Program szkolenia - Podstawy Programowania Aplikacji C/C++ (VS Code / Qt)

Miejsce realizacji	Fablab Chrzanów, ul. Janiny Woynarowskiej 1, 32-500 Chrzanów
Liczebność grupy	maksymalnie 6 osób
Forma wsparcia	warsztaty praktyczne przy stanowiskach komputerowych + wykłady interaktywne
Czas trwania	4 spotkania po 4 godziny (łącznie 16 godzin), raz w tygodniu
Cel główny	Zdobycie praktycznych umiejętności programowania w językach C i C++ z wykorzystaniem środowisk VS Code oraz Qt Creator, obejmujące pisanie, kompilowanie, debugowanie i testowanie aplikacji.

ETAP I – CHARAKTERYSTYKA SZKOLENIA

Grupa docelowa

Dorośli (18+) zainteresowani programowaniem systemów wbudowanych, aplikacji desktopowych i narzędziowych. Kurs skierowany do inżynierów mechanicznych, elektryków, automatyków oraz osób z innych branż technicznych, które chcą samodzielnie pisać lub modyfikować oprogramowanie. Zalecana znajomość podstaw obsługi komputera; znajomość jakiegokolwiek języka programowania jest atutem, ale nie jest wymagana.

Zakres tematyczny podlegający ocenie (kompetencja)

Podstawy programowania aplikacji C/C++ – obejmujące:

- konfigurację środowisk programistycznych VS Code i Qt Creator,
- fundamenty języka C: typy danych, zmienne, operatory, instrukcje sterujące, funkcje,
- zarządzanie pamięcią: wskaźniki, tablice, dynamiczna alokacja (malloc/new),
- programowanie obiektowe w C++: klasy, dziedziczenie, polimorfizm, szablony,
- budowę aplikacji GUI w Qt: sygnały i sloty, widgety, Qt Widgets,
- pracę z plikami i kontenerami STL (vector, map),
- pisanie testów jednostkowych i podstawy code review.

ETAP II – WZORZEC (EFEKTY UCZENIA SIĘ)

WIEDZA — uczestnik/uczestniczka wie:

- jaka jest historia i filozofia języków C i C++ – gdzie są stosowane i dlaczego
- czym różni się kompilacja od interpretacji – jak działa kompilator (gcc/g++) i linker
- jakie są podstawowe typy danych w C/C++: ich rozmiary, zakresy i konwersje
- co to są wskaźniki i jak zarządzanie pamięcią wpływa na stabilność i wydajność programu

Projekt „Utworzenie innowacyjnych przestrzeni typu Fablab na terenie Małopolski Zachodniej wraz z organizacją działań mobilnych ”dofinansowany w ramach priorytetu 8 Fundusze europejskie dla sprawiedliwej transformacji Małopolski Zachodniej 8.2 Edukacja dla transformacji, typ A: Tworzenie przestrzeni typu fablab programu Fundusze Europejskie dla Małopolski 2021-2027. Nr projektu: FEMP.08.02-IP.01-0017/23

- czym jest programowanie obiektowe (OOP): hermetyzacja, dziedziczenie, polimorfizm, abstrakcja
- jak działa mechanizm sygnałów i slotów w Qt i dlaczego jest bezpieczniejszy od callbacków
- czym jest STL (Standard Template Library) i jakie kontenery udostępnia (vector, list, map, set)
- co to są wyjątki (try/catch/throw) i jak odróżnić je od kodów błędów
- na czym polega cross-compilation i kiedy jest stosowana (np. dla systemów wbudowanych)
- czym jest test jednostkowy i jaką rolę odgrywa w procesie wytwarzania oprogramowania
- jak działa system budowania CMake – struktura pliku CMakeLists.txt

UMIEJĘTNOŚCI — uczestnik/uczestniczka potrafi:

- skonfigurować środowisko VS Code z kompilatorem gcc/g++, debuggerem GDB i narzędziami CMake
- napisać, skompilować i uruchomić program w języku C i C++ – obsługa błędów kompilacji
- zaimplementować algorytmy sterujące (pętle, warunki, rekurencja) i funkcje z parametrami
- dynamicznie alokować i zwalniać pamięć – unikać wycieków pamięci (Valgrind)
- zaprojektować i zaimplementować klasę C++ z konstruktorem, destruktorem i metodami
- zbudować prostą aplikację Qt Widgets z interfejsem GUI, sygnałami i slotami
- odczytywać i zapisywać dane do pliku tekstowego (fstream, QFile)
- korzystać z kontenerów STL: vector, map – dodawanie, usuwanie, iteracja, sortowanie
- napisać i uruchomić testy jednostkowe w Google Test lub Qt Test
- przeprowadzić code review i poprawić kod według standardów (styl, komentarze, dokumentacja)
- samodzielnie zrealizować mały projekt aplikacji konsolowej lub Qt

KOMPETENCJE SPOŁECZNE — uczestnik/uczestniczka:

- systematycznie debuguje kod: stosuje podejście hipoteza → weryfikacja → naprawa
- dba o czytelność i dokumentację kodu – stosuje nazewnictwo, komentarze i strukturę
- uczestniczy w code review: konstruktywnie przekazuje i przyjmuje uwagi techniczne
- planuje pracę nad projektem: dzieli zadanie na etapy, szacuje czas i priorytetyzuje
- korzysta z dokumentacji oficjalnej (cppreference.com, docs.qt.io) i zasobów społeczności
- rozumie znaczenie testowania dla jakości i niezawodności oprogramowania
- zwiększa samodzielność i pewność siebie w pisaniu, modyfikowaniu i debugowaniu kodu

TREŚCI PROGRAMOWE

Nr	Tytuł spotkania	Treści programowe
S1	**Środowisko pracy, fundamenty C i sterowanie przepływem**	Test początkowy wiedzy programistycznej Instalacja i konfiguracja VS Code z gcc/g++, CMake Tools i debuggerem GDB Struktura projektu: pliki .c/.cpp, .h, CMakeLists.txt; Hello World – kompilacja i uruchomienie Typy danych: int, float, double, char, bool – rozmiary, zakresy, konwersje Instrukcje warunkowe (if/else, switch-case) i pętle (for, while, do-while)

Projekt „Utworzenie innowacyjnych przestrzeni typu Fablab na terenie Małopolski Zachodniej wraz z organizacją działań mobilnych ”dofinansowany w ramach priorytetu 8 Fundusze europejskie dla sprawiedliwej transformacji Małopolski Zachodniej 8.2 Edukacja dla transformacji, typ A: Tworzenie przestrzeni typu fablab programu Fundusze Europejskie dla Małopolski 2021-2027. Nr projektu: FEMP.08.02-IP.01-0017/23

		Funkcje: deklaracja, definicja, parametry przez wartość i wskaźnik Debugger VS Code: breakpoints, watch, step over/into Ćwiczenie: implementacja kalkulatora z menu wyboru operacji
S2	**Wskaźniki, pamięć i programowanie obiektowe C++**	Tablice i wskaźniki – deklaracja, dereferencja, arytmetyka wskaźnikowa Dynamiczna alokacja: malloc/free (C) i new/delete (C++) – unikanie wycieków (Valgrind) Klasy i obiekty: pola, metody, konstruktory, destruktory, hermetyzacja Dziedziczenie i polimorfizm – metody wirtualne, override Szablony (templates) – funkcje i klasy generyczne Ćwiczenie: implementacja klasy Stack z dynamiczną tablicą; test dziedziczenia
S3	**Qt GUI, STL i praca z plikami**	Instalacja Qt Creator; tworzenie projektu Qt Widgets; sygnały i sloty Budowa prostego interfejsu GUI: QPushButton, QLineEdit, QLabel, QMessageBox Biblioteka STL: vector, map – dodawanie, usuwanie, iteracja, sort Operacje na plikach: fstream (C++) i QFile (Qt) – zapis i odczyt CSV Obsługa błędów: kody zwrotne i wyjątki (try/catch/throw) Ćwiczenie: aplikacja Qt – lista zadań z zapisem/odczytem pliku CSV
S4	**Projekt końcowy, testy jednostkowe i walidacja**	Samodzielna realizacja projektu: mała aplikacja Qt lub konsolowa C++ wg specyfikacji Testy jednostkowe: Google Test lub Qt Test – pisanie i uruchamianie testów (min. 5 testów) Code review – wzajemna inspekcja kodu: styl, komentarze, dokumentacja Test końcowy (post-test) – wiedza teoretyczna i analiza fragmentu kodu Prezentacja projektu końcowego i ocena praktyczna przez prowadzącego Samoocena uczestnika i podsumowanie nabytych kompetencji

ETAP III – KRYTERIA I METODY WERYFIKACJI EFEKTÓW UCZENIA SIĘ

Metody weryfikacji teoretycznej

- Test końcowy (post-test) – typy danych, wskaźniki, OOP, STL, Qt, testowanie
- Pytania ustne podczas sesji ćwiczeniowych – analiza i wyjaśnienie działania kodu
- Ocena jakości kodu projektu: styl, komentarze, struktura plików

Metody weryfikacji praktycznej

- Poprawna implementacja ćwiczeń z każdego spotkania – kompilacja bez błędów
- Projekt końcowy: działająca aplikacja Qt lub konsolowa C++ zgodna ze specyfikacją

Projekt „Utworzenie innowacyjnych przestrzeni typu Fablab na terenie Małopolski Zachodniej wraz z organizacją działań mobilnych ”dofinansowany w ramach priorytetu 8 Fundusze europejskie dla sprawiedliwej transformacji Małopolski Zachodniej 8.2 Edukacja dla transformacji, typ A: Tworzenie przestrzeni typu fablab programu Fundusze Europejskie dla Małopolski 2021-2027. Nr projektu: FEMP.08.02-IP.01-0017/23

- Testy jednostkowe projektu: min. 5 testów przechodzących (green)
- Ocena code review: min. 3 konstruktywne uwagi do kodu innego uczestnika
- Samoocena uczestnika

Uczestnik nabywa kompetencje, jeśli:

Kryterium	Wymagany poziom
Post-test wiedzy teoretycznej	min. 80% poprawnych odpowiedzi
Projekt końcowy – kompilacja i działanie	aplikacja kompiluje się i przechodzi test funkcjonalny
Testy jednostkowe	min. 5 testów zielonych (passing)
Code review	min. 3 merytoryczne uwagi do kodu kolegi
Obecność	min. 80% godzin zajęciowych (min. 13 z 16 h)