

KARTA OPISU MODUŁU KSZTAŁCENIA		
Nazwa modułu/przedmiotu Podstawy programowania		Kod 1010531121010541915
Kierunek studiów Automatyka i robotyka	Profil kształcenia (ogólnoakademicki, praktyczny) ogólnoakademicki	Rok / Semestr 1 / 2
Ścieżka obieralności/specjalność -	Przedmiot oferowany w języku: polski	Kurs (obligatoryjny/obieralny) obligatoryjny
Stopień studiów: I stopień	Forma studiów (stacjonarna/niestacjonarna) stacjonarna	
Godziny Wykłady: 30 Ćwiczenia: - Laboratoria: 30 Projekty/seminaria: -	Liczba punktów 5	
Status przedmiotu w programie studiów (podstawowy, kierunkowy, inny) (ogólnouczelniany, z innego kierunku) kierunkowy z danego kierunku		
Obszar(y) kształcenia i dziedzina(y) nauki i sztuki nauki techniczne		Podział ECTS (liczba i %) 5 100%
Odpowiedzialny za przedmiot / wykładowca: dr inż. Jakub Kołota email: Jakub.Kolota@put.poznan.pl tel. 61 6652751 Katedra Inżynierii Komputerowej PP ul. Piotrowo 3A, 60-965 Poznań		
Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:		
1	Wiedza:	Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z zakresu programowania strukturalnego oraz sprzętu komputerowego i jego obsługi.
2	Umiejętności:	Powinien posiadać umiejętność rozwiązywania podstawowych problemów w obszarze modelowania algorytmów, programowania funkcyjnego oraz umiejętność pozyskiwania informacji ze wskazanych źródeł. Powinien również rozumieć konieczność poszerzania swoich kompetencji jak również być gotowym do podjęcia współpracy w ramach zespołu.
3	Kompetencje społeczne	Ponadto w zakresie kompetencji społecznych student musi prezentować takie postawy jak uczciwość, odpowiedzialność, wytrwałość, ciekawość poznawcza, kreatywność, kultura osobista, szacunek dla innych ludzi.
Cel przedmiotu: Cel modułu kształcenia: 1. Zapoznanie z metodologią i zasadami programowania obiektowego wykorzystując język programowania C++. 2. Rozwijanie u studentów umiejętności rozwiązywania problemów w obszarze modelowania i implementacji systemów informatycznych. Studenci uczą się przeprowadzać symulację i analizę działania obiektowych programów informatycznych oraz planować i dokumentować wykonaną pracę informatyczną. 3. Kształtowanie u studentów umiejętności programistycznych. Kreowanie świadomości konieczności profesjonalnego podejścia do zagadnień technicznych, skrupulatnego zapoznania się z dokumentacją systemów informatycznych typu UML. Student uczy się wyznaczać cele i określać priorytety prowadzące do realizacji zadania poprzez obiektową implementację kodu.		
Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia		
Wiedza: 1. Ma uporządkowaną wiedzę w zakresie wybranych algorytmów i struktur danych oraz metodyki i technik programowania proceduralnego i obiektowego; - [K_W8]		
Umiejętności: 1. potrafi skonstruować algorytm dla prostego zadania inżynierskiego oraz zaimplementować, przetestować i uruchomić go w wybranym środowisku programistycznym na komputerze klasy PC dla wybranych systemów operacyjnych; - [K_U26] 2. potrafi analizować i symulować działanie algorytmów, dobierając struktury danych do pożądanej funkcjonalności kodu - []		
Kompetencje społeczne: 1. rozumie potrzebę i zna możliwości ciągłego dokształcania się ? podnoszenia kompetencji zawodowych, osobistych i społecznych - [K_K1]		

Sposoby sprawdzenia efektów kształcenia

Efekty kształcenia przedstawione wyżej weryfikowane są w następujący sposób:

Ocena formująca:

a) w zakresie wykładów:

na podstawie odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach,

b) w zakresie laboratoriów:

na podstawie oceny bieżącego postępu realizacji zadań,

Ocena podsumowująca:

a) w zakresie wykładów weryfikowanie założonych efektów kształcenia realizowane jest przez:

a. na podstawie odpowiedzi na pytania dotyczące materiału omówionego na wykładach (implementacja kodu weryfikowana na komputerze klasy PC w środowisku programistycznym CodeBlocks). Egzamin obejmuje indywidualne podejście do każdego egzaminowanego studenta i analizę kodu napisanego podczas egzaminu będącego obiektywnym rozwiązaniem powierzonego zadania;

b. ocenę wiedzy i umiejętności wykazanych przez studenta podczas egzaminu praktycznego przy komputerze. Student otrzymuje zadanie o złożoności pozwalającej zweryfikować nabytą wiedzę w zakresie 0-20 pkt. Egzamin polega na zadaniu pytań z zakresu treści tematycznych omawianych na wykładzie i napisaniu przez studenta w języku C++ rozwiązania obiektowego.

ocena 2.0 10 i mniej punktów; ocena 3.0 11-12 punktów; ocena 3.5 13-14 punktów; ocena 4.0 15-16 punktów;

ocena 4.5 17-18 punktów; ocena 5.0 19-20 punktów

b) w zakresie laboratoriów weryfikowanie założonych efektów kształcenia realizowane jest przez:

a. dwa sprawdziany praktyczne przy komputerze (po 10 pkt każde)

b. Ocena podsumowująca wystawiana jest jako suma punktów z obu sprawdzianów wg schematu poniżej:

ocena 2.0 10 i mniej punktów; ocena 3.0 11-12 punktów; ocena 3.5 13-14 punktów; ocena 4.0 15-16 punktów;

ocena 4.5 17-18 punktów; ocena 5.0 19-20 punktów

c. ocenianie ciągle, na każdych zajęciach (odpowiedzi ustne) ? premiowanie przyrostu umiejętności posługiwania się poznanymi zasadami i metodami,

Uzyskiwanie punktów dodatkowych (maksymalnie do 2 pkt) za aktywność podczas zajęć, a szczególnie za:

a. omówienia dodatkowych aspektów zagadnienia omawianego podczas laboratorium/wykładu

b. efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,

c. uwagi związane z udoskonaleniem materiałów dydaktycznych,

Treści programowe

Pogram wykładu obejmuje następujące zagadnienia:

W trakcie semestru prowadzący przedmiot kompleksowo omawia podczas wykładu semantykę języka C++ w ujęciu obiektowym. Każdy z wykładów poświęcony jest innemu zagadnieniu i prezentuje rozwiązania w postaci gotowych implementacji. Studenci otrzymują materiały wykładowe w postaci plików zawierających gotowe pliki źródłowe wraz z kodem omawianym podczas danego wykładu. Treści wykładu uzupełnione są o dwa wykłady poświęcone modelowaniu UML, podczas których student zapoznaje się ze sposobem tworzenia dokumentacji i notacją UML (diagram klas i obiektów, diagram czynności, diagram sekwencji, diagram przypadków użycia, itd.) Ćwiczenia laboratoryjne przygotowane zostały w postaci plików pdf, które stanowią zadania do samodzielnej realizacji podczas zajęć. Student wykorzystuje w tym celu konto studenta na portalu e-learningowym Katedry Inżynierii Komputerowej (<http://dydaktyka.cce.put.poznan.pl/moodle>) i pobiera dokument z treścią zadania. Podczas zajęć, prowadzący przypomina w skrócie treści programowe związane z danym ćwiczeniem (omawiane wcześniej przez prowadzącego przedmiot podczas wykładu). Kolejne laboratoria obejmują następujące zagadnienia programowania obiektowego:

1. Język C++ - powtórka z semestru pierwszego i wyrównanie poziomu podstawowego grupy laboratoryjnej
2. Klasa i obiekt? kwantyfikatory sekcji prywatnej oraz publicznej klasy
3. Konstruktor (konstruktor kopiujący), destruktor, składniki statyczne klasy (modyfikator static)
4. Wskaźniki, tablice dynamiczne (poruszanie się po pamięci i dynamiczne jej przydzielenie/zwalnianie)
5. Dziedziczenie (kwantyfikator protected, zagadnienia dziedziczenia wielopoziomowego) oraz mechanizm funkcji zaprzyjaźnionych
6. Metody wirtualne oraz zagadnienie polimorfizmu
7. Abstrakcja klasy
8. Sprawdzian (wartość maksymalna 10 pkt)
9. Przeladowanie funkcji i operatorów
10. Rzutowanie i konwersja typów
11. STL, kontener listy (list) oraz kontener tablicy (vector)
12. Interfejs graficzny aplikacji okienkowej (dwa spotkania laboratoryjne)
13. Sprawdzian (wartość maksymalna 10 pkt)

Dodatkową treścią wykładów są ciekawe i inspirujące zagadnienia proponowane przez studentów na trakcie semestru, które następnie dyskutowane są w postaci prezentacji na ostatnim wykładzie w semestrze.

Metody dydaktyczne:

1. wykład: prezentacja multimedialna obejmująca implementację zagadnień programowania obiektowego omawianych na danym wykładzie (wszelkie materiały opracowane elektronicznie i osadzone na platformie e-learningowej)
2. ćwiczenia laboratoryjne: ćwiczenia praktyczne, implementacja zadań i algorytmów zdefiniowanych w opracowaniach powierzonych studentowi, dyskusja nad złożonością i optymalizacją kodu w trakcie zajęć (wszelkie materiały opracowane elektronicznie i osadzone na platformie e-learningowej)

Literatura podstawowa:

1. D. Vandevoorde, J. Mincer-Daszkiwicz, Język C++ : ćwiczenia i rozwiązania, Wyd. Naukowo-Techniczne, 2001
2. K. Walczak, Nauka programowania obiektowego w języku C++, Wydaw. W&W, 2004.
3. B. Stroustrup, Język C++, wydanie V, WNT, Warszawa 2000
4. Jerzy Grębosz, Symfonia C ++ Standard, Editions 2000, Kraków 2005
5. P. Paczuski, Programowanie w C++ : szkoła pisania programów od podstaw, Springer Polska, 2005.
6. Zbigniew Koza, Język C++. Pierwsze starcie, Helion, Gliwice, 2008

Literatura uzupełniająca:

1. R. Wirfs-Brock, A. McKean, Projektowanie obiektowe. Role, odpowiedzialność i współpraca. Wyd. Helion 2005
2. B. Eckel, Thinking In C++, Edycja polska, Wydawnictwo Helion

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)
1. udział w zajęciach laboratoryjnych:	30
2. udział w konsultacjach związanych z realizacją procesu kształcenia, w szczególności ćwiczeń laboratoryjnych	2
3. napisanie programu / programów, uruchomienie i weryfikacja (czas poza zajęciami laboratoryjnymi poświęcony pracy na platformie e-learningowej)	20
4. udział w wykładach:	30
5. zapoznanie się ze wskazaną literaturą / materiałami dydaktycznymi (10 stron tekstu naukowego = 1 godz.), 100 stron	10
6. przygotowanie do egzaminu z wykładów i udział w egzaminie (15+ 2 godz.)	17
7. przygotowanie do sprawdzianów	10

Obciążenie pracą studenta		
forma aktywności	godzin	ECTS
Łączny nakład pracy	119	5
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	64	3
Zajęcia o charakterze praktycznym	60	2