BO·TT·EGA
IT minds

Program szkolenia:

Jakość w C++

Informacje:

Nazwa:	**Jakość w C++**
Kod:	**ccpp-quality**
Kategoria:	C i C++
Odbiorcy:	developerzy
Czas trwania:	3 dni
Forma:	50% wykłady / 50% warsztaty

Zalety szkolenia:

- Kompleksowe podejście do szeroko rozumianej jakości
- Nauka w oparciu o rzeczywiste przykłady
- Nacisk na pisanie poprawnego, czystego i wydajnego kodu

BOTTEGA IT minds

Szczegółowy program:

1. Wstęp: quality – czy to tylko testowanie?

- 1.1. Czym jest kompleksowe podejście do szeroko rozumianej jakości?
  - 1.1.1. Typowe problemy natury technicznej, biznesowej i personalnej
  - 1.1.2. Praca nad wymaganiami przyrastającymi w czasie i konieczność regresyjnego podejścia do jakości
- 1.2. Pierwsze rozmyślania architektoniczne: złożoność esencjonalna i przypadkowa
- 1.3. Co i jak testować
  - 1.3.1. Uporządkowanie tematyki testowania: koncept (odwróconej) piramidy testów
  - 1.3.2. V-model ze szczególnym naciskiem na fundament: „coding"
  - 1.3.3. Metryki jakości kodu, czyli od czego zacząć by zacząć dbać o jakość?

2. Przystanek 1: proces budowania kodu

- 2.1. Jak powinien budować się kod by mu ufać?
- 2.2. Flagi kompilacji absolutnie wymagane
- 2.3. Unifikacja i automatyzacja budowania wszystkich części kodu na tzw. „jedną modłę”

3. Jak pisać kod by móc go testować?

- 3.1. TDD = Test Driven Development
  - 3.1.1. Czym jest?
  - 3.1.2. Kiedy (nie) stosować TDD?
  - 3.1.3. Jak pracować z kodem „legacy”, czyli TDR = Test Driven Refactoring
  - 3.1.4. Co trzeba wiedzieć by móc z powodzeniem stosować te metody?
- 3.2. Architektura
  - 3.2.1. Architektura systemowa vs aplikacyjna
  - 3.2.2. Przypisywanie odpowiedzialności obiektom – metoda dziel i zwyciężaj
  - 3.2.3. Jak poprawnie interpretować OOP by (z czasem) nie powstał legacy code?

BOTTEGA IT minds

3.2.4. Inwersja kontroli i wstrzykiwanie zależności

3.2.5. Systemy sterowane zdarzeniami vs sekwencyjne

3.2.6. Architektura pluginowa

4. Testy deweloperskie po raz pierwszy: Testy Jednostkowe = UT = Unit Test

4.1. UT vs (odwrócona) piramida testów

4.2. Kontekst i środowisko uruchomieniowe UT

4.3. Z czego powinien być zbudowany dobry testy, czyli odpowiedź na pytanie co powinien zawierać test?

4.3.1. Obiekt i jego wymagania – granica testu

4.3.2. Ustalenie i gromadzenie danych wejściowych

4.3.3. Ustawienie poprawność sprawdzeń

4.4. Framework i jego możliwości na rzecz automatycznych testów

4.5. Zaślepianie zależności: dummy, stub, mock

4.6. „Czas to pieniądz”, czyli implementacja inteligentnych zaślepek z wykorzystaniem framework-ów

4.7. Badanie pokrycia kodu testami

4.7.1. Narzędzia i automatyzacja procesu

4.7.2. Typowe problemy: statystyki celem samym w sobie, lawinowe zmiany testów przy drobnych zmianach...

5. Testy developerskie po raz drugi: Testy Modułowe = MT = Module Test

5.1. Czym jest test modułowy?

5.2. UT vs MT vs (odwrócona) piramida testów

5.3. Jak zaprojektować MT?

5.4. Zalety MT w szczególności przy pracy z kodem „legacy”

5.5. UT, MT vs KPI(s) stawiane przez biznes, czyli odpowiedź na pytanie jak wyważyć mus zaspokojenia statystyk biznesowych i jakości testów w szerokim horyzoncie czasu

5.6. Automatyzacja UT i MT na rzecz wspólnych statystyk

BOTTEGA IT minds

6. Pozostałe metryki jakości

- 6.1. Statyczna analiza kodu
  - 6.1.1. CPPCheck
  - 6.1.2. Clang Static Analyzer
  - 6.1.3. Automatyzacja
- 6.2. Złożoność cyklometryczna
  - 6.2.1. Co to jest i jakie ma znaczenie, szczególnie przy pracy z kodem „legacy”
  - 6.2.2. Narzędzie CCCC
  - 6.2.3. Automatyzacja
- 6.3. CPD – wykrywanie metody „Copiego i Pasta”
  - 6.3.1. Narzędzie PMD: CPD
  - 6.3.2. Znacznie CPD dla dobrego startu procesu refaktoryzacji kodu
  - 6.3.3. Automatyzacja

7. Refaktoryzacja

- 7.1. Czym (nie) jest refaktoryzacja?
- 7.2. Kiedy wolno i kiedy nie wolno refaktoryzować?
- 7.3. Jak ją dobrze wykonać?
- 7.4. Jak przekonać „górę”, że to się opłaci?
- 7.5. Moc narzędzi

8. Narzędzia do zaawansowanej diagnostyki kodu C/C++

- 8.1. Debugger tekstowy GDB
  - 8.1.1. Rola GDB w procesie dbania o jakość ze szczególnym uwzględnieniem MT
  - 8.1.2. Jak z tym wystartować?
  - 8.1.3. Debugging zdalny

BO·TT·EGA
IT minds

8.1.4. Analiza plików coredump i ich generowanie (gcore) z uwzględnieniem problemów wynikających ze współbieżności

8.2. Badanie wycieków pamięci: Valgrind: Memcheck

8.2.1. Czym są wycieki pamięci i dlaczego są takie groźne?

8.2.2. Podstawowe użycie Memcheck-a i analiza wyników

8.2.3. Automatyzacja wywołań Memcheck-a i rola MT

8.3. Analiza wydajności: Valgrind: Callgrind

8.3.1. Co to jest profileing i jaka jest jego rola w procesie dbania o jakość ze szczególnym naciskiem na prace refaktoryzacyjne oraz MT?

8.3.2. Podstawowe użycie Callgrind-a i analiza wyników

8.3.3. Automatyzacja wywołań Callgrind-a i rola MT

8.4. Analiza wielowątkowości: Valgrind: Helgrind

8.4.1. Typowe problemy z wielowątkowością: wyścig, dead-lock

8.4.2. Podstawowe użycie Helgrind-a i analiza wyników

8.4.3. Automatyzacja wywołań Helgrinda-a i rola MTWstęp: quality – czy to tylko testowanie?