

Kod szkolenia: **C/ADVA**

Tytuł szkolenia: **Zaawansowane programowanie w języku C**

Dni: 3

Opis:

Adresaci szkolenia

Szkolenie adresowane jest do osób znających język C i chcących rozszerzyć wiedzę z tego zakresu.

Cel szkolenia

Uczestnicy szkolenia zapoznają się z zaawansowanymi aspektami programowania w języku C. Poruszona zostanie tematyka między innymi związana z tworzeniem aplikacji na architektury 32 i 64 bitowe, bezpiecznym zarządzaniem pamięcią oraz z zakresu tworzenia i testowania aplikacji wielowątkowych. Szkolenie oparte jest o aktualny standard języka C11.

Mocne strony szkolenia

Podczas szkolenia uczestnicy:

- wykonają wiele praktycznych zadań, które zobrazują poruszane problemy implementacyjne,
- zapoznają się z aspektami związanymi z profilowaniem i optymalizacją aplikacji w języku C.

Wymagania

Od uczestników szkolenia wymagana jest umiejętność programowania w języku C.

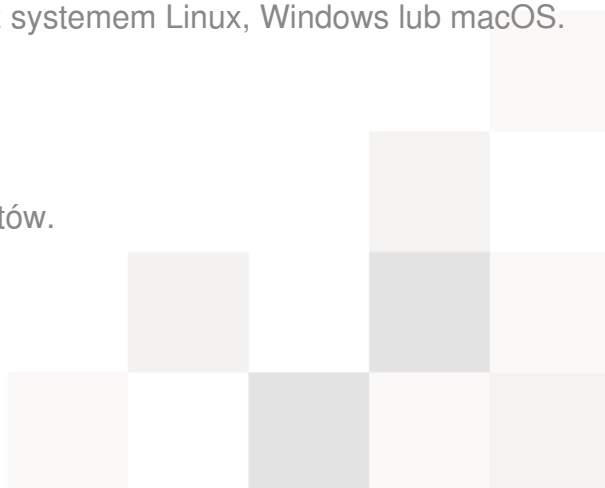
Specjalne wymagania techniczne

Uczestnicy w trakcie zajęć korzystają z komputera z systemem Linux, Windows lub macOS. Zalecany jest system Linux.

Parametry szkolenia

3 * 8 godzin (3 * 7 godzin netto) wykładów i warsztatów.

Program szkolenia:



1. Algorytmy i iteratory biblioteki standardowej.
 - Przegląd dostępnych algorytmów.
 - Wydajność i aspekty implementacji.
 - Praktyczna implementacja z użyciem algorytmów z biblioteki standardowej.
 - Implementacja własnych algorytmów do ponownego wykorzystania.
 - Złożoność obliczeniowa i zakres zastosowań.
 - Warsztaty wykorzystania algorytmów.
 - Rodzaje iteratorów dostępnych w bibliotece standardowej.
2. Iteratory i programowanie uogólnione.
 - Aspekty implementacji iteratorów.
 - Implementacja własnych iteratorów.
 - Programowanie uogólnione – koncepcje i zastosowania.
3. Szablony STL – wydajne stosowanie i wstęp do metaprogramowania.
 - Wzorce – pułapki i zastosowania.
 - Konkretyzacja – reguły i działanie.
 - Cechy typów.
 - Metaprogramowanie – podstawy.
 - Algorytmy na poziomie meta.
4. Wzorce projektowe.
 - Wprowadzenie wzorców projektowych.
 - Strukturalne wzorce projektowe.
 - Konstrukcyjne wzorce projektowe.
 - Behawioralne wzorce projektowe.
 - Specyfika implementacji wzorców projektowych w języku C++.
5. Użyteczne biblioteki
 - Boost – przegląd użytecznych rozwiązań.
 - Biblioteki rozwijające paradygmat programowania uogólnionego.
 - Biblioteki przetwarzania danych.
 - ML i AI w języku C++ - przydatne biblioteki
6. Testowanie i metodyka testowania.
 - Proces TDD w pracy programisty.
 - Proces BDD i różnice w stosunku do TDD
 - Rodzaje bibliotek testów i dostępne asercje.
 - Projektowanie i implementowanie testów.
7. Praca z narzędziem debuggera w języku C++.
 - Dostęp do podstawowych struktur C++.
 - Automatyzacja i skrypty.
 - Wstęp do integracji z infrastrukturą testowania.
8. Programowanie wielowątkowe.
 - Wyzwania i problemy.
 - Współpraca wątków.
 - Inne biblioteki obsługi wielowątkowości (Boost Threads, Intel Threading Building Blocks)
 - Prymitywy synchronizujące.
 - Sekcje krytyczne.
 - Komunikacja i dane współdzielone w aplikacjach wielowątkowych

- Testowanie aplikacji wielowątkowych.
- 9. Wzorce wielowątkowości.
 - Wzorce wielowątkowości – przegląd.
 - Implementacja najbardziej użytecznych wzorców wielowątkowości.
 - Zaawansowane wzorce wielowątkowe.
- 10. Nowe standardy C++.
 - Przegląd nowości wprowadzonych w nowych standardach C++.
 - Zarządzanie pamięcią (wskaźniki inteligentne).
 - Nowe aspekty składni.
 - Obiekty funkcyjne (typ `std::function`, funkcje lambda, łączenie obiektów)
 - Zarządzanie zakresami.
 - Ujednoliconą obsługę systemu plików i czasu.
 - Cechy typów oraz rozszerzenia wielowątkowości (implementacja typów atomowych).
- 11. Wykładnie i nowoczesne projektowanie aplikacji.
 - Zasady S.O.L.I.D. , GRASP, YAGNI - przegląd i przypomnienie.
 - Wykładnie w programowaniu obiektowym - aspekt praktyczny
 - Implementacja wykładni (ang. policy) we własnych aplikacjach.
 - Rola polimorfizmu statycznego i technik wykonania na etapie kompilacji.
- 12. Tworzenie aplikacji niezawodnych.
 - Aspekty wydajności obsługi wyjątków.
 - Tworzenie aplikacji odpornej na błędy.
 - Programowanie aspektowe.

