

## Program szkolenia:

# Narzędzia wspomagające pracę w złożonych projektach.

## Informacje:

<b>Nazwa:</b>	<b>Narzędzia wspomagające pracę w złożonych projektach.</b>
<b>Kod:</b>	<b>ccpp-C++ Tools</b>
<b>Kategoria:</b>	C i C++
<b>Odbiorcy:</b>	developerzy
<b>Czas trwania:</b>	3 dni
<b>Forma:</b>	30% wykłady / 70% warsztaty

---

Szkolenie przedstawia zbiór narzędzi i dobrych praktyk w znacznym stopniu usprawniających pracę w dużych i małych projektach. Sprawność podnoszona jest w dwóch wymiarach: szybkości pracy z kodem oraz jakości i poprawności dostarczanego kodu.

Uczestnicy zdobywają wiedzę w oparciu o zadania co umożliwia właściwie ntachmiastowe wykorzystanie omawianych elementów w codziennej pracy.

Szkolenie przeznaczone jest dla programistów (głównie C++) i integratorów pracujących w projektach o dużej złożoności. Zdobyta wiedza przekłada się w bezpośredni sposób na komfort pracy z kodem, jakość dostarczanego kodu oraz ciągły nadzór nad poprawnością i sprawnością kodu.

## Zalety szkolenia:

- Szkolenie kładzie szczególny nacisk na jakość i poprawność kodu
- Zdobyta wiedza może zostać wdrożona właściwie „od ręki”
- Nauka w oparciu o przykłady

## Szczegółowy program:

### 1. Build system – poprawna organizacja środowiska budowania

1.1. Repozytorium

1.2. Zależności pomiędzy komponentami

1.3. Makefile/CMake/Autotools (do wyboru)

1.3.1. Podstawowa składnia

1.3.2. Definiowanie reguł prostych i automatycznych

1.3.3. Definiowanie zależności „małych” i „dużych”

1.3.4. Clean lokalny i całosciowy

### 2. CI – Continuous Integration – czym jest i jak używać

2.1. Przedstawienie koncepcji CI

2.1.1. Co to jest CI: cechy i korzyści

2.1.2. Znaczenie CI w dużych i małych projektach

2.1.2.1. Ciągła integracja jako kompleksowy proces

2.2. Jenkins overview

2.2.1. Jenkins a Hudson

2.2.2. Wymagania

2.2.3. Instalacja i konfiguracja

2.2.4. Dostęp i uprawnienia

2.2.5. Dobre praktyki i ciekawostki

2.2.6. Kopie zapasowe

2.2.7. Narzędzia towarzyszące

2.2.8. Pluginy

2.2.8.1. Zestaw absolutnie podstawowy

## 2.2.8.2. Pluginy rozwiązujące konkretne problemy

### 3. Testowanie kodu C++

#### 3.1. GTest framework

3.1.1. API – podstawowe API testów jednostkowych/modułowych

3.1.2. GMock – jak szybko i skutecznie implementować zaśllepki

3.1.3. LCOV – badanie pokrycia testowanego kodu

#### 3.2. Integracja testów w CI

### 4. Statyczna analiza kodu

4.1. Pojęcie i znaczenie statycznej analizy kodu

4.2. Coding rules – znaczenie zbioru zasad i dobrych praktyk na rzecz poprawności i jednolitości kodu

4.3. Cppcheck – automat do sprawdzania poprawności kodu bez jego uruchamiania

4.4. CPD – wszystko na temat powielania tego samego kodu

4.5. CCCC – „lokalny” stopień złożoności kodu

4.6. Integracja narzędzi statycznej analizy kodu w CI

4.7. Code review – obowiązkowa część procesu dostarczania kodu