PROGRAMOWANIE OBIEKTOWE – LABORATORIUM

ĆWICZENIE 9 – KOMUNIKACJA SIECIOWA I

1. Cel ćwiczenia

Praktyczne zastosowanie umiejętności komunikacji sieciowej oraz umiejętności tworzenia, wysyłania i odbierania pakietów oraz wysyłania większych ilości danych.

2. Pojęcia

UDP – bezpołączeniowy protokół wymiany danych, nie wymagający nawiązywania połączenia i śledzenia sesji. Nie posiada mechanizmów kontroli przepływu i retransmisji. Nie gwarantuje dostarczenia danych. Ze względu na uproszczoną budowę zapewnia szybszą transmisję danych.

TCP – protokół wymiany danych przeznaczony do zastosowania w architekturze klient-serwer. Gwarantuje dostarczenie wszystkich pakietów z zachowaniem kolejności i bez duplikatów. Protokół zapewnia niezawodną wymianę danych kosztem zwiększenia ilości wysyłanych danych.

Gniazdo (ang. *Socket*) – pojęcie abstrakcyjne reprezentujące dwukierunkowy punkt końcowy połączenia. Dwukierunkowość oznacza możliwość wysyłania i odbierania danych. Gniazda wykorzystywane są przez aplikacje do komunikowania się przez sieć w ramach komunikacji międzyprocesowej.

Architektura klient-serwer

Architektura systemu komputerowego, w której strony komunikacji posiadają przydzielone role klienta (strona aktywna) lub serwera (strona pasywna). Zadaniem serwera jest aktywnie oczekiwanie (nasłuch) na próbujących się połączyć klientów, akceptacja ich i odpowiedź na komunikaty, wysyłane przez połączonych klientów. Zadaniem klienta jest inicjalizacja transmisji poprzez próbę połączenia z serwerem, a następnie nadawanie komunikatów i ewentualny odczyt odpowiedzi z serwera.

ASIO (ang. *ASynchronous Input/Output*) – międzyplatformowa (m.in. dla takich systemów operacyjnych jak Linux czy Windows) biblioteka C++ do asynchronicznego zarządzania interfejsami wejściowo-wyjściowymi, także sieciowymi. Jej główne cechy obejmują między innymi:

- Możliwość wykorzystania w różnych systemach operacyjnych,
- Wsparcie adresacji IPv4 i IPv6,
- Wsparcie przetwarzania asynchronicznego,
- Kompatybilność z biblioteką *iostream*,
- Wsparcie komunikacji sieciowej.

3. Instrukcja

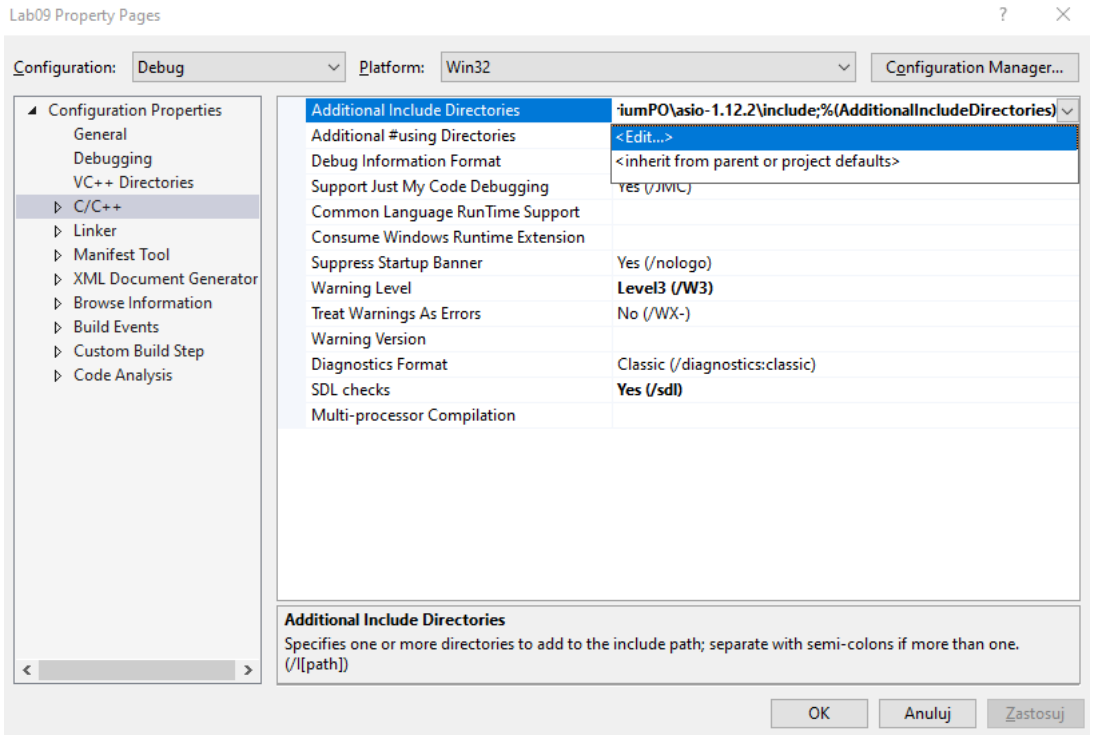
Instalacja asio

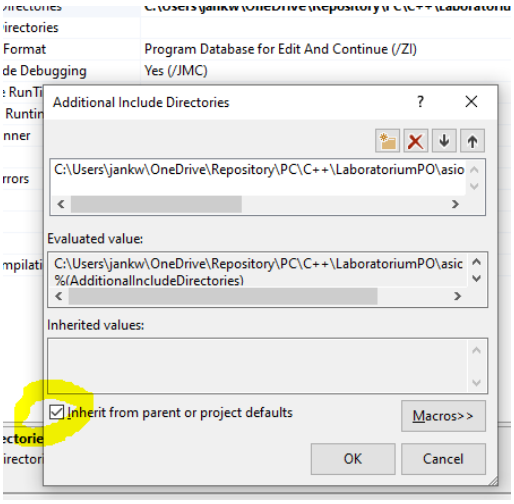
a. pobierz asio-x.yy.x.zip (najnowsza wersja) ze strony https://think-async.com/Asio/Download.html

b. Wypakuj pobrany plik.

c. We właściwościach projektu wybierz zakładkę C/C++ Directories. Jeżeli nie widzisz zakładki we właściwościach projektu, dodaj jakikolwiek plik .cpp do projektu (np. *main.cpp*)

d. Dodaj to *General → Additional Include Directories* ścieżkę do folderu *…\asio-1.12.2\include* (ścieżka, gdzie wypakowano plik .zip) **UWAGA**: w celu uniknięcia błędów można *Additional Include Directories* edytować za pomocą wyskakującego okna, które można otworzyć klikając na strzałkę po prawej stronie. Ponadto warto zaznaczyć checkbox *Inherit from parent or project defaults*.





e. Wybierz zakładkę *C/C++* → *Preprocessor*.

f. Dopisz do *Preprocessor Definitions* ASIO_STANDALONE. Warto ponownie zastosować okno *Edit…*, jak w podpunkcie d.

Serwer TCP

Napisz klasę *Server*, realizującą działanie asynchronicznego serwera. Klasa Server musi dziedziczyć po interfejsie *IThread* z laboratorium 7 oraz implementować metodę *ThreadRoutine*. Wątek po uruchomieniu ma rozpocząć nasłuch i akceptację przychodzących połączeń. Po ustanowieniu połączenia, dane mają być odebrane poprzez socket, a następnie odesłane do klienta (operacja echo, obsługiwana w osobnej funkcji `void session(asio::ip::tcp::socket socket)`). Serwer musi umożliwić realizację komunikacji kilku klientom jednocześnie, poprzez realizację wielowątkowej obsługi komunikacji. Nowy wątek musi być uruchamiany w ThreadRoutine i realizowany niezależnie (`std::thread(session, std::move(socket)).detach();`). Komunikacja musi być realizowana z zastosowaniem biblioteki Asio.
Konstruktor klasy *Server* musi przyjmować pojedynczy parametr, jakim jest numer portu, na którym nasłuchuje zgłoszeń klientów.

Klient TCP

Napisz klasę *Client*, zawierającą publiczną metodę:

```
void Echo(std::string text, const asio::ip::address_v4 ip, const unsigned short port);
```

Metoda *Echo* musi inicjalizować połączenie z serwerem o wybranym adresie i na wybranym porcie. Po ustanowieniu połączenia musi wysyłać dane tekstowe do serwera i odbierać odpowiedź (echo) z serwera.

Sprawozdanie – po drugiej części zajęć (ćwiczenie 10)