



KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu

Podstawy informatyki [N1AiR2>PI]

Przedmiot

Kierunek studiów

Automatyka i robotyka

Rok/Semestr

1/1

Studia w zakresie (specjalność)

–

Profil studiów

ogólnoakademicki

Poziom studiów

pierwszego stopnia

Język oferowanego przedmiotu

polski

Forma studiów

niestacjonarne

Wymagalność

obligatoryjny

Liczba godzin

Wykład

20

Laboratorium

20

Inne

0

Ćwiczenia

0

Projekty/seminaria

0

Liczba punktów ECTS

5,00

Koordynatorzy

dr inż. Adam Turkot

adam.turkot@put.poznan.pl

Wykładowcy

mgr inż. Szymon Sobczak

szymon.sobczak@put.poznan.pl

mgr inż. Mikołaj Przybylski

mikolaj.przybylski@put.poznan.pl

dr inż. Janusz Pochmara

janusz.pochmara@put.poznan.pl

dr inż. Adam Turkot

adam.turkot@put.poznan.pl

mgr inż. Paweł Czopek

pawel.czopek@put.poznan.pl

Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z zakresu sprzętu komputerowego i jego obsługi. Powinien posiadać umiejętność rozwiązywania podstawowych problemów w obszarze pozyskiwania informacji ze wskazanych źródeł. Powinien również rozumieć konieczność poszerzania swoich kompetencji jak również być gotowym do podjęcia współpracy w ramach zespołu. 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

Zapoznanie z metodologią i zasadami programowania strukturalnego oraz obiektowego wykorzystując język programowania C++. 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. Kształtowanie u studentów umiejętności programistycznych. Kreowanie świadomości konieczności profesjonalnego podejścia do zagadnień technicznych, skrupulatnego zapoznania się z dokumentacją systemów informatycznych. Student uczy się wyznaczać cele i określać priorytety prowadzące do realizacji zadania poprzez strukturalną oraz obiektową implementację kodu.

Przedmiotowe efekty uczenia się

Wiedza:

K1_W8, K1_W9

- ma uporządkowaną wiedzę w zakresie wybranych algorytmów i struktur danych oraz metodyki i technik programowania proceduralnego i obiektowego

Umiejętności:

K1_U1

- 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
- potrafi analizować i symulować działanie algorytmów, dobierając struktury danych do pożądanej funkcjonalności kodu

Kompetencje społeczne:

K1_K2, K1_K5

- rozumie potrzebę i zna możliwości ciągłego dokształcania się, podnoszenia kompetencji zawodowych, osobistych i społecznych

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

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 rozwiązaniem powierzonego zadania. W przypadku uzyskania wysokiej oceny z końcowej z laboratorium 4.0-5.0 prowadzący może przepisać ją jako część egzaminacyjną.

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

Treści programowe

W trakcie semestru prowadzący przedmiot kompleksowo omawia podczas wykładu semantykę języka C++ w ujęciu strukturalnym oraz 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 na platformie e-learningowej. Treści wykładu uzupełnione są o wykłady poświęcone modelowaniu Unified Modeling Language (ang. 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.)

Tematyka zajęć

Omawiane podczas wykładu zagadnienia ćwiczone są w części laboratoryjnej przedmiotu. 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 e-kursy 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: wprowadzenie do składni języka i podstawowych instrukcji; klasa i obiekt – kwantyfikatory sekcji prywatnej oraz publicznej klasy; konstruktor (konstruktor kopiujący), destruktor, składniki statyczne klasy (modyfikator static); wskaźniki, tablice dynamiczne (poruszanie się po pamięci i dynamiczne jej przydzielanie/zwalnianie); dziedziczenie (zagadnienia dziedziczenia wielopoziomowego) oraz mechanizm funkcji zaprzyjaźnionych; metody wirtualne oraz zagadnienie polimorfizmu; abstrakcja klasy; przeładowanie funkcji i operatorów; rzutowanie i konwersja typów; STL, kontener listy (list) oraz kontener tablicy (vector); interfejs graficzny aplikacji okienkowej; realizacja i obrona własnego projektu.

Metody dydaktyczne

wykład: prezentacja multimedialna obejmująca implementację zagadnień programowania obiektowego omawianych na danym wykładzie (wszelkie materiały opracowane elektronicznie i osadzone na platformie e-learningowej)

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. Mirosław J. Kubiak, C++ : zadania z programowania z przykładowymi rozwiązaniami, Wyd. Helion, 2020
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

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

	Godzin	ECTS
Łączny nakład pracy	125	5,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	42	2,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwium/egzaminu, wykonanie projektu)	83	3,00