Zajęcia 1. Wprowadzenie

1. Zapoznanie się ze środowiskiem pracy

Praca w sieci lokalnej

Laboratorium prowadzone jest na komputerach wyposażonych w system operacyjny **Linux** z Oracle VM VirtualBox. Ponadto katalogi robocze są umieszczone na serwerze plikowym pracującym pod kontrolą systemu **Linux**. Aby przystąpić do zajęć laboratoryjnych, należy zalogować się do systemu Linux (użytkownik Student PRz bez hasła), a następnie uruchomić program Oracle VM VirtualBox Manager i wybrać maszynę wirtualną „**Lab_A300_VS_Community**", następnie wcisnąć ikonę „**Start**".

Sposób logowania

Aby się zalogować do systemu należy wykonać następujące czynności:

1. Wybrać: **Inny użytkownik** (w dolnym lewym rogu)
2. W miejscu **Nazwa użytkownika** należy podać domenę oraz identyfikator: **FD2lab*vxyz***, gdzie ***vx*** oznacza numer grupy laboratoryjnej; ***yz*** oznacza numer zespołu w grupie (przykładowy identyfikator **FD2lab0205 –** czyli L02, stanowisko nr 5)
3. W miejsce **Hasło** wpisujemy: student

Uwaga! Identyfikatory **FD2lab*vxyz*** zostaną nadane poszczególnym studentom przez prowadzącego zajęcia i są ważne do końca semestru, są one związane z danym studentem, a nie z konkretnym komputerem.

Katalogi robocze

Pliki robocze (źródłowe i pliki projektu) powinny się znaleźć w katalogach umieszczonych na serwerze plikowym na dysku **h:**.

Praca w środowisku Visual Studio

Aby uruchomić środowisko programistyczne należy przycisnąć ikonę **Visual Studio 2022**, ikona znajduje się na pulpicie, pasku zadań.

Tworzenie projektu

Aby utworzyć nowy projekt należy wybrać „Utwórz nowy projekt" lub po wybraniu opcji „kontynuuj bez kodu" („continue without code") należy wybrać **Plik|Nowy** (**File|New)** a następnie zakładkę **Projekt… (Project…)**.

Ponieważ program przedmiotu dotyczy tylko podstaw programowania obiektowego w języku C++, zatem na zajęciach będziemy korzystać tylko z jednego z dostępnych typ typów projektów czyli: **C++|Windows|Konsola**.(**C++|Windows|Konsola**), gdzie należy wybrać „**Pusty projekt**" (**Empty Project**) i nacisnąć klawisz **Dalej (Next)**. Następnie w polu **Nazwa Projektu** (**Project Name)** należy podać jego nazwę i wskazać lokalizację, w której ma być on utworzony (**…**).

Ten typ projektu zapewnia łatwą komunikację z urządzeniami zewnętrznymi i tworzy program działający w trybie „okienka konsolowego". W przypadku pomyłki w typie projektu zaleca się usunięcie projektu i utworzenie nowego wg podanych wyżej zasad. Ponadto zaleca się podawanie nazw projektów identyfikujących kolejne zajęcia. Pole z nazwą katalogu jest uaktualniane automatycznie (w razie potrzeby można zmienić miejsce składowania plików projektu). Następnie wskazujemy zawartość tworzonego projektu. Chcąc tworzyć własny program zaznaczamy zazwyczaj pusty projekt. Struktura plików automatycznie utworzonego projektu jest pokazana w okienku **Solution Explorer**. Po utworzeniu nowego projektu wybieramy menu lokalnego **Projekt|Dodaj nowy element (Project|Add New item)** tworzymy

nowy plik **Visual C++|Plik C++ (.cpp) (Visual C++|Code|C++ File (*.cpp)**. Następnie w polu **Nazwa: (Name:)** wpisujemy nazwę nowego pliku. Przełączenie widoku na **ClassView** pokaże strukturę klas, metod i funkcji (jeśli zostały zdefiniowane) w projekcie. Struktura klas jest tworzona na podstawie deklaracji w plikach nagłówkowych (**Header Files**).
Do projektu można wstawiać pliki już istniejące (źródłowe albo nagłówkowe lub inne) przez otworzenie menu lokalnego (naciśnięcie prawego klawisza myszki **Dodaj|Istniejący element (Add|Existing Item)**) dla konkretnego folderu (**Source Files**, **Header Files**) i wybranie pliku, który ma być dołączony.

Kompilacja

Narzędzia umożliwiające kompilację i konsolidację znajdują się w menu **Kompilowanie (Build)**. Poszczególne pliki źródłowe można kompilować z osobna poleceniem **Compile** (po uprzednim wskazaniu pliku) albo wybrać **Kompiluj rozwiązanie** (**Build Solution)**, aby przeprowadzić proces kompilacji (gdy nie ma modułów **obj**, bądź są przestarzałe) i konsolidacji albo wybrać **Kompiluj ponownie rozwiązanie** (**Rebuild Solution)**, aby skompilować ponownie wszystkie pliki źródłowe i przeprowadzić konsolidację. Alternatywnie można posłużyć się klawiszami skrótów lub paletą narzędziową **Build**
Wyniki kompilacji i konsolidacji (w szczególności lista błędów) są pokazywane w dolnym panelu (**Output**). W przypadku pojawienia się błędu można dwukrotnie kliknąć na linii z komunikatem o błędzie, co spowoduje przeniesienie kursora w panelu edycyjnym do odpowiedniej linii w celu skorygowania błędu.

Uruchamianie programu

Po utworzeniu programu exe można go uruchomić, aby zobaczyć efekty pracy przez wybranie polecenia **Debugowanie|Uruchom bez debugowania** (**Debug|Start Without Debugging)** albo klikając pusty zielony trójkąt w palecie **Debug**.
Rezultat działania zobaczymy w postaci „okienka konsolowego". Po zakończeniu działania proces jest wstrzymywany i czeka na reakcję użytkownika.
Wykonanie programu spowoduje także wybranie polecenia **Debugowanie|Uruchom bez debugowania (Debug|Start Debugging)** albo kliknięcie zielonego trójkąta z palety **Debug**. Porównaj działanie obu sposobów.

Praca krokowa

Praca krokowa polega na wstrzymywaniu działania programu po wykonaniu instrukcji zakodowanych w jednej linii. Rozpoczęcie tego trybu pracy odbywa się przez wybranie **Debug|Step Into** (albo **Run to cursor**). Wykonanie pojedynczej linii programu odbywa się przez **Debug|Step Over**. Wygodniejsze jest jednak korzystanie z palety uruchomieniowej **Debug**.
W czasie pracy krokowej dolny panel zawiera listę zmiennych (obiektów) tworzonych w trakcie działania programu umożliwiając śledzenie zmian ich wartości. Ponadto w panelu **Watch 1** możemy utworzyć własną listę zmiennych, których zawartości chcemy podglądać.
Zakończenie pracy krokowej i powrót do trybu edycji odbywa się przez wybranie polecenia **Debug|Stop Debugging** z menu albo z palety uruchomieniowej.

Opracowanie prostych programów w języku C/C++

Zdefiniuj funkcję `main`, w której opracuj i przetestuj następujące zadania:

- funkcję obliczającą silnię liczby,
- funkcję obliczającą pierwiastki równania kwadratowego,
- funkcję zwracającą *liczbę Fibonacciego* o zadanym numerze (np. nr=`1` zwróci 1, nr=`8` zwróci `21`),
- funkcję, która ma za zadanie wypisać na ekran dowolny napis przekazany do funkcji w taki sposób, że każda litera podanego napisu na ekranie ma pojawić się dwa razy

- funkcję posiadającą dwa argumenty. Pierwszym argumentem funkcji ma być napis, drugim ma być znak (`char`). Funkcja ma obliczyć liczbę wystąpień podanego do funkcji znaku w napisie przekazanym do funkcji i zwrócić tę liczbę jako wynik działania funkcji.
- funkcję, która ma za zadanie odwrócić ciąg liter napisu przekazanego do funkcji.
- funkcję sortującą rosnąco tablicę liczb typy `double` (tablicę należy przesłać jako argument funkcji)