AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Nazwa modułu zajęć: Języki programowania obiektowego

Rok akademicki: 2019/2020 Kod: IETP-1-506-n Punkty ECTS: 3

Wydział: Informatyki, Elektroniki i Telekomunikacji

Kierunek: Elektronika i Telekomunikacja Specjalność: —

Poziom studiów: Studia I stopnia Forma studiów: Niestacjonarne

Język wykładowy: Polski Profil: Ogólnoakademicki (A) Semestr: 5

Strona www: —

Prowadzący moduł: dr inż. Frankowski Marek (mfrankow@agh.edu.pl)

Treści programowe zapewniające uzyskanie efektów uczenia się dla modułu zajęć

Programowanie obiektowe. Projektowanie współczesnych aplikacji obiektowych. UML i wzorce projektowe oprogramowania.

Opis efektów uczenia się dla modułu zajęć

Kod MEU	Student, który zaliczył moduł zajęć zna i rozumie/potrafi/jest gotów do	Powiązania z KEU	Sposób weryfikacji i oceny efektów uczenia się osiągniętych przez studenta w ramach poszczególnych form zajęć i dla całego modułu zajęć
Wiedza: zna i rozumie			
M_W001	Student posiada wiedzę w zakresie sposobów reprezentacji danych w programie, stosowania podstawowych instrukcji, operatorów, wyrażeń i funkcji w językach C/C++.	ETP1A_W06, ETP1A_W14	Kolokwium
M_W002	Student posiada wiedzę dotyczącą języków programowania obiektowego, dostępnych bibliotek oraz ich zastosowań.	ETP1A_W06, ETP1A_W14	Kolokwium
M_W003	Student posiada wiedzę dotyczącą klas i obiektów, mechanizmów dziedziczenia, polimorfizmu, przeciążania operatorów, tworzenia funkcji wirtualnych oraz rzutowania typów, tworzenia funkcji i klas szablonowych, dysponuje wiedzą z zakresu elementów biblioteki STL, jak również zasad tworzenia i testowania oprogramowania.	ETP1A_W06, ETP1A_W14	Kolokwium

M_W004	Student zna podstawy języka UML oraz umie go zastosować do opisu wzorców projektowych oprogramowania	ETP1A_U15, ETP1A_W14	Aktywność na zajęciach
Umiejętności: potrafi			
M_U001	Student potrafi opracować i zaimplementować określony algorytm posługując się podstawowymi mechanizmami programowania obiektowego, potrafi korzystać z zasobów biblioteki STL oraz funkcji i klas szablonowych do tworzenia oprogramowania.	ETP1A_U15	Kolokwium
M_U002	Student potrafi konstruować oprogramowanie zgodnie z określonymi standardami programistycznymi, tworzyć proste graficzne interfejsy użytkownika oraz tworzyć odpowiednią dokumentację projektu.	ETP1A_U15	Kolokwium
Kompetencje społeczne: jest gotów do			
M_K001	Student ma świadomość ciągłego podnoszenia swoich kwalifikacji.	ETP1A_K01	Studium przypadków
M_K002	Student ma świadomość odpowiedzialności za pracę własną oraz gotowość podporządkowania się zasadom pracy w zespole i ponoszenia odpowiedzialności za wspólnie realizowane zadania.	ETP1A_K04	Zaangażowanie w pracę zespołu
M_K003	Student potrafi wykorzystać opanowane technologie w przedsiębiorstwach oraz projektach komercyjnych.	ETP1A_K04	Udział w dyskusji

Liczba godzin zajęć w ramach poszczególnych form zajęć

	Forma zajęć dydaktycznych										
Suma	Wykład	Ćwiczenia audytoryjne	Ćwiczenia laboratoryjne	Ćwiczenia projektowe	Konwersatorium	Zajęcia seminaryjne	Zajęcia praktyczne	Zajęcia terenowe	Zajęcia warsztatowe	Prace kontrolne i przejściowe	Lektorat
30	16	0	14	0	0	0	0	0	0	0	0

Matryca kierunkowych efektów uczenia się w odniesieniu do form zajęć i sposobu zaliczenia, które pozwalają na ich uzyskanie

Kod MEU	Student, który zaliczył moduł zajęć zna i rozumie/potrafi/jest gotów do	Forma zajęć dydaktycznych

		Wykład	Ćwiczenia audytoryjne	Ćwiczenia laboratoryjne	Ćwiczenia projektowe	Konwersatorium	Zajęcia seminaryjne	Zajęcia praktyczne	Zajęcia terenowe	Zajęcia warsztatowe	Prace kontrolne i przejściowe	Lektorat
Wiedza: zna i rozumie												
M_W001	Student posiada wiedzę w zakresie sposobów reprezentacji danych w programie, stosowania podstawowych instrukcji, operatorów, wyrażeń i funkcji w językach C/C++.	+	-	+	-	-	-	-	-	-	-	-
M_W002	Student posiada wiedzę dotyczącą języków programowania obiektowego, dostępnych bibliotek oraz ich zastosowań.	+	-	+	-	-	-	-	-	-	-	-
M_W003	Student posiada wiedzę dotyczącą klas i obiektów, mechanizmów dziedziczenia, polimorfizmu, przeciążania operatorów, tworzenia funkcji wirtualnych oraz rzutowania typów, tworzenia funkcji i klas szablonowych, dysponuje wiedzą z zakresu elementów biblioteki STL, jak również zasad tworzenia i testowania oprogramowania.	+	-	+	-	-	-	-	-	-	-	-
M_W004	Student zna podstawy języka UML oraz umie go zastosować do opisu wzorców projektowych oprogramowania	-	-	-	-	-	-	-	-	-	-	-
Umiejętności: potrafi												
M_U001	Student potrafi opracować i zaimplementować określony algorytm posługując się podstawowymi mechanizmami programowania obiektowego, potrafi korzystać z zasobów biblioteki STL oraz funkcji i klas szablonowych do tworzenia oprogramowania.	+	-	+	-	-	-	-	-	-	-	-
M_U002	Student potrafi konstruować oprogramowanie zgodnie z określonymi standardami programistycznymi, tworzyć proste graficzne interfejsy użytkownika oraz tworzyć odpowiednią dokumentację projektu.	+	-	+	-	-	-	-	-	-	-	-
Kompetencje społeczne: jest gotów do												
M_K001	Student ma świadomość ciągłego podnoszenia swoich kwalifikacji.	+	-	+	-	-	-	-	-	-	-	-

M_K002	Student ma świadomość odpowiedzialności za pracę własną oraz gotowość podporządkowania się zasadom pracy w zespole i ponoszenia odpowiedzialności za wspólnie realizowane zadania.	+	-	+	-	-	-	-	-	-	-	-
M_K003	Student potrafi wykorzystać opanowane technologie w przedsiębiorstwach oraz projektach komercyjnych.	+	-	+	-	-	-	-	-	-	-	-

Nakład pracy studenta (bilans punktów ECTS)

Forma aktywności studenta	Obciążenie studenta
Udział w zajęciach dydaktycznych/praktyka	30 godz
Przygotowanie do zajęć	24 godz
przygotowanie projektu, prezentacji, pracy pisemnej, sprawozdania	8 godz
Samodzielne studiowanie tematyki zajęć	24 godz
Sumaryczne obciążenie pracą studenta	86 godz
Punkty ECTS za moduł	3 ECTS

Pozostałe informacje

Szczegółowe treści kształcenia w ramach poszczególnych form zajęć (szczegółowy program wykładów i pozostałych zajęć)

Wykład

Zajęcia w ramach modułu prowadzone są w postaci wykładu (18 godzin) oraz ćwiczeń laboratoryjnych (16 godzin).

Wykłady

1.Podstawy – Wprowadzenie do programowania obiektowego w C++

Wstęp do programowania; Przegląd języków programowania; Etapy budowy oprogramowania; Elementy Unified Modeling Language (UML): diagramy przypadków użycia, aktywności; Podstawy reprezentacji danych: stałe i zmienne, podstawowe typy danych i zasady ich używania (int, char, long, double); Podstawy C++: instrukcje if oraz for, podstawowe operatory arytmetyczne oraz logiczne, elementarne funkcje, przydatne obiekty (cout, cin, vector<>, string) wraz z przykładami; Przykład użycia podstawowych konstrukcji języka do implementacji prostego przykładu.

2.Podstawowe instrukcje, operatory, wyrażenia i funkcje

Podstawowe instrukcje C/C++: warunkowa (if, switch-case), pętle (while, for, do-

while), throw-try-catch; Przegląd wszystkich grup operatorów, prawa hierarchii oraz łączności: arytmetyczne, logiczne, bitowe, warunkowe, inne; Wskaźniki: rola, składania, semantyka; Tablice; Struktury; Referencje; Budowa wyrażeń; Funkcje: rola, struktura, przekazywanie parametrów do funkcji, zwracanie wartości z funkcji; Organizacja projektu C/C++: preprocesing, kompilacja, konsolidacja, debuggowanie; Przykłady w rzeczywistych aplikacjach;

3.Programowanie obiektowe

Koncepcja obiektu i klasy: zasadnicze paradygmaty programowania obiektowego (encapsulation, polymorphism, code reusability, code safety), relacje has-a, is-a, knows; Anatomia klasy: składowe klasy, wskaźnik this, ochrona składowych, grupa metod konstrukcyjnych, grupa metod dostępu do danych, operacje input/output, static members, member pointers; Typ enumerowany; Zasady projektowania klas na przykładach; Przeciążanie operatorów: podstawowe konstrukcje, przykłady; Dziedziczenie i polimorfizm: podstawowe pojęcia, funkcje wirtualne, projektowanie hierarchii klas, wirtualne klasy bazowe; Rzutowanie typów; Klasy specjalne: funktory, friends; Obsługa wyjątków; Przestrzenie nazw; Analiza istniejących rozwiązań;

4.Szablony C++

Idea meta-programowania, wstęp od szablonów; Funkcje szablonowe: deklaracja parametrów; Szablony klas: deklaracja parametrów szablonowych, sposoby łączenia, template-template parameters, szablony typowe, szablony nietypowe, funkcje wirtualne, szablonowe składowe klasy (template members), parametry domniemane (default), szablonowe hierarchie klas. Instancjacja klas szablonowych; Proces dedukcji argumentów szablonowych; Specjalne klasy szablonowe: traits oraz policy, funkcje określania typu, smart pointers, funktory oraz funcje typu callback; Metodyka projektowania konstrukcji szablonowych; Przykłady w rzeczywistych aplikacjach;

5.Biblioteka standardowa STL

Wprowadzenie do STL: przykład użycia podstawowych konstrukcji (vector, string), podział biblioteki (kontenery, iteratory, algorytmy); Podstawowe kontenery, ich właściwości oraz użycie: vector, list, set oraz multiset, maps oraz multimaps; Iteratory: kategorie iteratorów (input, output, forward, bidirectional, random access), adaptery iteratorów; Obiekty funkcyjne STL: budowa oraz używanie predykatów; Algortymy STL: niemodyfikujące (liczenie elementów, minimum/maksimum, przeszukiwanie kontenerów, porównywanie zakresów), modyfikujące (kopiowanie, transformacje, zamiana, przypisywanie wartości elementów), usuwające składowe kontenerów, mutujące (odwracanie kolejności, rotacje, permutacje), sortowanie, algorytmy numeryczne; Kontenery specjalne: kolejka, stos; Operacje input/output: podstawowe klasy I/O, przeciążenia operatorów stream, manipulatory, formatowanie, dostęp do plików, strumienie ciągów znakowych;

6.Elementy projektowania oprogramowania. Konstrukcje zaawansowane

Rozszerzenie wiadomości o UML; Projektowanie oprogramowania: etapy konstrukcyjne, dokumentacja; Standardy programistyczne; Konstrukcja bibliotek; Testowanie oprogramowania: rodzaje błędów, konstruowanie testów, programowanie przez kontrakt, sztuka debuggowania;

7.Inne języki obiektowe

Wstęp do języka Python: specyfika języków skryptowych, podstawowe konstrukcje, tryby używania języka, różnice w stosunku do C++, programowanie obiektowe, wstęp do bibliotek Pythona; Przykłady w rzeczywistych aplikacjach;

Ćwiczenia laboratoryjne

Ćwiczenia laboratoryjne

1.Opracowanie oraz implementacja problemu reprezentacji oraz obliczania pierwiastków trójmianu kwadratowego – Praca w środowisku Linux, dokumentacja wykonywanego zadania, realizacja programowa, metody kompilacji, konsolidacji oraz debuggowania, metody radzenia sobie z błędami.
2.Opracowanie oraz implementacja kalkulatora walut – Praca w środowisku Visual C++, utworzenie aplikacji do poszukiwania kursów walut w danych z banku, realizacja połączenia internetowego, projekt interfejsu użytkownika (GUI), wykonanie w MFC i/lub FLTK.
3.Opracowanie oraz implementacja dziennika zajęć – Implementacja prostej bazy danych z funkcjami dodawania/usuwania/sortowania rekordów; Umiejętność obsługi operacji wejścia/wyjścia, praca z plikami.
4.Opracowanie oraz implementacja klasy liczb zespolonych – Implementacja klasy liczb zespolonych, przeciążanie operatorów, operacje przesyłania danych do i z plików, operatory konwersji typów. Implementacja metod programowania “przez kontrakt”.
5.Opracowanie oraz implementacja klasy do reprezentacji macierzy, wzorzec handle-body, wzorzec proxy – Implementacja określonych wzorców projektowych na przykładzie macierzy. Przykłady wykorzystania w aplikacjach obliczeniowych.
6.Opracowanie oraz implementacja aplikacji wykorzystującej grafikę komputerową z OpenGL – zapoznanie z biblioteką graficzną OpenGL, projekt prostej aplikacji w oparciu o zasoby biblioteki graficznej, praca z hierarchiami klas oraz z funkcjami wirtualnymi. Przykłady wykorzystania w aplikacjach graficznych.
7.Opracowanie oraz implementacja prostej gry komputerowej – Stworzenie aplikacji zgodnie z prawidłowymi wzorcami projektowymi, modelowanie problemu za pomocą języka UML, dokumentacja projektu.
8.Projekt własny – Projekt i implementacja własnej aplikacji napisanej w języku Python; Wykorzystanie bibliotek Python, zarządzanie projektem oraz grupą projektową.

Metody i techniki kształcenia:

Wykład: Treści prezentowane na wykładzie są przekazywane w formie prezentacji multimedialnej w połączeniu z klasycznym wykładem tablicowym wzbogaconymi o pokazy odnoszące się do prezentowanych zagadnień.
Ćwiczenia laboratoryjne: W trakcie zajęć laboratoryjnych studenci samodzielnie rozwiązują zadany problem praktyczny, dobierając odpowiednie narzędzia. Prowadzący stymuluje grupę do refleksji nad problemem, tak by otrzymane wyniki miały wysoką wartość merytoryczną.

Warunki i sposób zaliczenia poszczególnych form zajęć, w tym zasady zaliczeń poprawkowych, a także warunki dopuszczenia do egzaminu:

Ocenie podlega rozwiązywanie problemów zadanych na zajęciach, wykonywanie programistycznych zadań domowych oraz pisanie, krótkich kartkówek na zajęciach. Aby uzyskać zaliczenie należy zebrać przynajmniej 50% z każdej z tych trzech części. Następnie ocena jest wyznaczana zgodnie z regulaminem studiów.

Zasady udziału w poszczególnych zajęciach, ze wskazaniem, czy obecność studenta na zajęciach jest obowiązkowa:

Wykład:
– Obecność obowiązkowa: Nie
– Zasady udziału w zajęciach: Studenci uczestniczą w zajęciach poznając kolejne treści nauczania zgodnie z syllabusem przedmiotu. Studenci winni na bieżąco zadawać pytania i wyjaśniać wątpliwości. Rejestracja audiowizualna wykładu wymaga zgody prowadzącego.
Ćwiczenia laboratoryjne:
– Obecność obowiązkowa: Tak
– Zasady udziału w zajęciach: Studenci wykonują ćwiczenia laboratoryjne zgodnie z materiałami udostępnionymi przez prowadzącego. Student jest zobowiązany do przygotowania się w przedmiocie wykonywanego ćwiczenia, co może zostać zweryfikowane kolokwium w formie ustnej lub pisemnej. Zaliczenie zajęć odbywa się na podstawie zaprezentowania rozwiązania postawionego problemu. Zaliczenie modułu jest możliwe po zaliczeniu wszystkich zajęć laboratoryjnych.

Sposób obliczania oceny końcowej

Ocena końcowa jest oceną z laboratorium.

Sposób i tryb wyrównywania zaległości powstałych wskutek nieobecności studenta na zajęciach:

W przypadku nieobecności nie przekraczających liczby dopuszczalnej przez regulamin studiów, sposób nadrobienia materiału określa prowadzący laboratorium (zadanie domowe lub kolokiwum poprawkowe).

Wymagania wstępne i dodatkowe, z uwzględnieniem sekwencyjności modułów

Podstawy metodyki i technik programowania

Zalecana literatura i pomoce naukowe

Stroustrup B. The C++ Programming Language, 2000 (tł. polskie: Język C++, WNT).
Vandervoorde D., Josuttis N.M. C++ Templates. Addison Wesley, 2003 (tł. polskie: C++. Szablony. Vademecum profesjonalisty, Helion).
J. Gręboszc: Symfonia C++ standard. Wydawnictwo Edition 2000, Kraków 2005.
E. Gamma, J. Vlissides, R. Johnson, R. Helm, Wzorce projetkowe. Elementy oprogramowania wielokrotnego użytku, Wydawnictwo Helion, 2012.

Darmowe materiały online:
https://www.omg.org/spec/UML/
https://books.goalkicker.com/CPlusPlusBook/

Publikacje naukowe osób prowadzących zajęcia związane z tematyką modułu

- Spatial Spectrum Analyzer (SSA): A tool for calculations of spatial distribution of fast Fourier transform spectrum from Object Oriented Micromagnetic Framework output data, Marek Frankowski, Jakub Chęciński, Maciej Czapkiewicz, Computer Physics Communications 189, 207-212, 2015
- MAGE (M-file/Mif Automatic GEnerator): A graphical interface tool for automatic generation of Object Oriented Micromagnetic Framework configuration files and Matlab scripts for results analysis, Jakub Chęciński, Marek Frankowski, Computer Physics Communications 207, 487–498, 2016
- Electric-field tunable spin waves in PMN-PT/NiFe heterostructure: Experiment and micromagnetic simulations, Sławomir Ziętek, Jakub Chęciński, Marek Frankowski, Witold Skowroński, Tomasz Stobiecki, Journal of Magnetism and Magnetic Materials 428, 64–69, 2017

Informacje dodatkowe

Brak