Programowanie niskopoziomowe

Ćwiczenia laboratoryjne w środowisku Visual Studio 2019

Labolatoria 10

„Wstawki Assemblerowe”

Spis treści

1 Na laboratorium

1. Dodawanie wstawek assemblerowych w programie napisanym w c++,

2. Tworzenie biblioteki w assemblerze i wykorzystanie jej w programie napisanym w c++.

2 Zadania

1. Napisz w języku asemblera fragment programu, obliczający wartość funkcji y od czterech argumentów całkowitych według wzoru dla swojego zadania (patrz tabela wariantów). W środowisku Visual Studio 2015 utwórz aplikację konsolową(4.1). Zczytaj z klawiatury od użytkownika 3 argumenty, a następnie za pomocą wstawki oblicz wartoć funkcji(4.2).

2. Utwórz nowy projekt, bibliotekę statyczną napisaną w assemblerze(4.3). W bibliotece stwórz procedurę dla czterech parametrów i zwracającą wynik funkcji y(jak w zadaniu 1). Podmień wstawkę z zadania 1 na wywołanie procedury.

3. W bibliotece utworz procedurę wypełniającą bufor znaków o rozmiarze 4000 znakami spacji. Procedura przyjmuje 1 parametr, adres bufora. Pamiętaj że w assemblerze masz możliwoć przenoszenia 4 bajtów naraz.

4. Napisz procedurę realizującą funkcjonalność zadania 3 w c++. Porównaj czasy wykonań obu procedur

---

3 Tabela wariantów

Tabela 1: Tabela wariantów

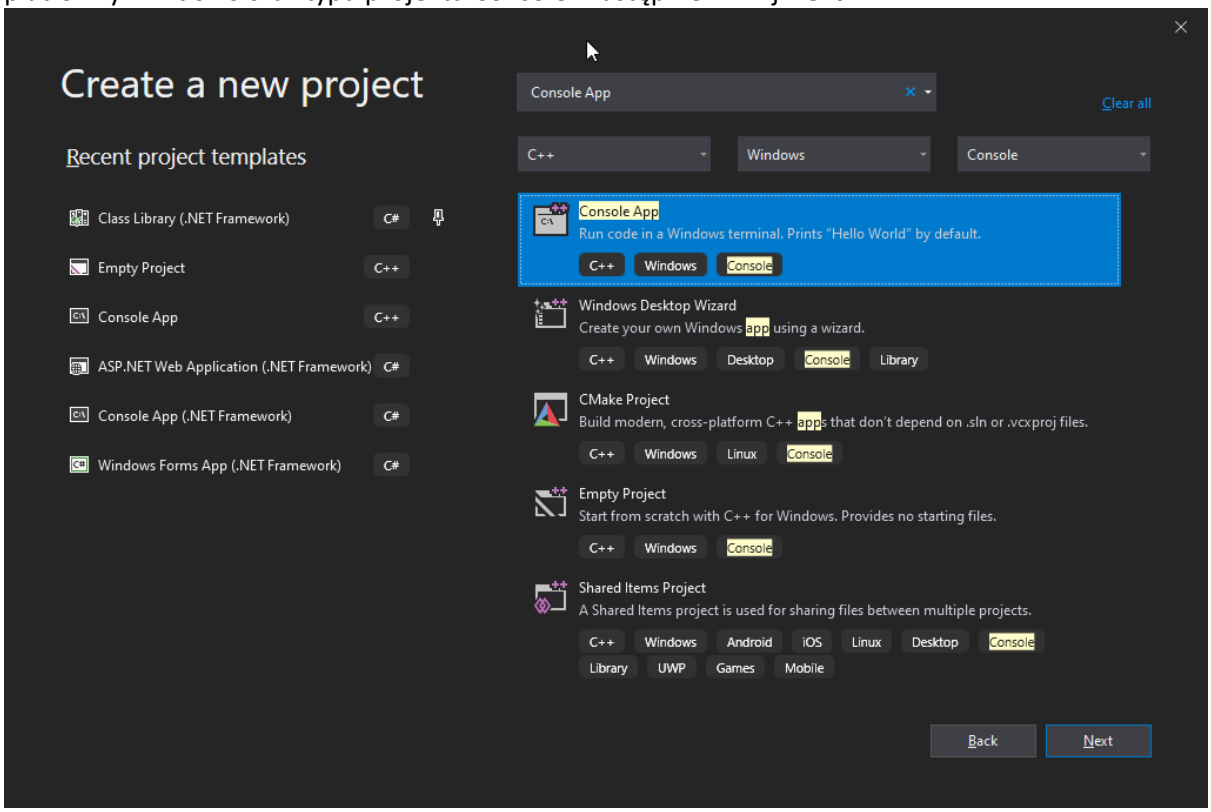
Numer Wariantu	Równanie
1	5*A+4*B-C
2	25*A - 4*A*B - C
3	7*(A-B)+C
4	11*A + B*C
5	2*A*B*C
6	2*A*C + 4*B*C
7	6*A + 4*B - 2*B*C
8	7*A + 2*B*C
9	2*A-B-C
10	(2*A + B + C)*C
11	5*A + 2*B - 10*C
12	17*A - 2*B - 2*C
13	10*A - 10*B - 10*C
14	2*A + 2*B - 2*C
15	5*A - 2*A*B - 4*B*C + 2*C

4 Pomoc

4.1 Tworzenie projektu konsolowego c++ w VS2019

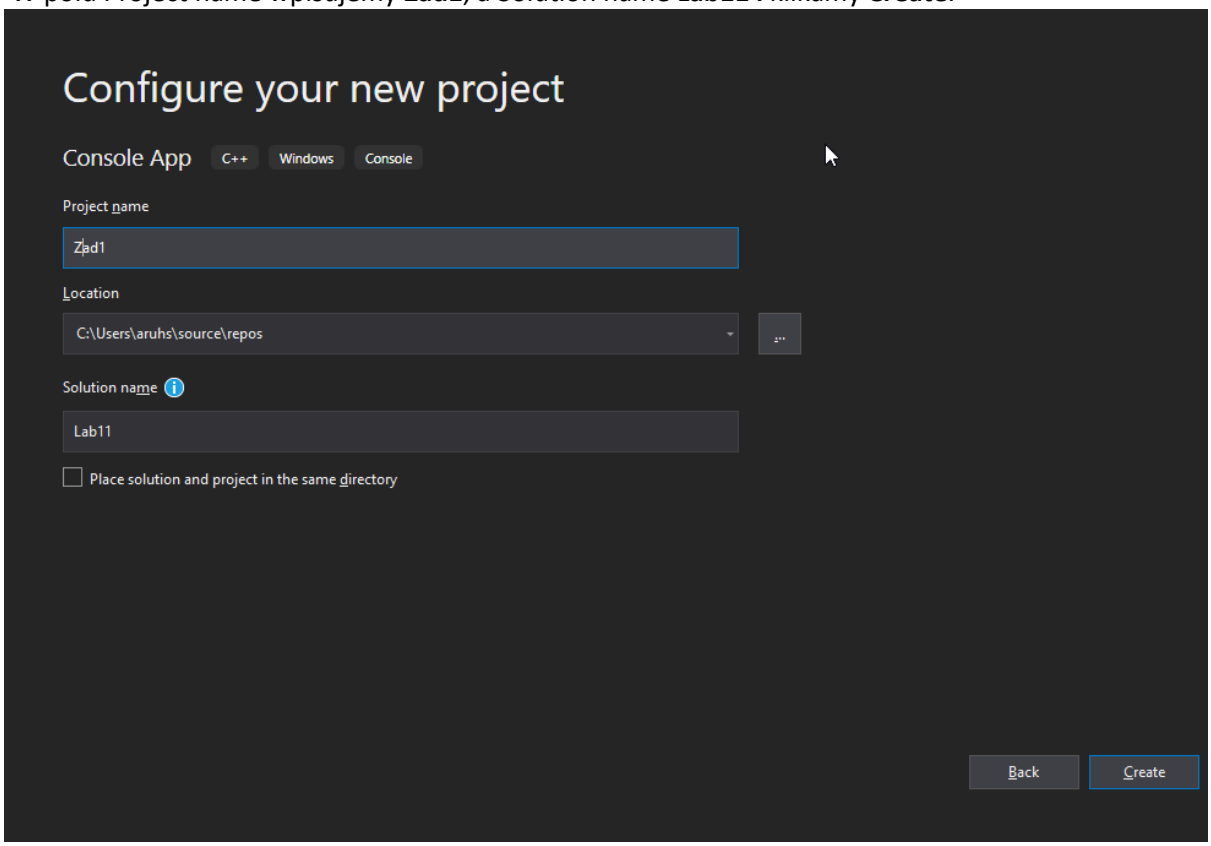
1. Uruchamiamy VS2019, a następnie klikamy File->New->Project

2. Następnie po prawiej stronie wybieramy Projekt temple **Console App** dla jezyka c++, platformy Windows oraz typu projektu Console. Następnie kliknij **Next**



Rysunek 1: Tworzenie nowego projektu

3. W polu Project name wpisujemy **Zad1**, a Solution name **Lab11 i** klikamy **Create**.



Rysunek 2: Tworzenie nowego projektu

4.2 Przykładowy program z wstawką assemblerową

Poniżej przedstawiony jest program który kolejno:

1. Wypisuje zachętę na konsolę,
2. Zczytuje wprowadzoną wartoć do zmiennej o nazwie *zm1*,
3. Pobiera adres zmiennej *zm1* do rejestru *EBX*, a następnie wstawia wartoć 10 w to miejsce w pamięci,
4. Wypisuje liczbę spod zmiennej *zm1*.

```
1   #include <iostream>
2
3   using namespace std;
4
5
6   int zm1;
7   int main()
8   {
9       cout << "Wprowadź pierwszą liczbę" << endl;
10      cin >> zm1;
11
12      _asm {
13          mov EBX,offset zm1
14          mov DWORD PTR [EBX],10
15      }
16
17      cout << "Liczba po zmianach to: " << zm1 << endl;
18      system("pause");
19      return 0;
20  }
21
22
```

Rysunek 3: Przykładowy program z wstawką assemblerową

Wstawka musi być umieszczona w nawiasach klamrowych po słowie kluczowym asm, lub asm. Można zamiast nawiasów klamrowych zapisać słowo kluczowe na początku każdego wiersza, na przykład:

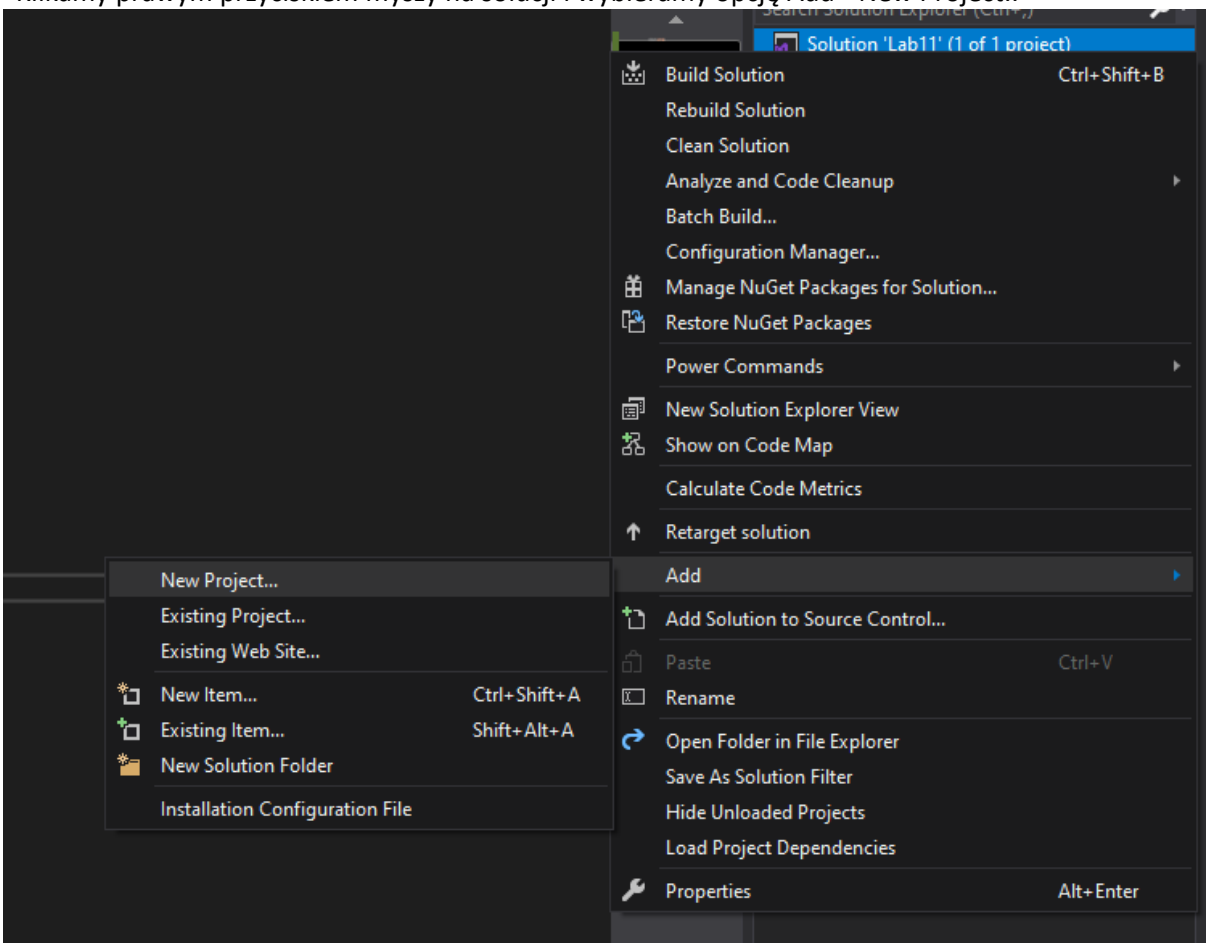
```
22      _asm push 10
```

Rysunek 4: Przykład wstawki bez klamr

4.3 Wykorzystanie biblioteki napisanej w assemblerze

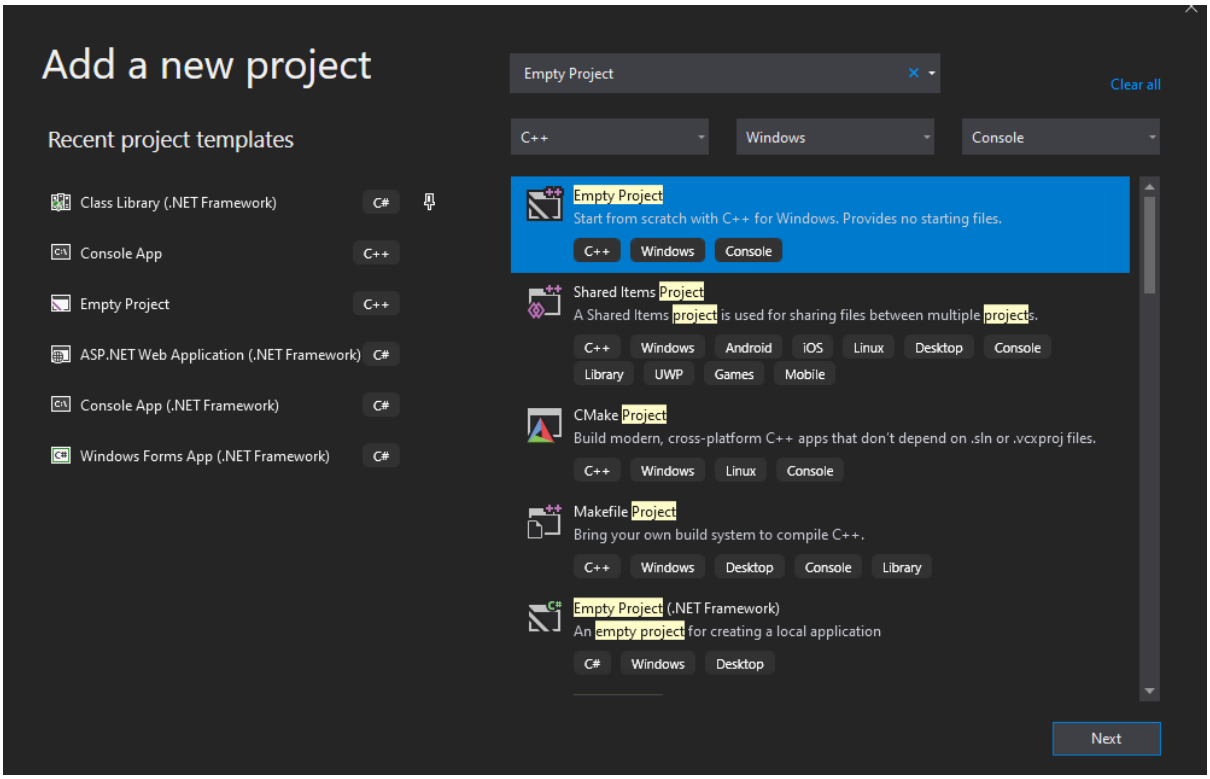
By wykorzystać zewnętrzną procedurę najpierw musimy utworzyć bibliotekę. W tym celu two-rzymy nowy projekt:

1. Klikamy prawym przyciskiem myszy na solucji i wybieramy opcję Add->New Project..



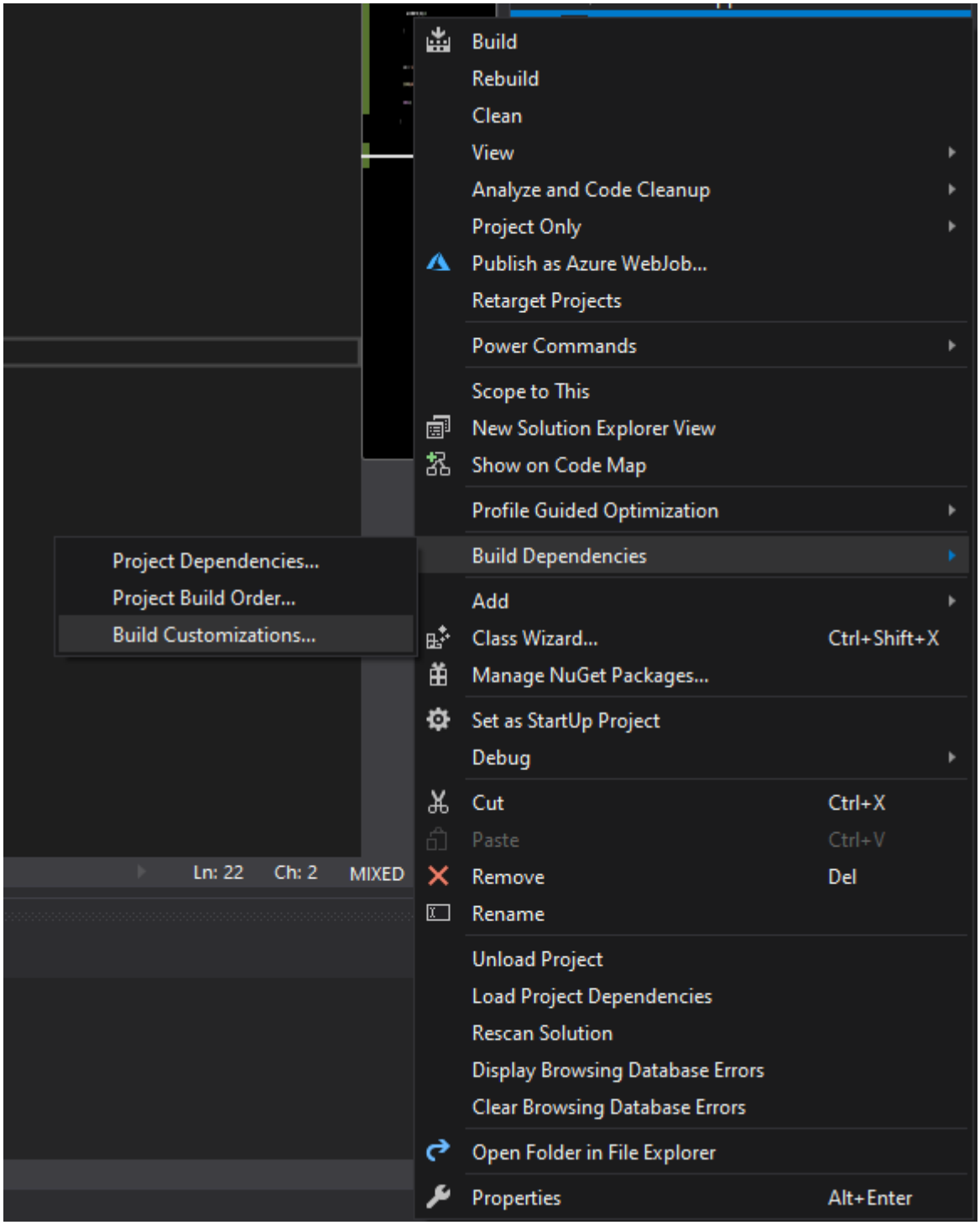
Rysunek 5: Tworzenie biblioteki w assemblerze

2. Wybieramy **Empty project** i nazywamy projekt **Zad2Lib** i klikamy **Ok**.



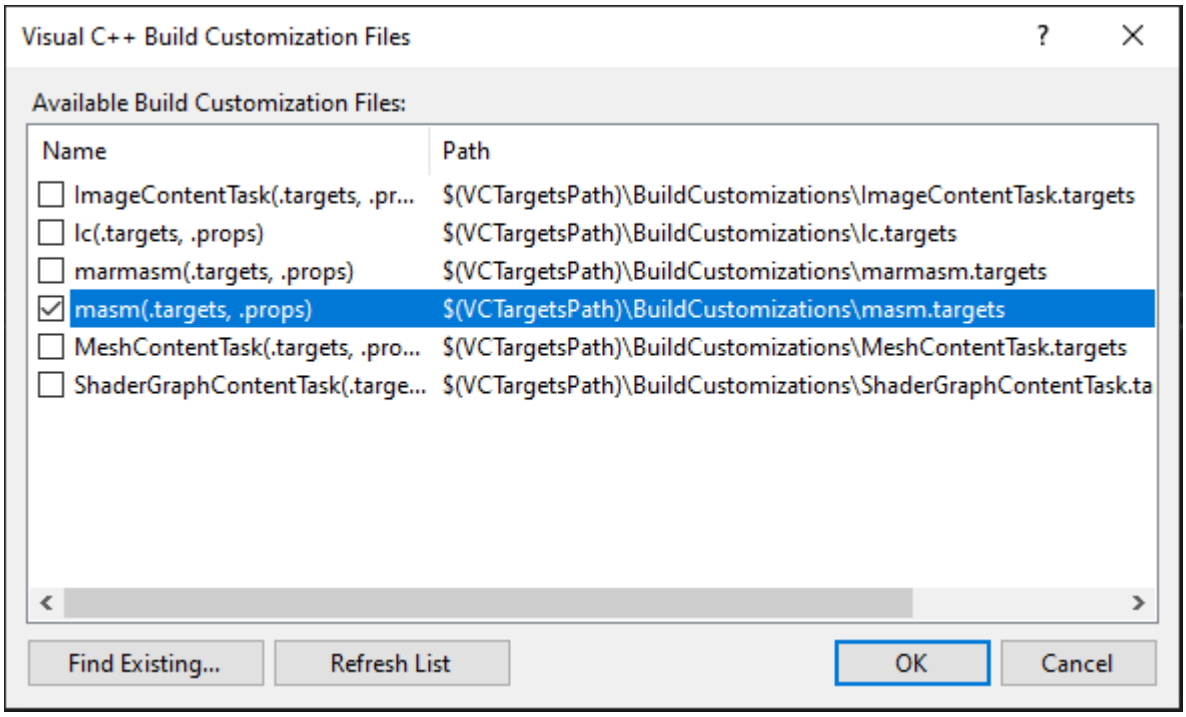
Rysunek 6: Tworzenie biblioteki w assemblerze

3. Klikamy prawym przyciskiem myszy na nowo utworzonym projekcie i wybieramy **Build Dependencies**->**Build Customization**



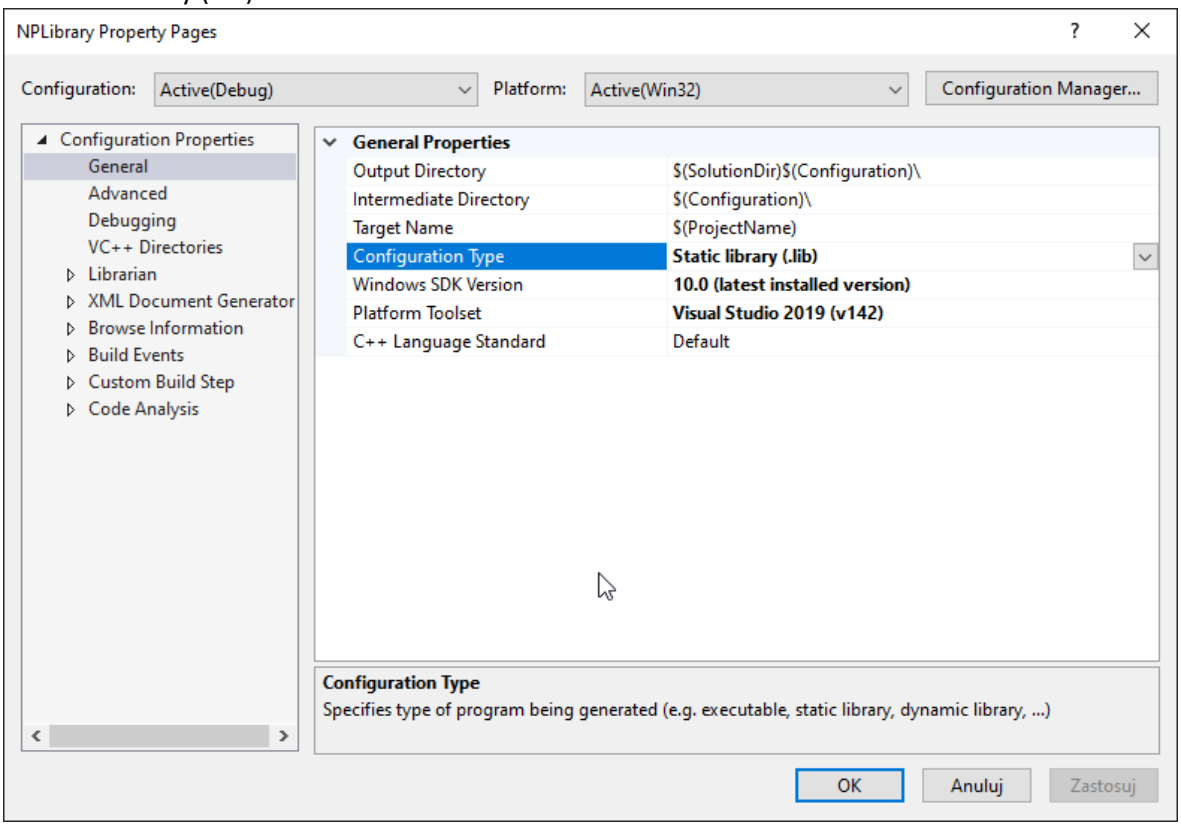
Rysunek 7: Tworzenie biblioteki w assemblerze

4. Zaznaczamy opcję **masm32**.



Rysunek 8: Tworzenie biblioteki w assemblerze

5. Wchodzimy w właściwości utworzonego projektu klikając prawym przyciskiem myszy na projekcie i wybieramy opcję Properties.
6. Przechodzimy do **Configuration properties** -> **General** i zmieniamy **Configuration type** na Static library (.lib).



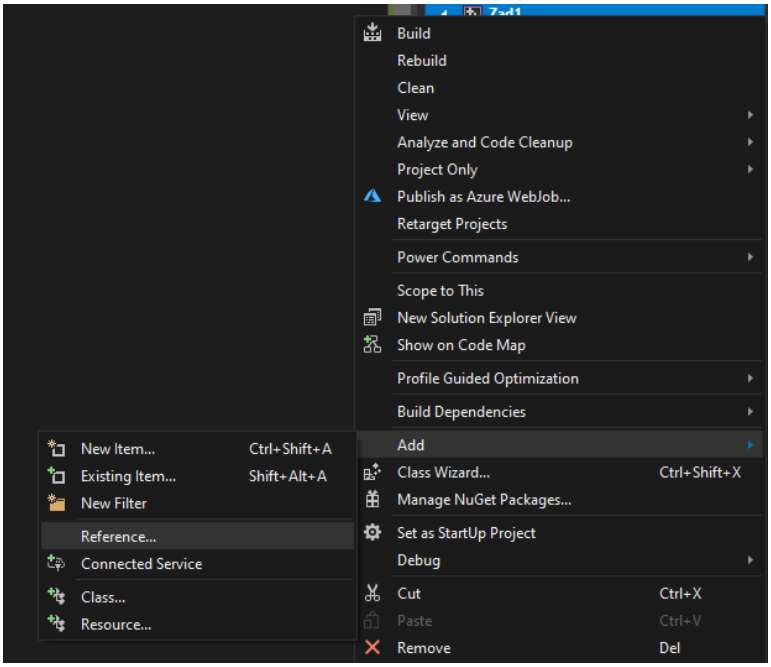
Rysunek 9: Tworzenie biblioteki w assemblerze

7. Do katalogu **Source Files** dodajemy nowy plik **Source.asm** i wypełniamy przykładowym kodem.

```
.386
.MODEL FLAT,STDCALL

PUBLIC IncrementNumber

.code
    IncrementNumber PROC liczba:DWORD
        inc liczba
        mov EAX,liczba

        ret

    IncrementNumber ENDP

END
```

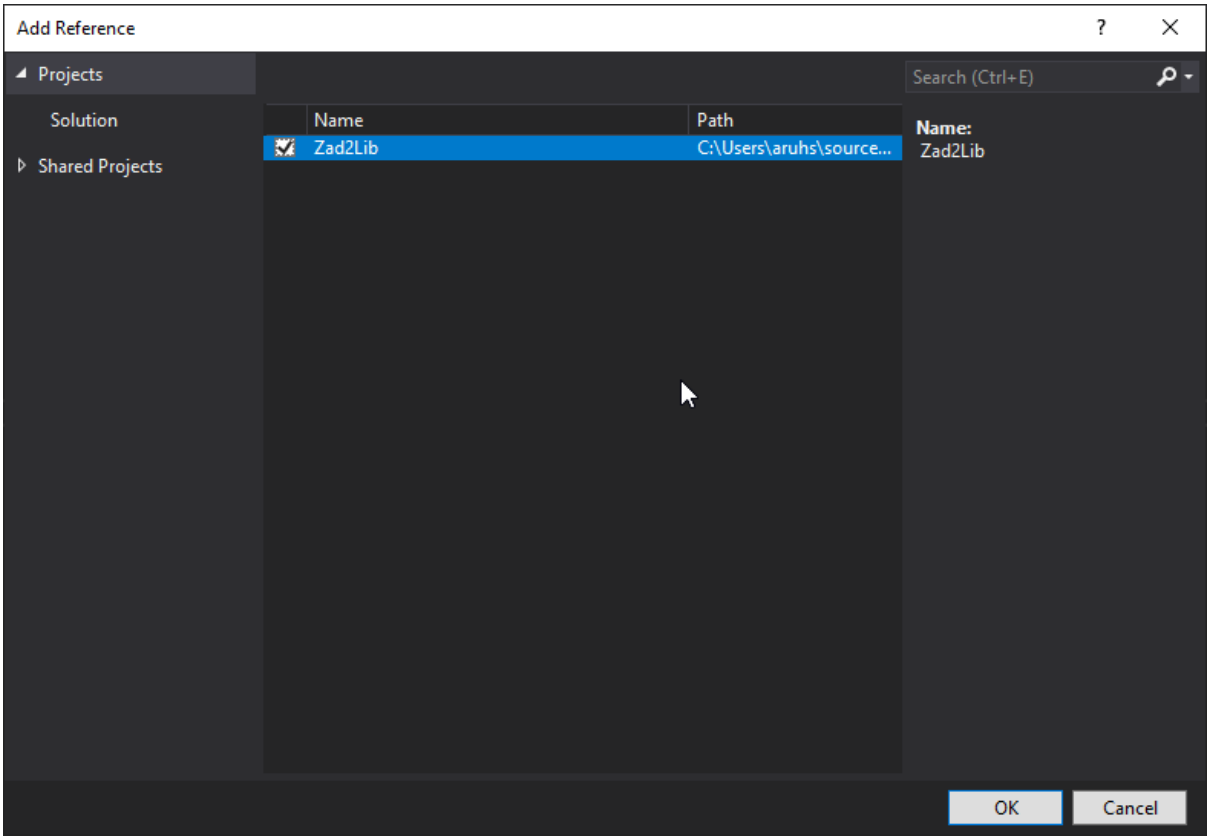
Rysunek 10: Tworzenie biblioteki w assemblerze

8. Dołączamy bibliotekę do projektu utworzonego w zadaniu 1 klikając prawym przyciskiem myszy na projekcie c++ i wybieramy opcję Add->Reference...



Rysunek 11: Dodanie biblioteki do projektu

9. Wybieramy projekt do którego referencję chcemy załączyć(Zad2Lib) i klikamy **OK**.



Rysunek 12: Dodanie biblioteki do projektu

10. By wykorzystać procedurę z utworzonej biblioteki musimy dodać odpowiedni prototyp w projekcie napisanym w c++.

```
#include "stdafx.h"
#include <iostream>
using namespace std;

extern "C" {
    int __stdcall IncrementNumber(int number);
}

int zm1;
int main()
{
    cout << "Wprowadź liczbę" << endl;
    cin >> zm1;
    zm1 = IncrementNumber(zm1);
    cout << "Liczba po inkrementacji to: " << zm1 << endl;
    system("pause");

    return 0;
}
```

4.4 Pomiar czasu wykonania

W celu zmierzenia czasu wykonania naszego programu i sprawdzenia wydajności możemy wykorzystać bibliotekę chronos.

```cpp
#include <iostream>
#include <chrono>

int main()
{
    std::chrono::steady_clock::time_point begin = std::chrono::steady_clock::now();
    std::chrono::steady_clock::time_point end = std::chrono::steady_clock::now();

    std::cout << "Time difference = " << std::chrono::duration_cast<std::chrono::microseconds>(end - begin).count() << "[µs]" << std::endl;
    std::cout << "Time difference = " << std::chrono::duration_cast<std::chrono::nanoseconds> (end - begin).count() << "[ns]" << std::endl;
    std::cin;
}
```

Rysunek 14: Pomiar czasu