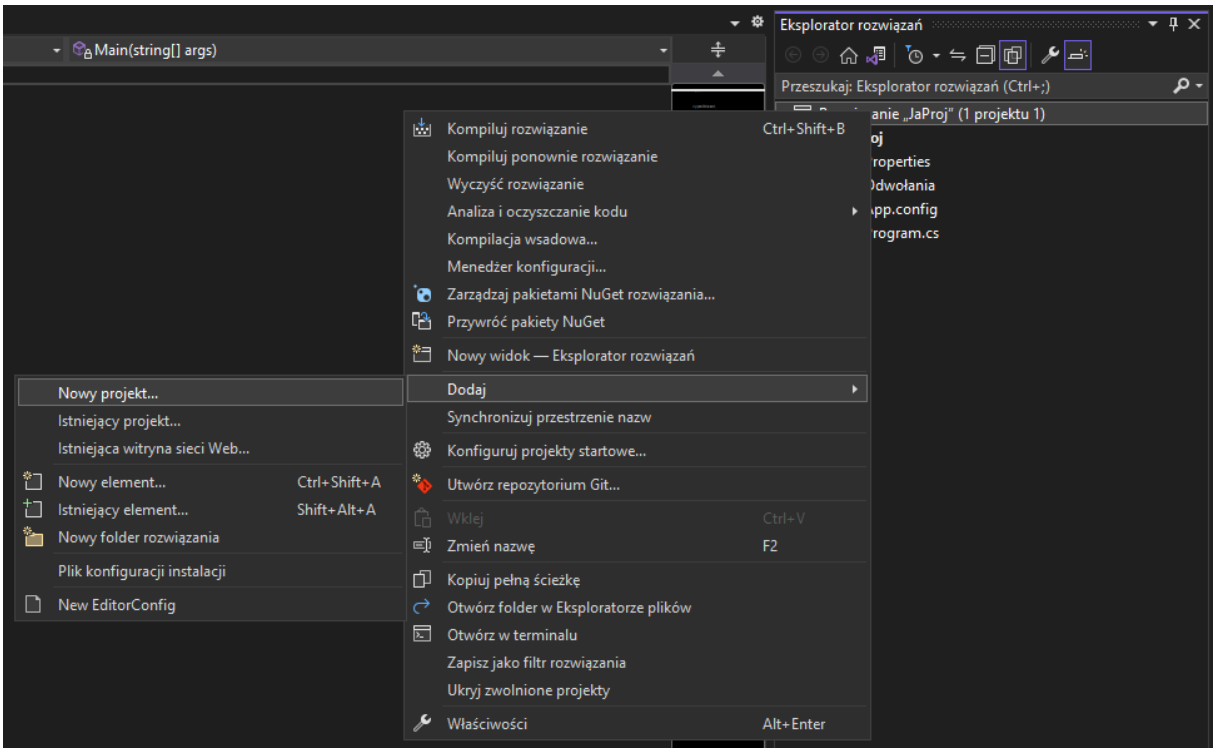
2. Z dostępnej listy aplikacji wybieramy jako typ „Aplikacja konsoli (.NET Core) i klikamy Dalej. (Jeżeli nie ma takiej pozycji, należy doinstalować odpowiednie składniki do VS2022, opcją Zainstaluj więcej narzędzi i funkcji)



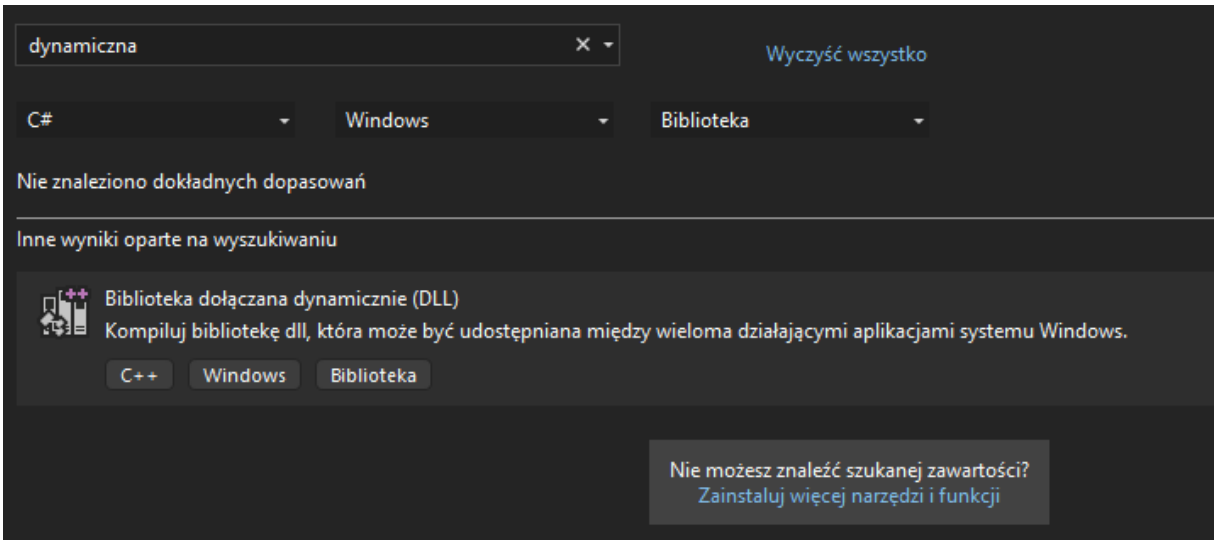
3. W konfiguracji projektu wybieramy nazwę, podajemy ścieżkę i finalizujemy tworzenie projektu

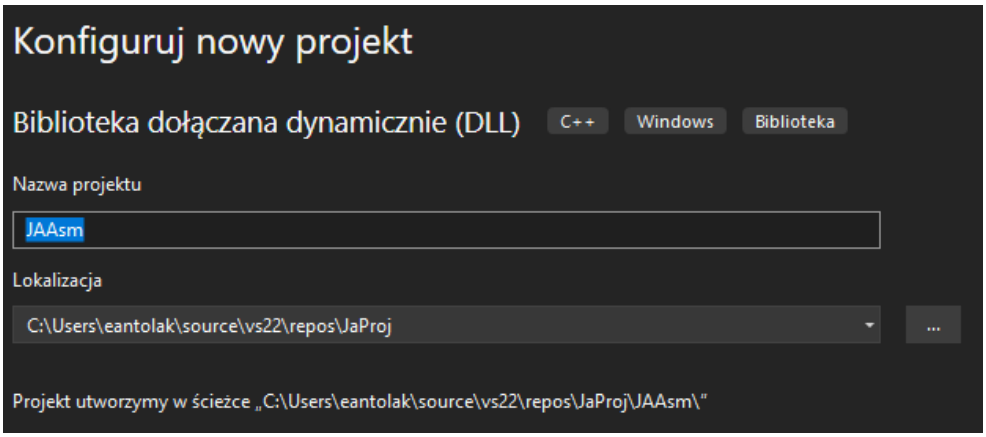


4. Kolejnym etapem jest dodanie kolejnego projektu do naszej solucji. W tym celu klikamy Prawym Przyciskiem Myszy (PPM) na naszą solucję (Rozwiązanie) i wybieramy „Dodaj” a następnie „Nowy projekt…”

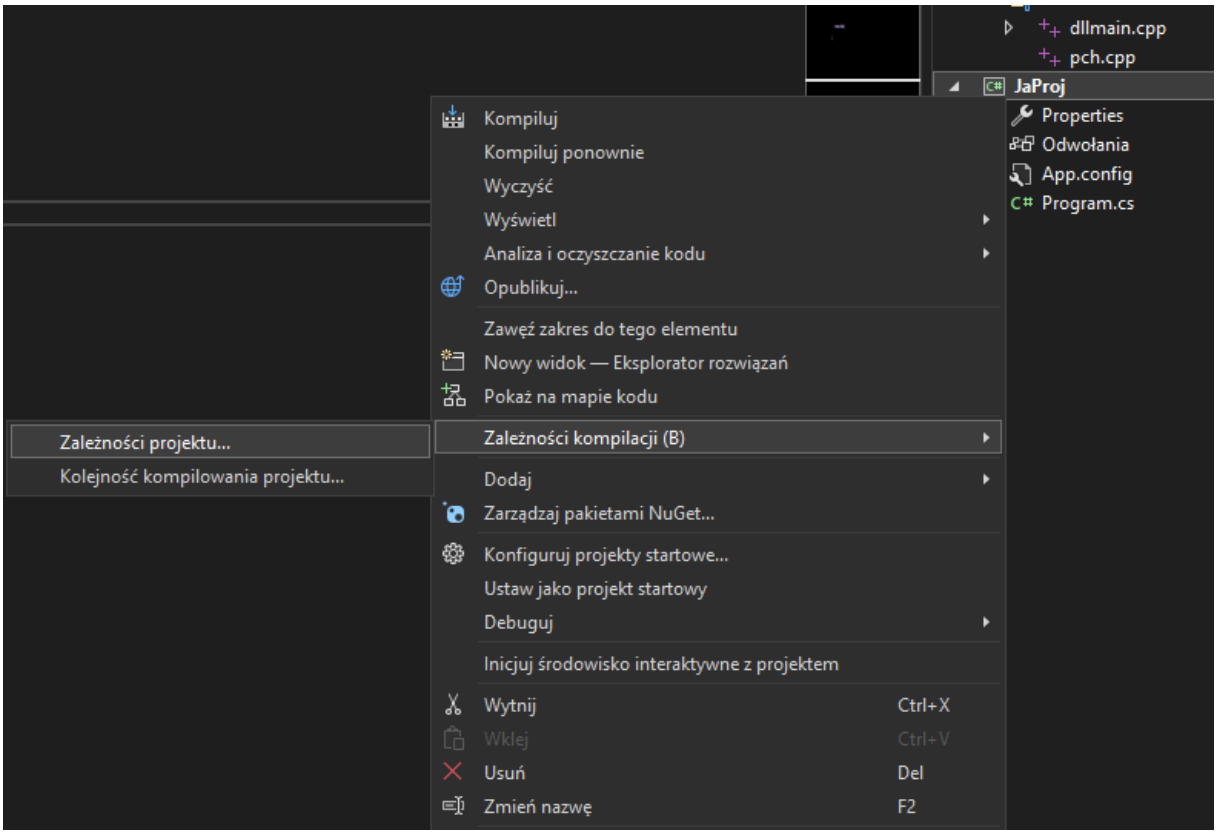


5. Z listy dostępnych projektów wyszukujemy „Biblioteka dołączana dynamicznie (DLL)” i analogicznie jak w poprzednim kroku podajemy nazwę projektu (Jeżeli nie ma takiej pozycji, należy doinstalować odpowiednie składniki do VS2022, opcją Zainstaluj więcej narzędzi i funkcji).

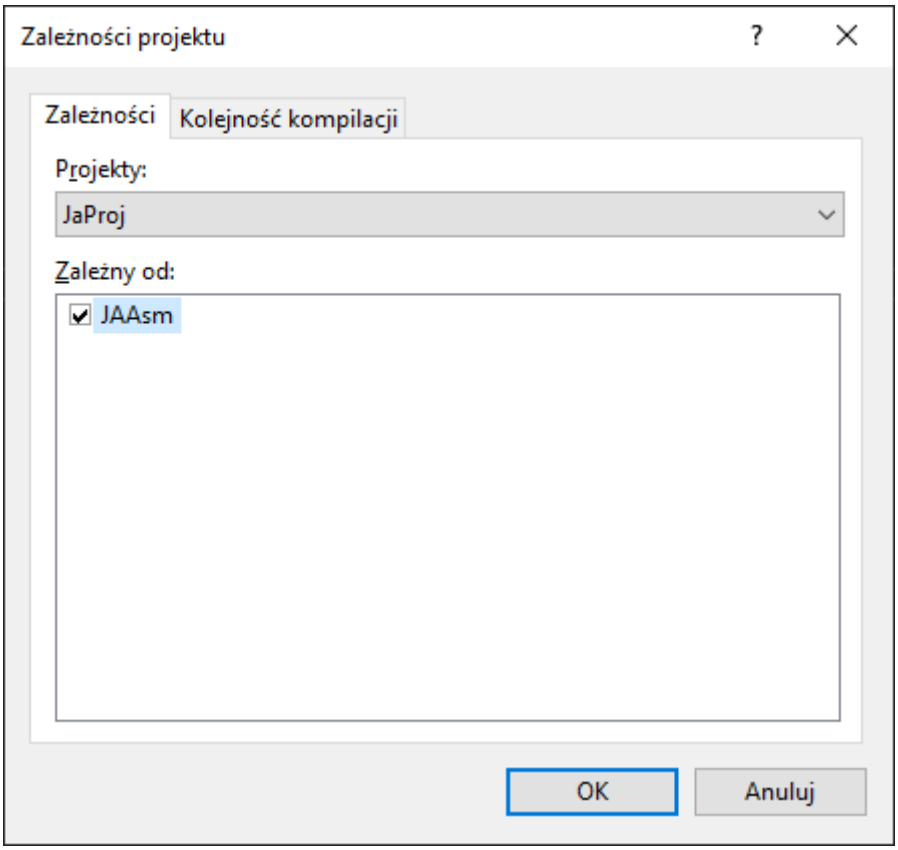




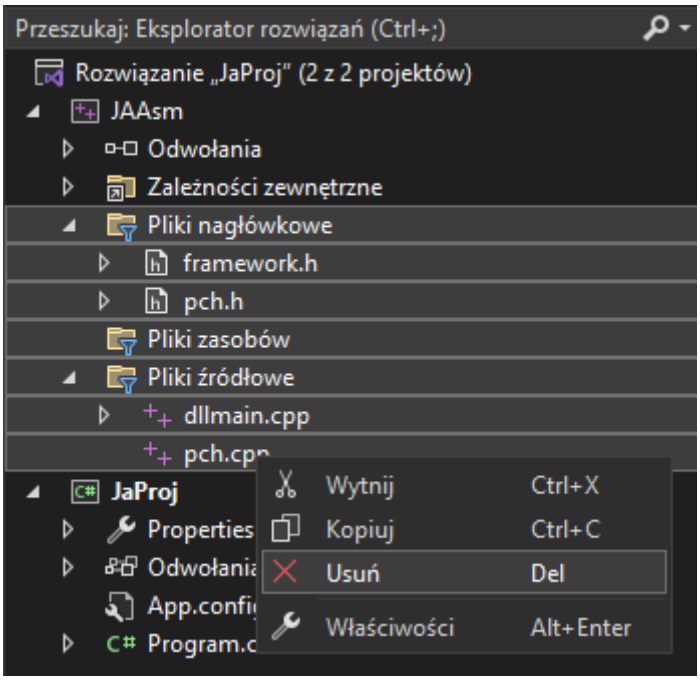
6. Następnym krokiem jest ustawienie zależności projektu w taki sposób, aby kompilacja naszej aplikacji wymuszała automatycznie kompilację biblioteki. W tym celu należy kliknąć PPM na projekt konsolowy a następnie wybrać menu „Zależności kompilacji” a następnie „Zależności projektu…”



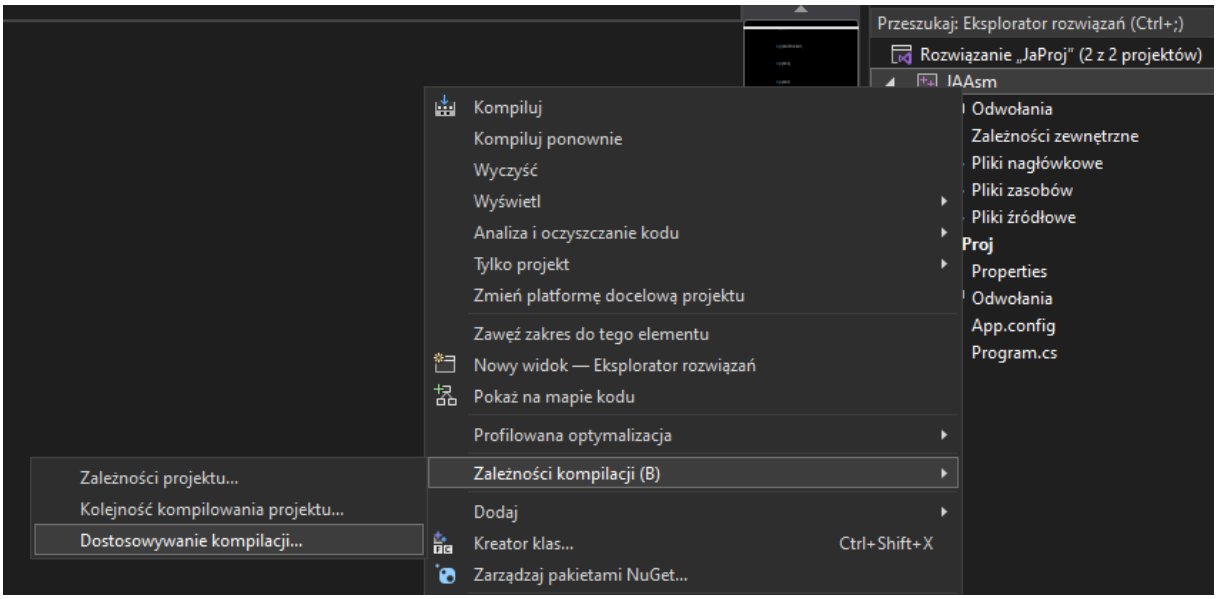
7. W nowo otwartym oknie zaznaczamy naszą bibliotekę i klikamy „OK”



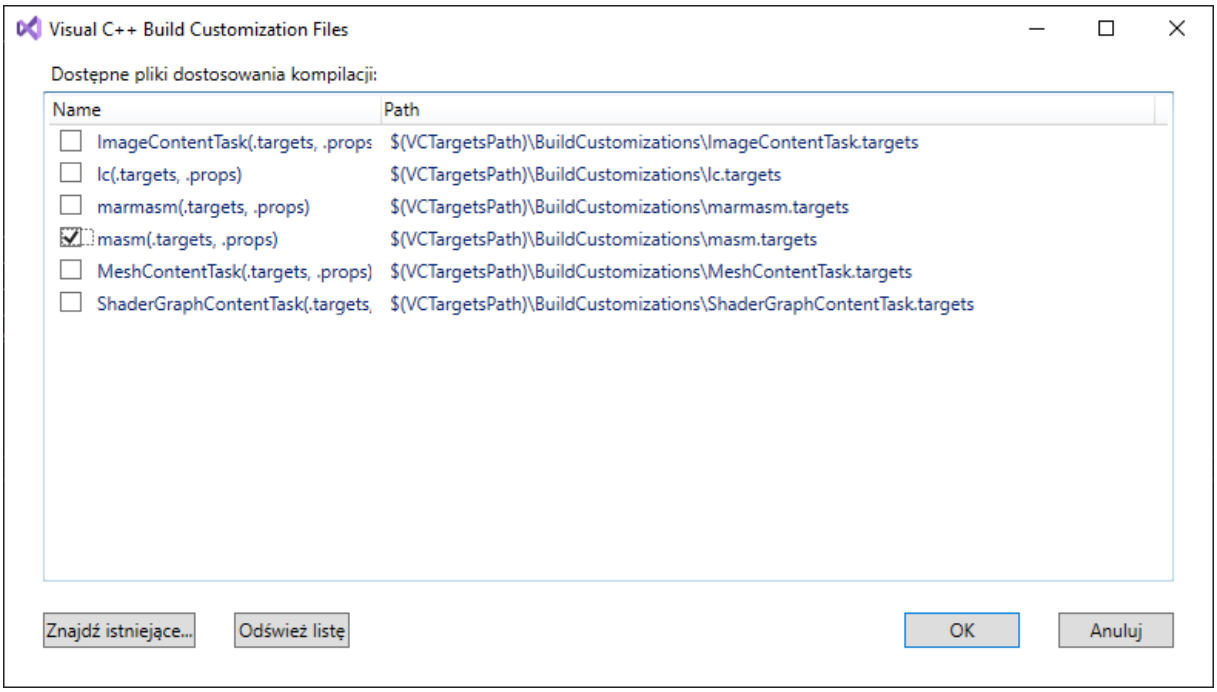
8. Następnym krokiem jest wyczyszczenie projektu biblioteki z automatycznie tworzonych plików. W tym celu wystarczy zaznaczyć wskazane pliki nagłówkowe i źródłowe, a następnie wybrać PPM i „Usuń” bądź też kliknąć Delete.



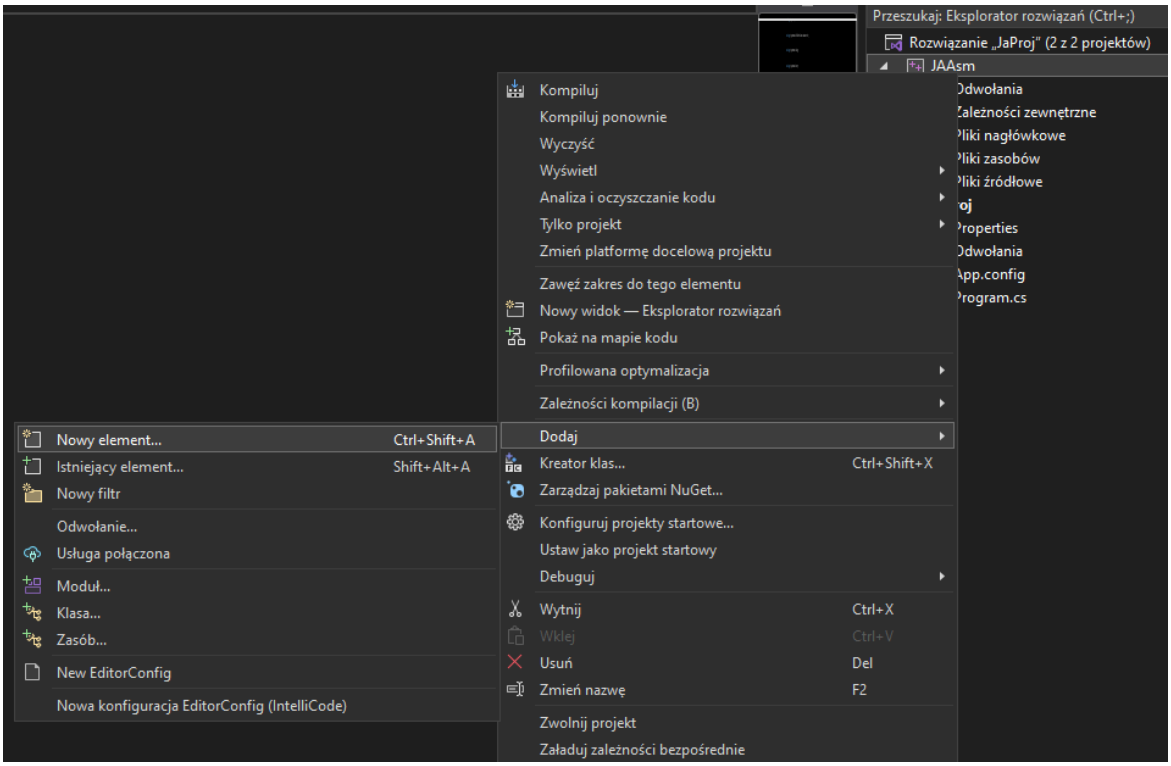
9. Dla biblioteki konieczne jest wskazanie dostosowania kompilacji (kompilacja jako masm). W tym celu klikamy projekt biblioteki PPM i z listy wybieramy „Zależności kompilacji” a następnie „Dostosowanie kompilacji…”



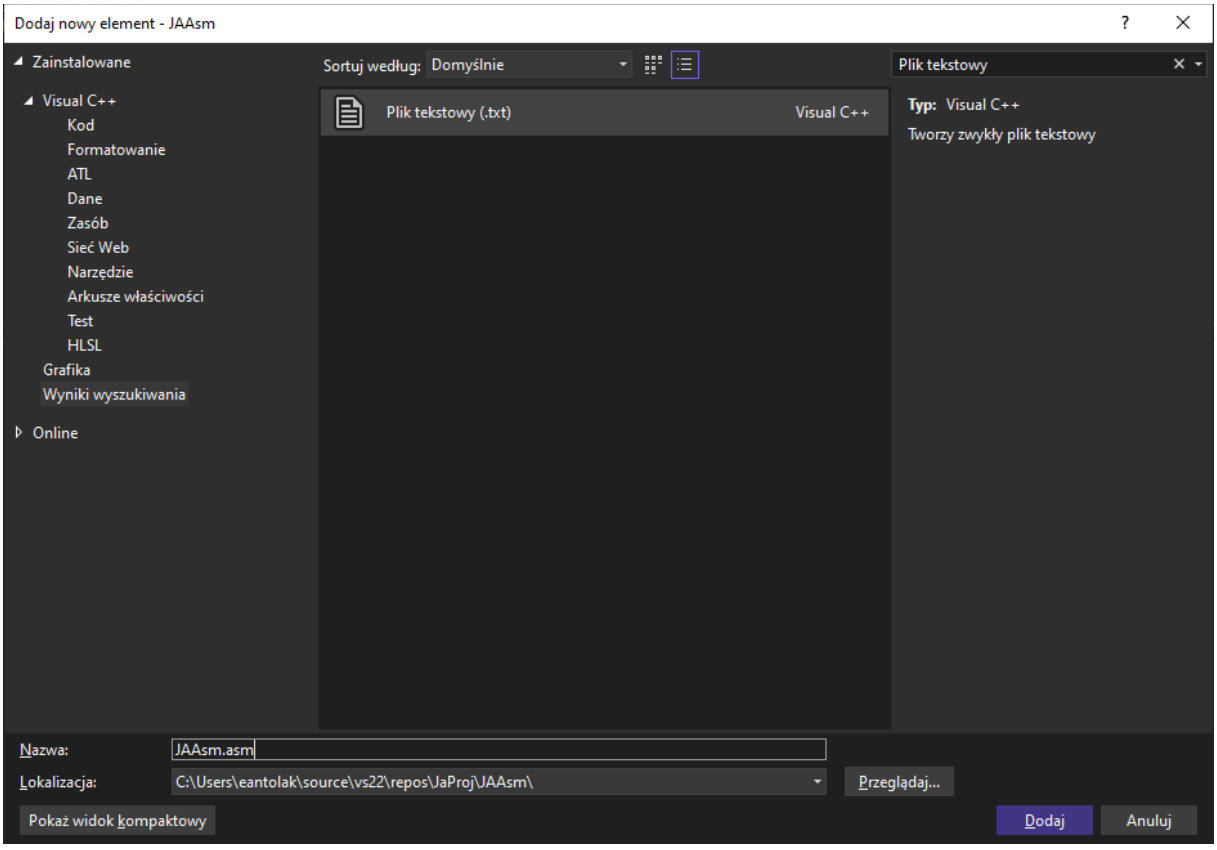
10. W nowo otwartym oknie zaznaczamy „masm" i potwierdzamy nasz wybór „OK"



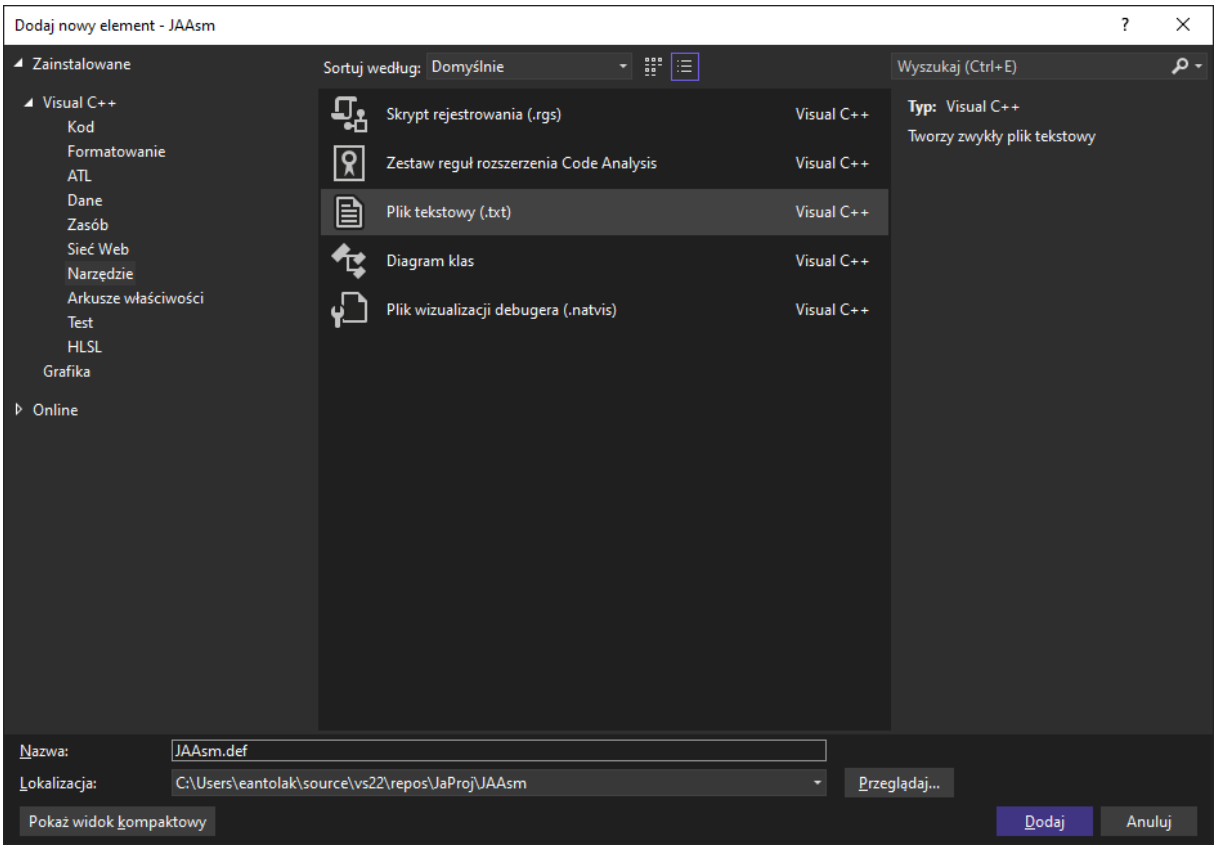
11. Kolejnym krokiem jest dodanie do biblioteki plików źródłowego oraz definicji. Aby to zrobić klikamy PPM na naszą bibliotekę, następnie wybieramy „Dodaj" oraz „Nowy element…"



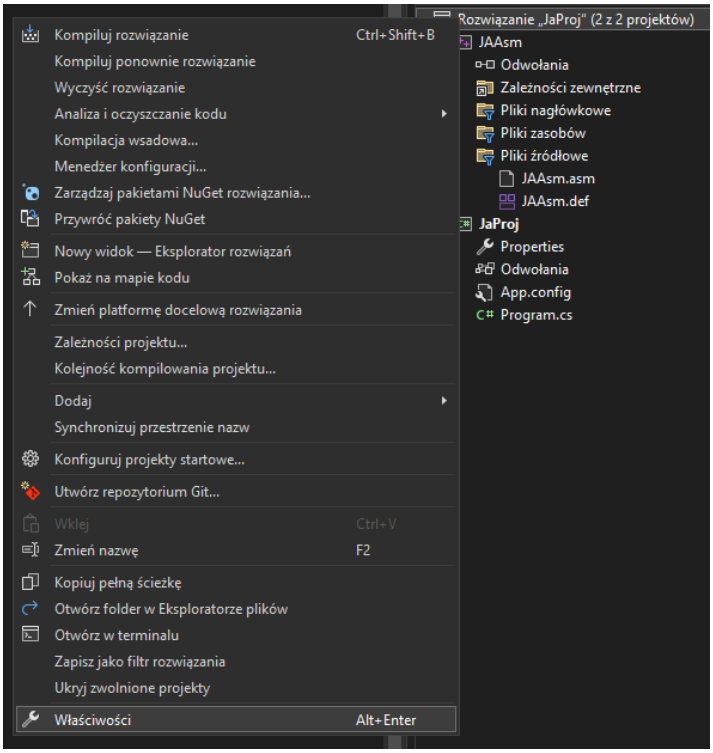
12. W nowo otwartym oknie wyszukujemy typ „Plik tekstowy" oraz wprowadzamy nazwę zakończoną rozszerzeniem „.asm" i klikamy „Dodaj".



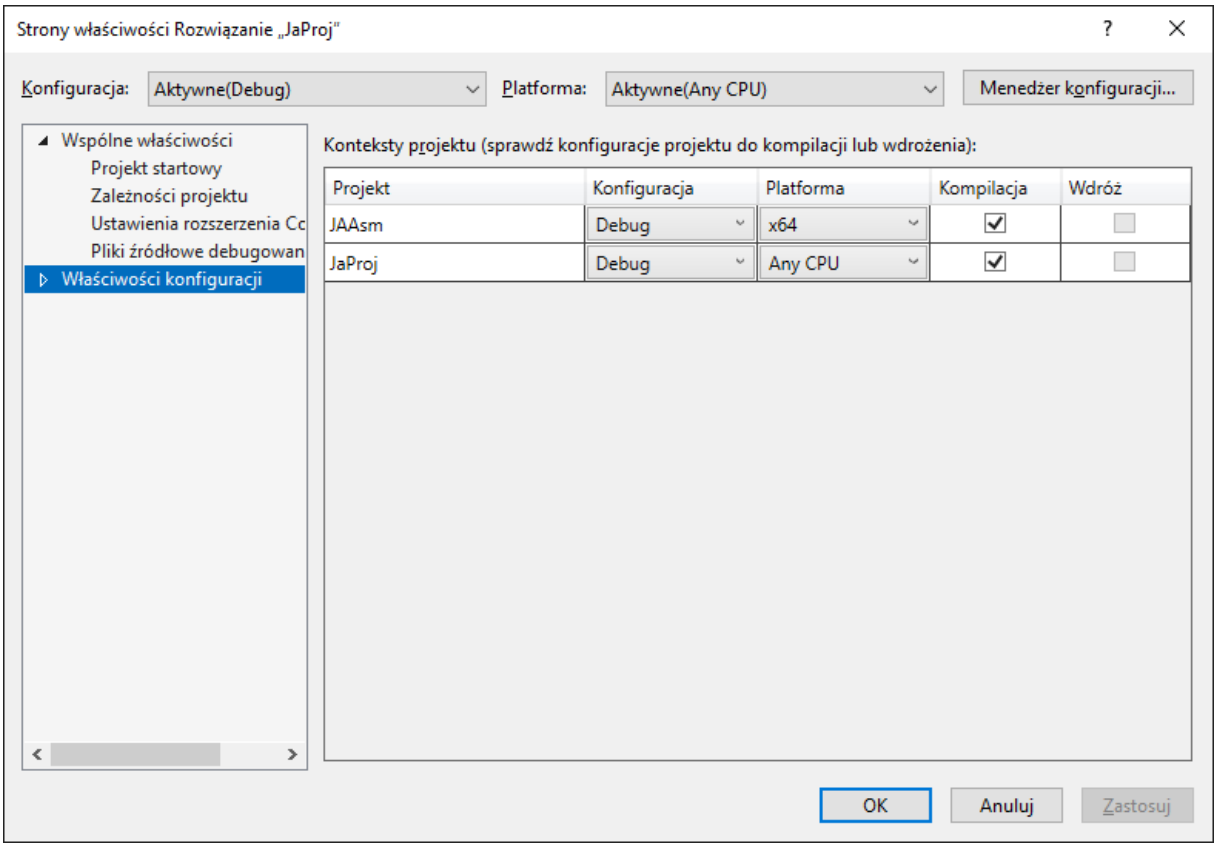
13. W analogiczny sposób dodajemy plik „.def”.



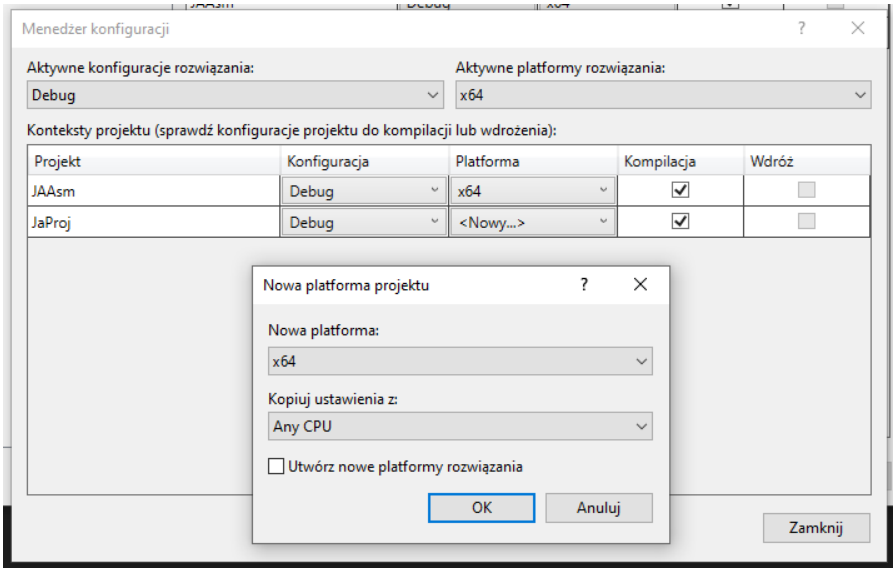
14. Następnie ustawiamy kompilację w trybie x64. W tym celu klikamy PPM na nasze rozwiązanie i wybieramy „Właściwości”

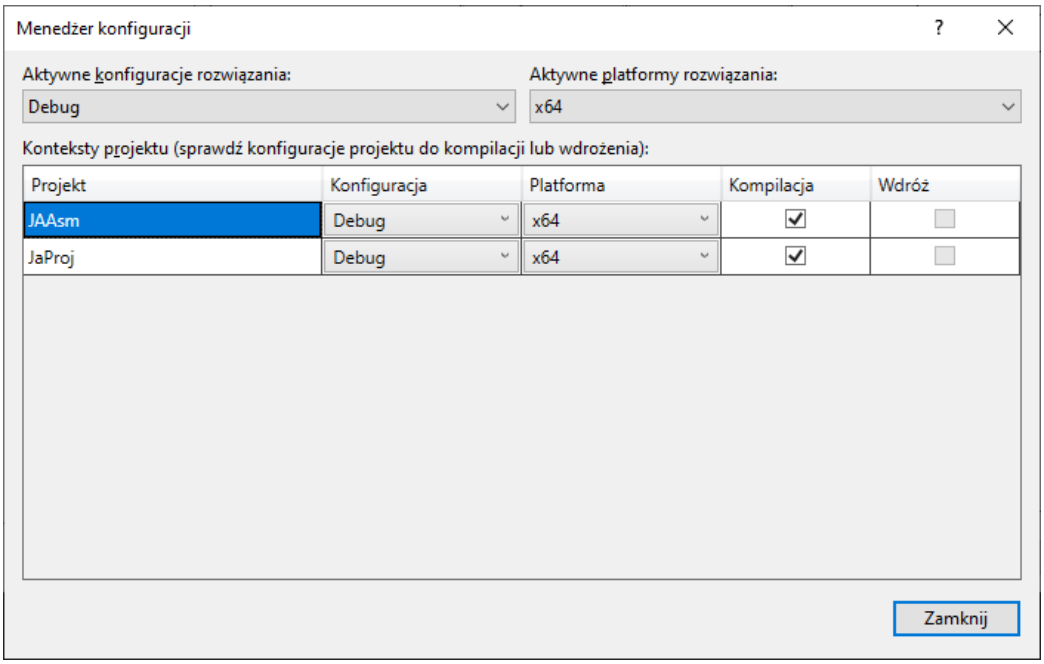


15. W oknie przechodzimy na zakładkę „Właściwości konfiguracji” -> „Konfiguracja” na następnie klikamy w „Menedżer konfiguracji”

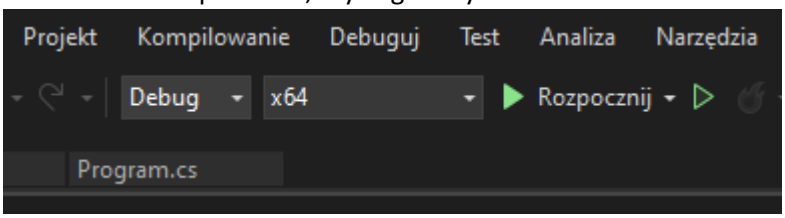


16. W nowym oknie wskazujemy wszędzie platformę x64 (gdyby dana opcja była niedostępna z dostępnej listy wybieramy pozycję „Nowy” a następnie tworzymy platformę x64 poprzez kopiowanie ustawień). Po wszystkim upewniamy się raz jeszcze, że wszędzie wskazana platforma to x64 i zamykamy okno

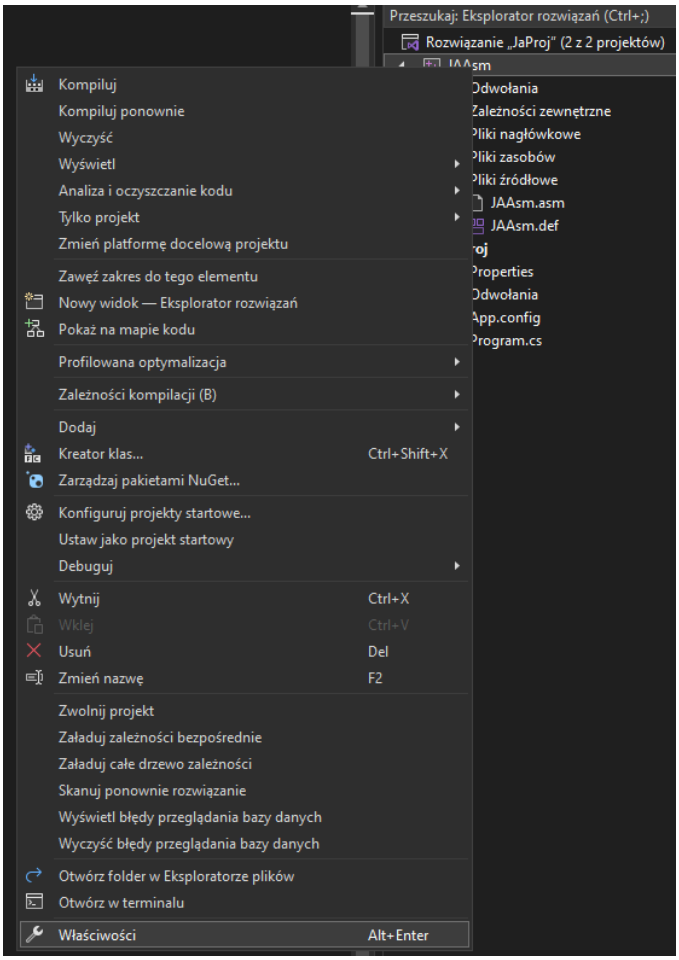




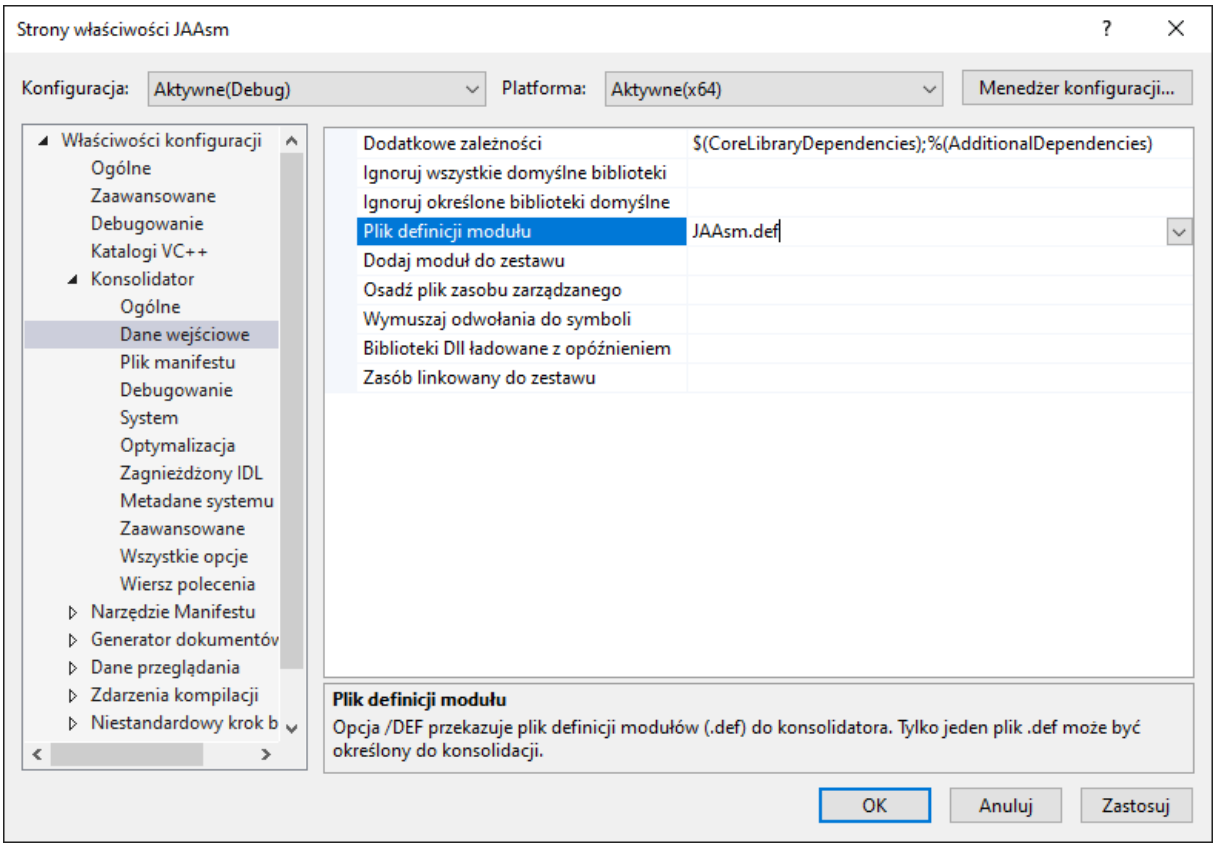
17. Warto również sprawdzić, czy w głównym oknie VS nasze ustawienia to „x64”



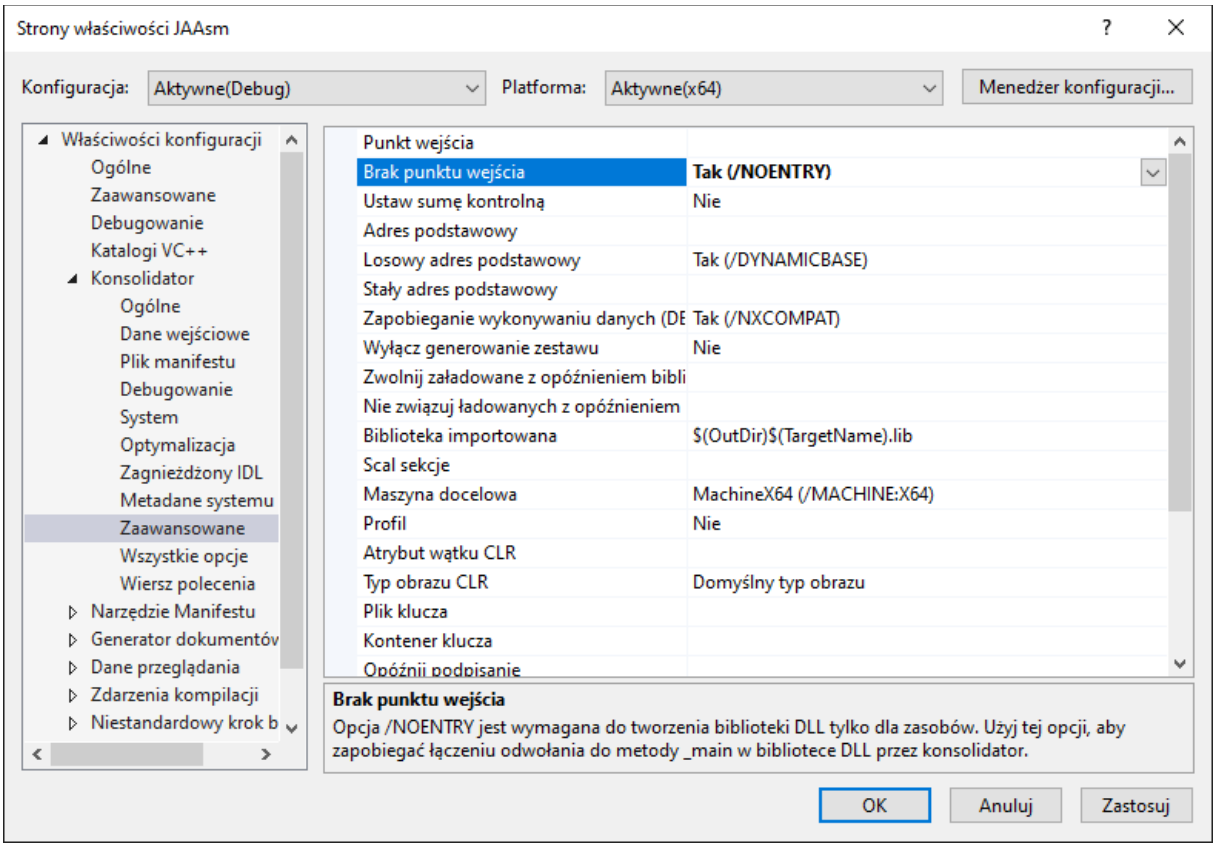
18. Dalej przystępujemy do konfiguracji biblioteki. W tym celu klikamy PPM na bibliotece i wybieramy „Właściwości”



19. W oknie przechodzimy do „Konsolidator" -> „Dane wejściowe" i w polu „Plik definicji modułu" wpisujemy nazwę naszego pliku „*.def



20. Przechodzimy do zakładki „Konsolidator” -> „Zaawansowane” i w polu „Brak punktu wejścia” wybieramy opcję „Tak (/NOENTRY)”. Zapisujemy wprowadzone ustawienia i zamykamy okno.



21. Finalnym etapem jest wprowadzenie kodu sprawdzającego poprawność utworzenia przez nas rozwiązania:

    W tym celu w pliku .asm wprowadzamy PRZYKŁADOWY kod:

```
.code
MyProc1 proc
add RCX, RDX
mov RAX, RCX
ret
MyProc1 endp
end
```

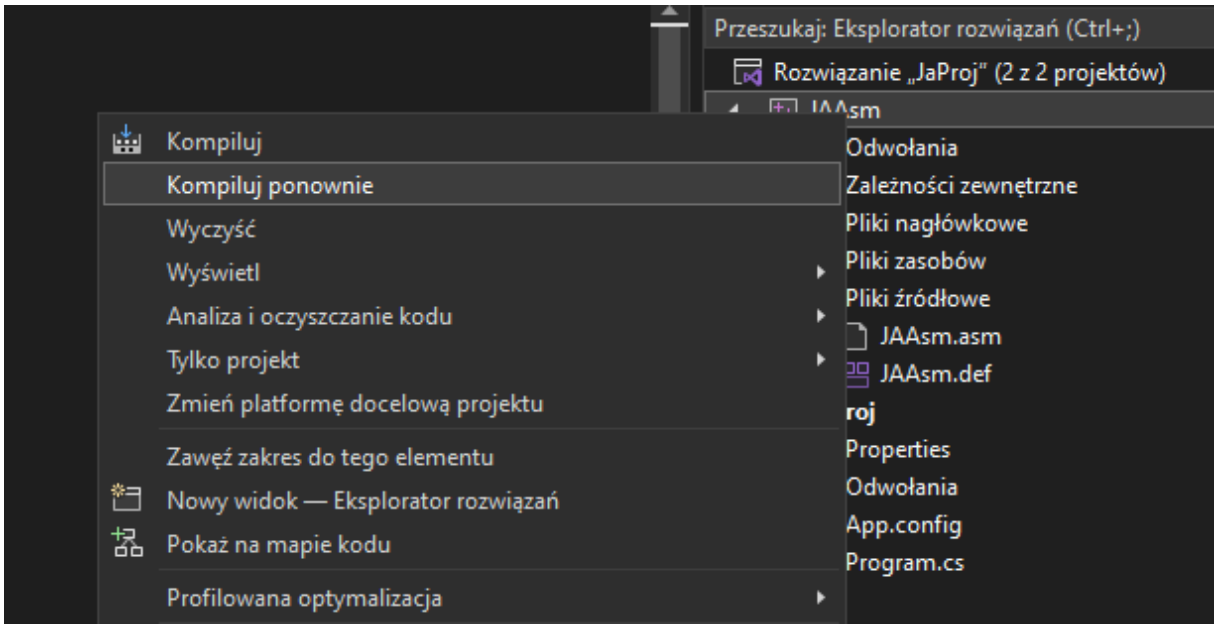
    W pliku .def deklarujemy utworzoną procedurę:

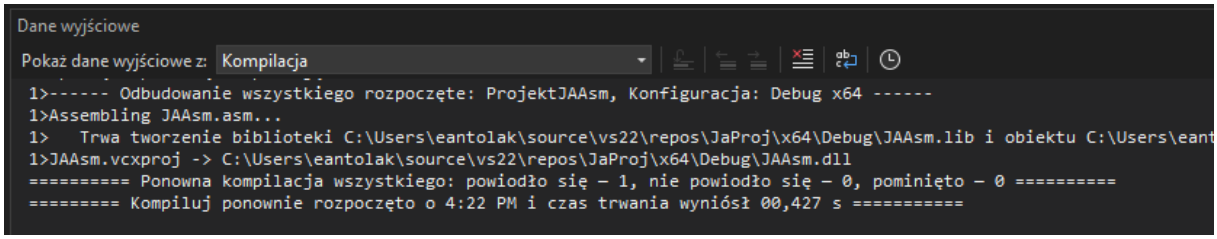
```
LIBRARY JAAsm
EXPORTS MyProc1
```

    A w pliku konsolowym, przed funkcją Main deklarujemy import biblioteki dll i użycie funkcji bibliotecznej:

```
[DllImport(@"ŚCIEŻKA DO PLIKU DLL ")]
static extern int MyProc1(int a, int b);
```

    Jeżeli nie jesteśmy pewni, jaka jest ścieżka do pliku dll wystarczy kliknąć PPM na naszą bibliotekę i wybrać opcję „Kompiluj ponownie”. Wówczas w terminalu VS otrzymamy komunikat o przebiegu kompilacji, wraz z wskazaniem ścieżki do utworzonej biblioteki:





Uwaga! Skorzystanie z funkcji DllImport może wymagać dołączenia dodatkowej referencji. VS zasugeruje to automatycznie, jednak gdyby tak się nie stało potrzebujemy dołączyć ją ręcznie:

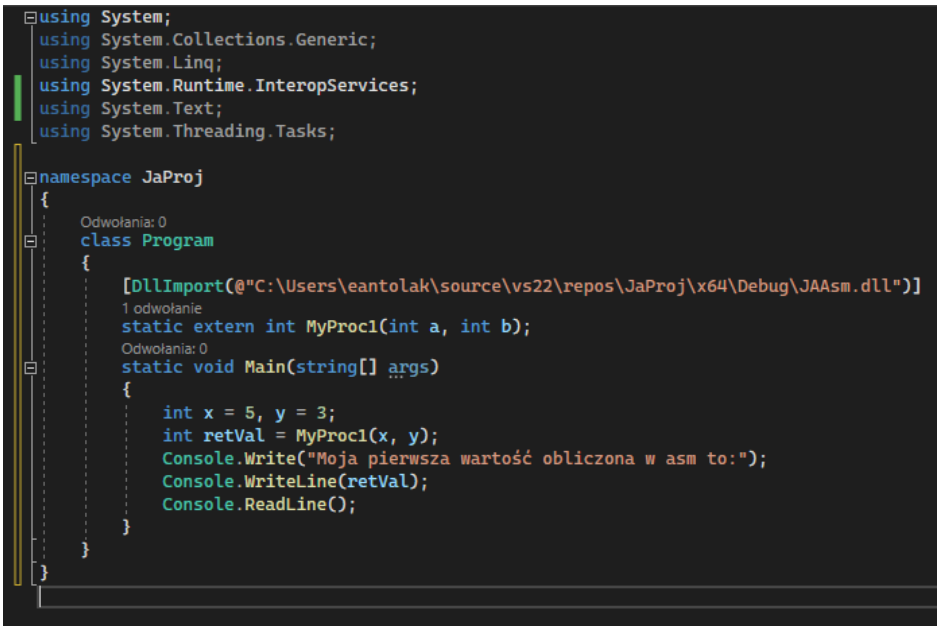
```
using System.Runtime.InteropServices;
```

22. Po poprawnie przeprowadzonej konfiguracji, możemy skorzystać z utworzonej przez nas funkcji dodającej w asemblerze. W konsoli wystarczy utworzyć zmienne i wywołać funkcję:

```
int x = 5, y = 3;
int retVal = MyProc1(x, y);
```

Całość kodu w pliku .cs może wyglądać w sposób następujący:

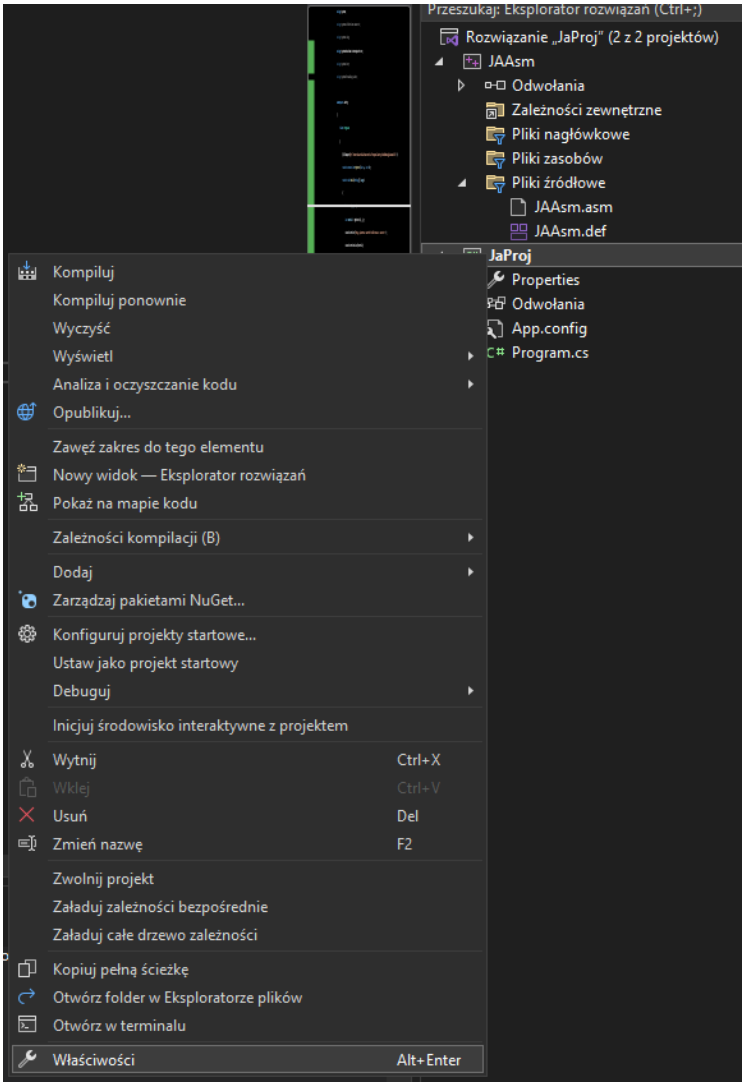
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;

namespace JaProj
{
    class Program
    {

[DllImport(@"C:\Users\eantolak\source\vs22\repos\JaProj\x64\Debug\JAAsm.dll")]
        static extern int MyProc1(int a, int b);
        static void Main(string[] args)
        {
            int x = 5, y = 3;
            int retVal = MyProc1(x, y);
            Console.Write("Moja pierwsza wartość obliczona w asm to:");
            Console.WriteLine(retVal);
            Console.ReadLine();
        }
    }
}
```



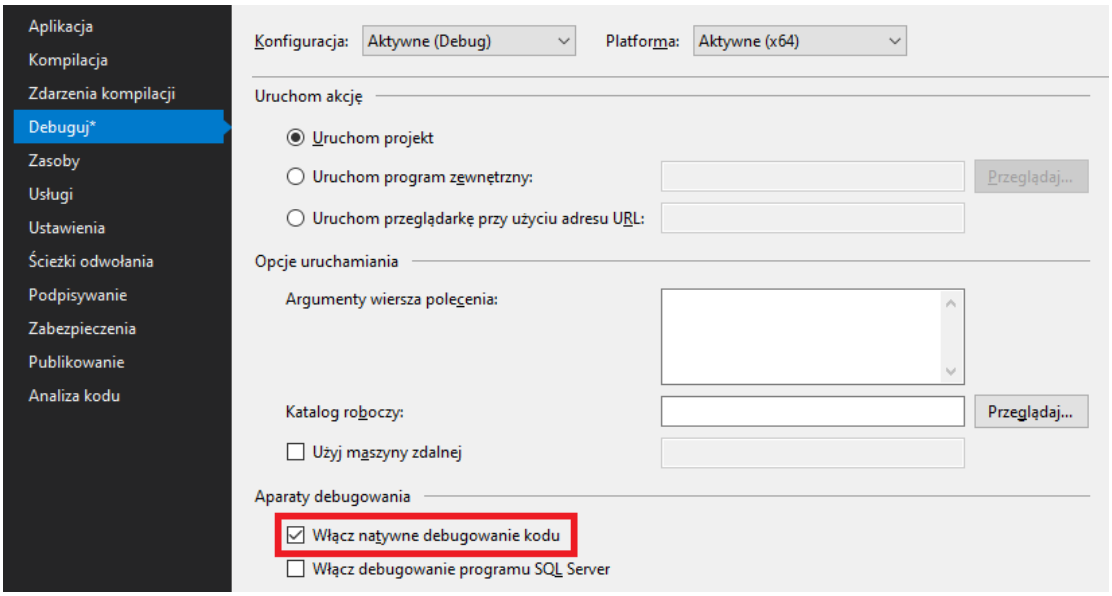
Link do tutorialu na YouTube: https://youtu.be/7lBb6eN_gbQ (VS2019)

Konfiguracja środowiska do debugowania:

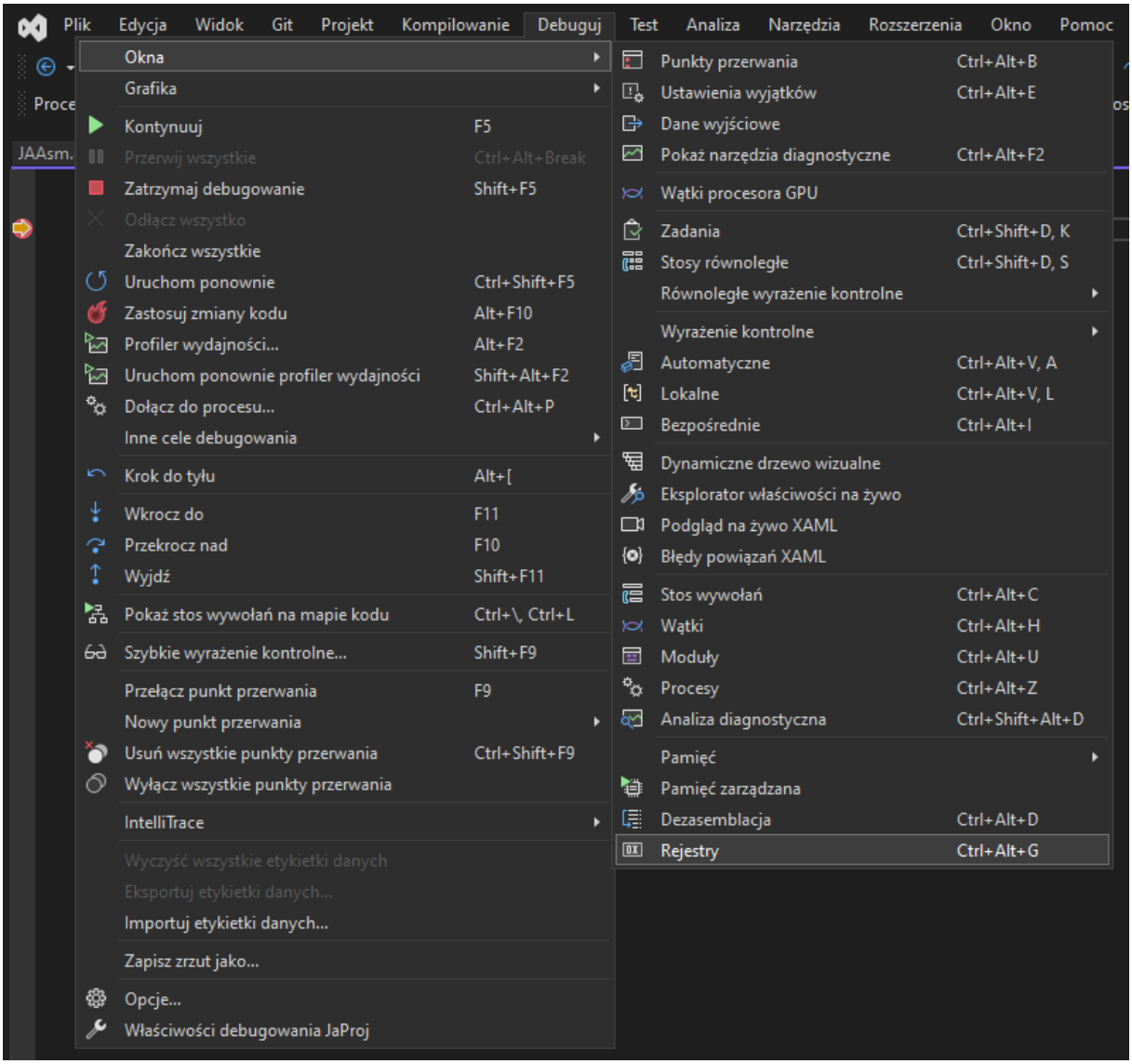
Aby umożliwić korkowe zatrzymanie programu na brakepoint'cie który znajduje się w części asemblerowej projektu, należy włączyć „**natywne debugowanie kodu**". W tym celu należy kliknąć PPM na projekt konsolowy i z rozwijanej listy wybrać „Właściwości":



Następnie wybieramy z menu „Debuguj" i zaznaczmy opcje „Włącz natywne debugowanie kodu". Jeżeli projekt jest innego typu jak „Aplikacja konsoli (.NET Framework)", opcja ta może znajdować się w innym miejscu, lecz zazwyczaj zawsze ją można znaleźć w menu „Debuguj*" (może być w jakimś podmenu)



Aby włączyć podgląd rejestrów procesora, podczas zatrzymanej pracy programu (podczas debugowania), należy wybrać menu „Debuguj” -> „Okna” i następnie „Rejestry”:



W nowo wyświetlonym oknie „Rejestry”, klikamy PPM i z rozwiniętej listy wybieramy interesujące nas rejestry (w tym wypadku będzie to „CPU” oraz „Flagi”):