Programowanie niskopoziomowe

Ćwiczenia laboratoryjne w środowisku Visual Studio 2019

Labolatoria 11

„Programowanie na architekturę 64 bitową”

Spis treści

1 Na laboratorium

1. Tworzenie projektu w architekturze 64 bitowej,
2. Różnice między programem 64 bitowym a 32 bitowym,
3. Prosty program obliczający równanie.

2 Zadania

1. Utwórz nowy projekt w architekturze 64 bitowej, zakończ go poprawnie za pomocą procedury ExitProcess.
2. Rozwiń zadanie 1 by wypisywało komunikat: „Witamy w architekturze 64 bitowej <imię>”. W miejsce <imię> wpisz swoje imię.
3. Rozwiń zadanie 2 by imię użytkownika było wprowadzane z konsoli.
4. Rozwiń zadanie 3 by po wprowadzeniu imienia użytkownik miał możliwość wprowadzenia 3 argumentów.
5. Oblicz równanie według przydzielonegowariantu.

---

3 Tabela wariantów

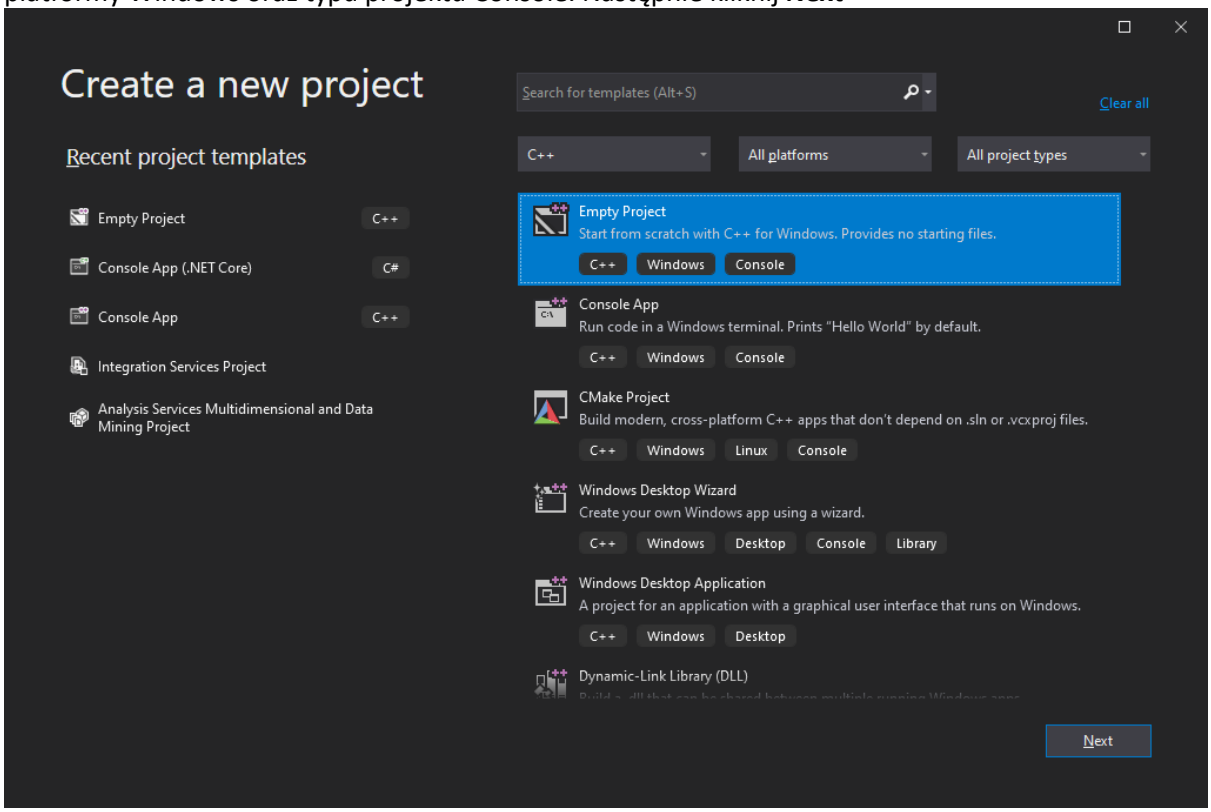
Tabela 1: Tabela wariantów

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

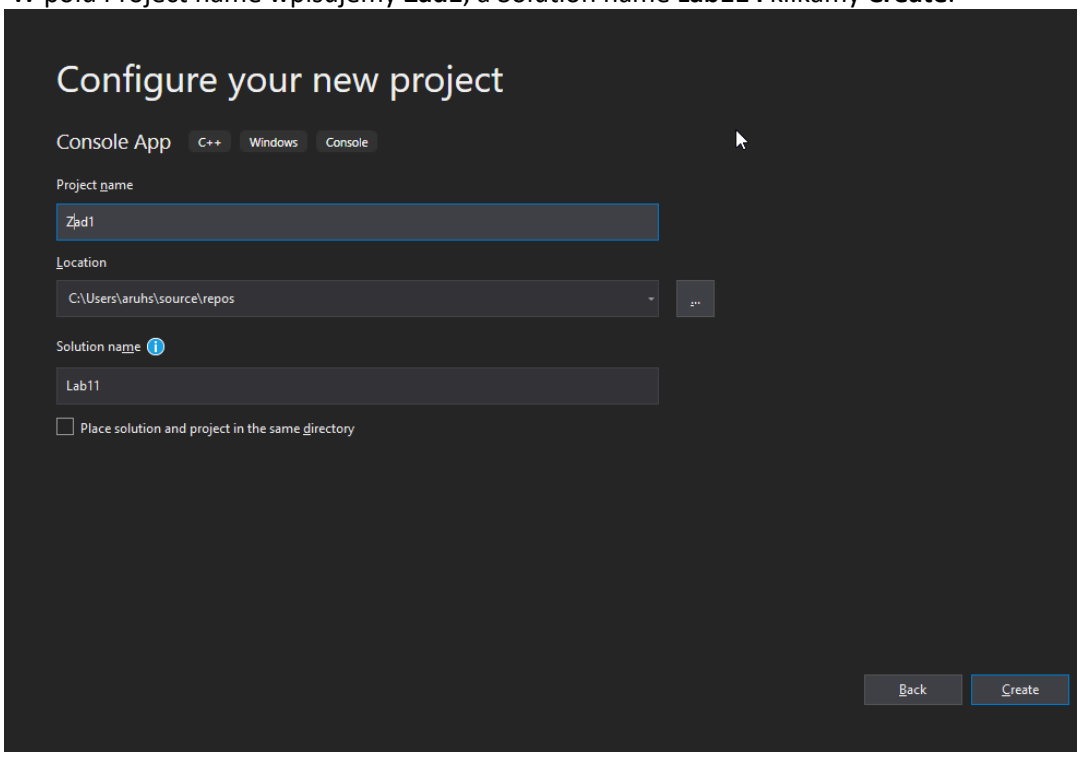
4.1 Tworzenie programu w architekturze 64 bitowej

1. Uruchamiamy VS2019, a następnie klikamy File->New->Project
2. Następnie po prawiej stronie wybieramy Projekt temple **Empty Project** dla jezyka c++, platformy Windows oraz typu projektu Console. Następnie kliknij **Next**



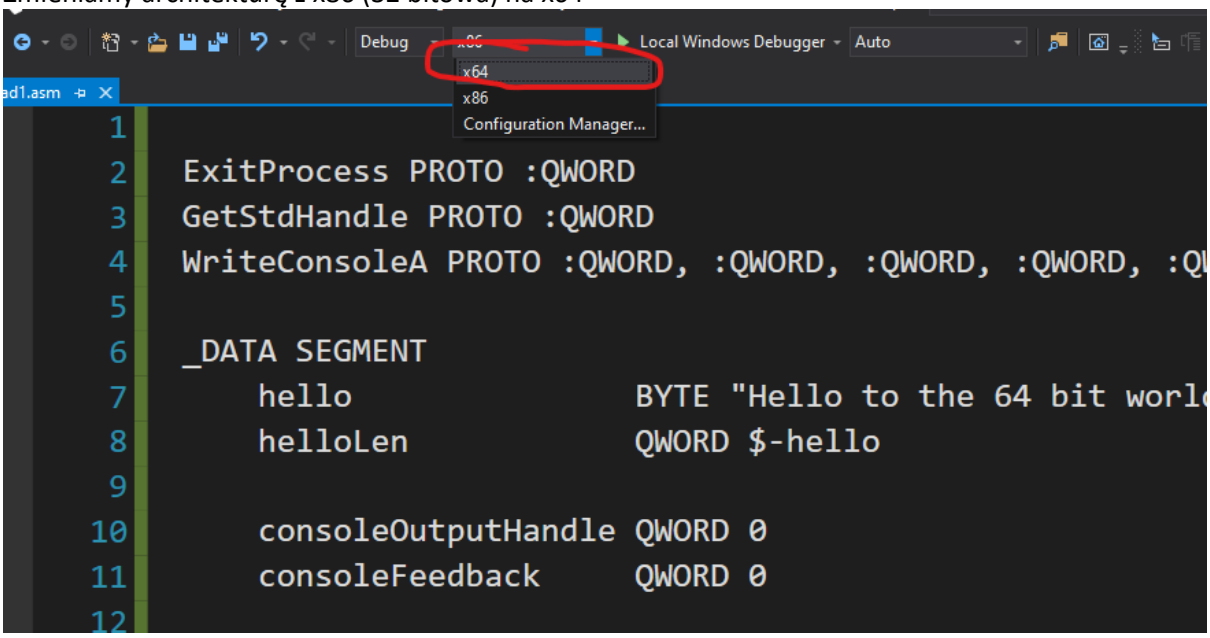
Rysunek 1: Tworzenie nowego projektu

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



Rysunek 2: Tworzenie nowego projektu

4. Zmieniamy architekturę z x86 (32 bitowa) na x64



Rysunek 3: Zmiana architektury

5. Ustawiamy klikając prawym przyciskiem na projekt **Build Customization** ->**Build Dependencies** i zaznaczamy **masm**
6. Ustawiamy odpowiednio linker i entry point na **main.**
7. Przykładowy kod programu:

```
ExitProcess PROTO :QWORD

_DATA SEGMENT
_DATA ENDS

_TEXT SEGMENT

main proc
    ; kolejnosc przekazywania argumentow od lewej
    ;RCX, RDX, R8, R9 potem wrzucamy na stos

    ;jesli chcemy przekazac 1 argument do procedury, uzywamy rejestru RCX np:
    mov RCX, 0

main endp

_TEXT ENDS
END
```

W architekturze 64 bitowej rozmiar wszystkich rejestrów zwiększył się odpowiednio z 32 bitów do 64 bitów. W celu przechowywania informacji pobieranych z tych rejestrów wprowadzony został nowy typ danych **QWORD** reprezentujący 64 bity (8 bajtów). Dodane zostało również 8 rejestrów roboczych R8 – R15. Nazwy wszystkich rejestrów zostały 64 bitowych zaczynają się od litery R, 64 bitowy odpowiednik rejestru EAX to RAX (E -> R).

Zmieniona również została domyślna konwencja przekazywania parametrów do procedur. W architekturze 64 bitowej wykorzystywane są do tego również rejestry:

```
func1(int a, int b, int c, int d, int e, int f);
// a w RCX, b w RDX, c w R8, d w R9, f potem e wrzucone na stos
```

5 Linki

1 Calling convention **- https://docs.microsoft.com/en-us/cpp/build/x64-calling-convention?view=msvc-160**