KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu	Kod
Podstawy programowania	**1010531121010551915**

Kierunek studiów	Profil kształcenia (ogólnoakademicki, praktyczny)	Rok / Semestr
Automatyka i robotyka	**ogólnoakademicki**	**1 / 2**
Ścieżka obieralności/specjalność	Przedmiot oferowany w języku:	Kurs (obligatoryjny/obieralny)
-	**polski**	**obligatoryjny**

Stopień studiów:	**Forma studiów** (stacjonarna/niestacjonarna)
I stopień	**stacjonarna**

Godziny	Liczba punktów
Wykłady: **30** Ćwiczenia: **-** Laboratoria: **30** Projekty/seminaria: **-**	**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	**Podział ECTS (liczba i %)**

Odpowiedzialny za przedmiot / wykładowca:

dr inż. Jakub Kołota
email: Jakub.Kolota@put.poznan.pl
tel. 61 6652751
Wydział Informatyki 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

Ocena podsumowująca:

a) w zakresie wykładów weryfikowanie założonych efektów kształcenia realizowane jest na podstawie odpowiedzi na pytania dotyczące materiału omówionego na wykładach (implementacja kodu weryfikowana na komputerze klasy PC w środowisku programistycznym CodeBlocks oraz Microsoft Visual Studio). Egzamin obejmuje indywidualne podejście do każdego egzaminowanego studenta i analizę kodu napisanego podczas egzaminu będącego obiektowym rozwiązaniem powierzonego zadania; W przypadku uzyskania wysokiej oceny z końcowej z laboratorium 4.0-5.0 prowadzący może przepisać jako część egzaminu.

b) w zakresie laboratoriów weryfikowanie założonych efektów kształcenia realizowane jest przez sprawdzian praktyczne przy komputerze oraz projekt końcowy aplikacji obiektowej. Ocena podsumowująca wystawiana jest jako średnia ocena uzyskana z projektu końcowego oraz kolokwium.

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 przydziela-nie/zwalnianie)
5. Dziedziczenie (kwantyfikator protected, zagadnienia dziedziczenia wielopoziomowego) oraz mechanizm funkcji zaprzyjaźnionych
6. Metody wirtualne oraz zagadnienie polimorfizmu
7. Abstrakcja klasy
8. Sprawdzian
9. Przeładowanie 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. Oddanie projektu

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-Daszkiewicz, 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. Jerzy Kisielewicz, Język C++ Programowanie obiektowe, Oficyna Wydawnicza Politechniki Wrocławskiej, 2005

Literatura uzupełniająca:

1. B. Eckel, Thinking In C++, Edycja polska, Wydawnictwo Helion
2. W. Gryglewicz-Kacerka, A. Duraj, Projektowanie obiektowe systemów informatycznych. Wydawnictwo PWSZ Włocławek, 2013

Bilans nakładu pracy przeciętnego studenta

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