Politechnika Warszawska

KARTA PRZEDMIOTU OFEROWANEGO W SZKOLE DOKTORSKIEJ

Kod przedmiotu	4606-PS-00000BC-0024	Nazwa przedmiotu	w j. polskim	**Język C++ w zaawansowanych pracach badawczo-rozwojowych (CPBR)**
			w j. angielskim	**C++ language in advanced research and developmend**
Przynależność do grupy przedmiotów	specjalnościowe			
Koordynator przedmiotu	Dr hab. inż. Mateusz Malanowski (prowadzący dr. inż. Marcin Bączyk)			
Jednostka realizująca	WEiTI	Dyscyplina/y naukowa*		
Poziom kształcenia	Kształcenie doktorantów	Semestr	zimowy~~/letni~~	
Język zajęć	polski/~~angielski~~			
Forma zaliczenia:	zaliczenie na ocenę	Sumaryczna liczba godzin w semestrze	45	Sumaryczna liczba ECTS: 3
Minimalna liczba uczestników	10	Maksymalna liczba uczestników	15	Dostępność dla studentów: Tak/~~Nie~~

Typ zajęć		Wykład	Ćwiczenia audytoryjne	Ćwiczenia projektowe	Laboratorium	Seminarium
Liczba godzin zajęć	tygodniowo	2			1	
	łącznie w semestrze	30			15	

* nie dotyczy warsztatu badacza

1. Wymagania wstępne

Od uczestnika kursu oczekuje się, że:

- zna język C++ i potrafi w nim tworzyć średnio rozbudowane aplikacje;
- potrafi wytwarzać kod, zarówno w strukturalnym, jak i obiektowym paradygmacie programowania;
- wie czym są podstawowe wzorce projektowe i jak ich używać;

2. Cele przedmiotu

Celem przedmiotu jest zaznajomienie słuchaczy z narzędziem jakim jest język C++ w kontekście wykorzystania go w zaawansowanych bracach badawczo-rozwojowych. Po uzupełnieniu wiadomości z zakresu podstaw współczesnego standardu języka C++, omawiane są zagadnienia związane z mierzeniem i optymalizacją wydajności poprzez zastosowanie nowoczesnych struktur języka C++, testowaniem poprawności implementacji oraz wytwarzaniem oprogramowania dużej skali.

W trakcie wykładu i zajęć laboratoryjnych duży uwagi poświęca się na indywidualne podejście do potrzeb doktorantów i dostosowanie treści wykładu do specyfiki problemów rozwiązywanych w ramach prac badawczych.

3. Treści programowe (dla każdego typu zajęć oddzielnie)

Wykład

Wykład odbywa się w laboratorium komputerowym. Słuchacze kursu, uczestnicząc w wykładzie jednocześnie sprawdzają omawiane zagadnienia na przygotowanych wcześniej fragmentach kodów.

1. Wprowadzenie do współczesnego języka C++
2. Programowanie generyczne z wykorzystaniem szablonów
3. Programowanie funkcyjne w języku C++
4. Wzorce projektowe w języku C++
5. Narzędzia budowania
6. Testowanie kodu
7. Ciągła integrcja i ciągłe wdrażanie
8. Projektowanie interfejsów API
9. Wydajność
10. Tworzenie aplikacji współbieżnych.
11. Komunikacja sieciowa i przetwarzanie równoległe.

Laboratorium
Zajęcia laboratoryjne odbywają się razem z częścią wykładową. Pod koniec zajęć słuchacze dostają do wykonania, krótkie zadanie wybrane przez prowadzącego lub dostosowane do pracy badawczej słuchacza przedmiotu.

4. Efekty uczenia się			
Rodzaj efektu	Opis efektu uczenia się	Odniesienie do efektów uczenia się w SD PW	Sposób weryfikacji efektów uczenia*
Wiedza			
W01	*Ma uporządkowaną wiedzę na temat współczesnego języka C++*	SD_W2 SD_W3	ocena projektu
W02	*Wie w jaki sposób tworzyć złożone projekty pisane w języku C++*	SD_W2 SD_W3	ocena projektu
W03	*Wie jakie są współczesne trędy dotyczące wytwarzania oprogramowania w C++*	SD_W2 SD_W3	ocena projektu
Umiejętności			
U01	*Potrafi rozwiązać złożony problem programistyczny w postaci programu napisanego w języku C++*	SD_U1 SD_U7	ocena projektu
U02	*Potrafi w ramach projektu efektywnie wykorzystywać różne konstrukcje języka C++*	SD_U1	ocena projektu
U03	*Potrafi tworzyć niezawodny i efektywny obliczeniowo kod w języku C++*	SD_U1	ocena projektu
Kompetencje społeczne			
K01	*Potrafi pracować w grupie oraz wspólnie rozwiązywać problemy*	SD_K2	ocena prezentacji
K02	*Jest odpowiedzialny za własny kod, podejmuje zobowiązania, z których jest wstanie się wywiązać*	SD_K2	ocena prezentacji
K03	*Potrafi zarządzać swoim czasem oraz ma umiejętność planowania w czasie pracy własnej oraz pracy członków zespołu i dotrzymuje terminów*	SD_K2	ocena prezentacji
K04	*Zna i stosuje podstawowe metodyki zespołowego wytwarzania oprogramowania*	SD_K2	ocena prezentacji

* dozwolone sposoby weryfikacji efektów uczenia się: egzamin; egzamin ustny; kolokwium pisemne; kolokwium ustne; ocena projektu; ocena sprawozdania; ocena raportu; ocena prezentacji; ocena aktywności w trakcie zajęć; prace domowe; test

5. Kryteria oceny
Laboratorium 100 punktów: 10 mini zadań programistycznych ocenianych w skali 0-10 punktów. Tabela ocen: < 51 – ocena 2 51-60 – ocena 3 61-70 – ocena 3.5 71-80 – ocena 4

81-90 – ocena 4.5 91-100 – ocena 5.0

6. Literatura
Literatura podstawowa: [1] Stephan Roth „Czysty kod w C++17. Oprogramowanie łatwe w utrzymaniu” [2] Ivan Čukić „Programowanie funkcyjne w języku C++. Tworzenie lepszych aplikacji” [3] Scott Meyers „C++. 50 efektywnych sposobów na udoskonalenie Twoich programów” Literatura uzupełniająca: [1] Gazihan Alankus, Olena Lizina, Rakesh Mane, Vivek Nagarajan, Brian Price „Advanced C++” [2] Alexandru Bolboaca „Hands-On Functional Programming with C++” [3] Adrian Ostrowski, Piotr Gaczkowski „Architektura oprogramowania bez tajemnic. Wykorzystaj język C++ do tworzenia wydajnych aplikacji i systemów”

7. Nakład pracy doktoranta niezbędny do osiągnięcia efektów uczenia się**		
Lp.	Opis	Liczba godzin
1	godziny kontaktowe z nauczycielem akademickim wynikające z planu	45
2	Godziny kontaktowe z nauczycielem akademickim w ramach konsultacji, egzaminów, sprawdzianów itp.	5
3	Godziny pracy samodzielnej doktoranta w ramach przygotowania do zajęć oraz opracowania sprawozdań, projektów, prezentacji, raportów, prac domowych	40
4	godziny pracy samodzielnej doktoranta w ramach przygotowania do egzaminu, sprawdzianu, zaliczenia	0
	Sumaryczny nakład pracy doktoranta	**90**
	Liczba punktów ECTS	**3**

** 1 ECTS pracy = 25-30 godzin nakładu pracy doktoranta (np. 2 ECTS = 60 godzin; 4 ECTS = 110 godzin