PODSTAWY PROGRAMOWANIA

Kod modułu: PPR

Rodzaj przedmiotu: kierunkowy; obowiązkowy

Wydział: Informatyki

Kierunek: Informatyka

Poziom studiów: pierwszego stopnia – VI poziom PRK

Profil studiów: praktyczny

Forma studiów: stacjonarna/niestacjonarna

Rok: 1

Semestr: 1

Formy zajęć i liczba godzin:

Forma stacjonarna

wykłady – 30

laboratorium – 25

Forma niestacjonarna

wykłady – 20

laboratorium – 15

Zajęcia prowadzone są w języku polskim.

Liczba punktów ECTS: 5

Osoby prowadzące:

wykład:
laboratorium:

---

1. Założenia i cele przedmiotu:

Podstawowym celem zajęć jest osiągnięcie dobrego poziomu opanowania umiejętności programistycznych, obejmujących podstawy programowania. Przewiduje się wprowadzenie, przećwiczenie i utrwalenie kluczowych zagadnień dla nauki programowania, realizowanych z wykorzystaniem języka C++. Stanowić to ma stabilną podbudowę zajęć z programowania, realizowanych w następnych semestrach. Zajęcia przedstawią pojęcie algorytmu, podstawowe konstrukcje programistyczne, implementacje algorytmów w języku C++, podstawowe struktury danych i wykonywane na nich operacje, również dynamiczny przydział pamięci oraz rekurencja i jej implementacja w językach wysokiego poziomu. Przewiduje się także omówienie metod weryfikacji poprawności programów.

Czysto praktycznym celem zajęć jest dobre opanowanie umiejętności związanych z posługiwaniem się środowiskiem programistycznym typu IDE. Obejmuje to edycję tekstu programów, manipulowanie plikami, kompilację, lokalizowanie, identyfikowanie i poprawiane zarówno błędów kompilacji jak i wykonania. Zakłada się, iż studenci będą rozumieć podstawowe pojęcia oraz będą potrafić je

praktycznie wykorzystać. Obejmuje to również umiejętność właściwego nazywania takich elementów programu jak: literał, zmienna, stała, operator, typ danych, deklaracja zmiennych, instrukcja przypisania, instrukcja warunkowa, iteracyjna, itd., itp.

Przedmiot Podstawy Programowania z grupy przedmiotów podstawowych, jest przedmiotem wprowadzającym do przedmiotów Języki Programowania, Języki Programowania Obiektowego oraz przedmiotów z zakresu specjalizacji Projektowanie systemów informatycznych i programowanie w środowisku internetowym.

2. Określenie przedmiotów wprowadzających wraz z wymaganiami wstępnymi:

-

3. Opis form zajęć

a) Wykłady

- **Treści programowe**

1. Pojęcie algorytmu. Metody zapisu algorytmów. Pseudokod, schematy blokowe.
2. Kod źródłowy, kod wynikowy. Kompilacja, interpretacja, maszyna wirtualna. Asembler, kompilator, interpreter.
3. Zapis algorytmów w języku C++. Struktura programu w języku C++. Pojęcie zmiennej i typu danych. Deklaracje zmiennych, definicje stałych. Instrukcja przypisania, budowanie wyrażeń arytmetycznych. Operacje wejścia-wyjścia.
4. Instrukcje sterujące wykonaniem programu. Instrukcja złożona, instrukcje warunkowe, wyboru iteracyjne. Zamienność instrukcji iteracyjnych.
5. Typy proste i złożone. Typy tablicowe. Podstawowe operacje na tablicach. Algorytmy wyszukiwania. Proste algorytmy sortowania tablic.
6. Koncepcja podprogramów, metody wykorzystania podprogramów. Deklaracja i wywołanie. Przekazywanie parametrów przez zmienną i wartość. Rekurencja.
7. Programowanie strukturalne. Wykorzystanie podprogramów w strukturalnej dekompozycji problemów. Budowanie złożonych programów z wykorzystaniem podprogramów i modułów.
8. Podstawowe struktury danych i wykonywane na nich operacje. Dynamiczny przydział pamięci oraz rekurencja i jej implementacja w językach wysokiego poziomu.
9. Omówienie metod weryfikacji poprawności programów.

- **Metody dydaktyczne:**

Wykład prowadzony metodą tradycyjną z wykorzystaniem rzutnika multimedialnego i prezentacją algorytmów oraz programów.

- **Forma i warunki zaliczenia:**

Warunkiem zaliczenia całości przedmiotu jest zdanie egzaminu w wymaganej formie, określonej właściwą siatką studiów.

- **Wykaz literatury podstawowej:**

1. P. Van Roy, Seif Haridi, Programowanie koncepcje techniki i modele, 2004, HELION.
2. S. Prata, Szkoła programowania. Język C++, 2006, Helion.
3. Gaddis T.: Projektowanie oprogramowania dla zupełnie początkujących. Gliwice: Helion, 2020.
4. Coldwind G.: Zrozumieć programowanie. Warszawa: Wydawnictwo Naukowe PWN, 2017.
5. N. Wirth, Algorytmy + struktury danych = programy wydanie siódme, 2004, wydanie najnowsze.

6. Martin R.C.: Czysta architektura. Struktura i design oprogramowania. Przewodnik dla profesjonalistów. Gliwice: Helion, copyright 2018.

- **Wykaz literatury uzupełniającej:**

1. A.V. Aho, J.E. Hopcroft, Jeffrey D. Ullman, Projektowanie i analiza algorytmów, 2003, HELION.
2. R. Hyde, Profesjonalne programowanie, część 1, Zrozumieć komputer, 2005, HELION.
3. R. Hyde, Profesjonalne programowanie, część 2, Myśl niskopoziomowo pisz wysokopoziomowo, 2005, HELION.
4. J. Robbins, Debugger usuwanie błędów z programów, 2001, READ ME.
5. Wróblewski P.: Algorytmy, struktury danych i techniki programowania. Gliwice: HELION, cop. 2019.

b) Ćwiczenia laboratoryjne

- **Treści programowe (tematyka zajęć):**

1. Metody zapisu algorytmów, pseudokod, schematy blokowe.
2. Kod źródłowy, kod wynikowy, kompilacja, kompilator, interpreter. Zintegrowane środowisko programisty i metody wykorzystania jego komponentów.
3. Struktura programu w języku C++, zmienne, typu danych, deklaracje zmiennych, definicje stałych.
4. Instrukcja przypisania, budowanie wyrażeń arytmetycznych, Operacje wejścia-wyjścia.
5. Instrukcje sterujące wykonaniem programu, instrukcja złożona, instrukcje warunkowe, wyboru iteracyjne.
6. Typy proste i złożone, typy tablicowe.
7. Podstawowe operacje na tablicach. Algorytmy wyszukiwania. Proste algorytmy sortowania tablic.
8. Koncepcja podprogramów, metody wykorzystania podprogramów.
9. Funkcje, deklaracja i wywołanie. Przekazywanie parametrów przez zmienną i wartość. Rekurencja.
10. Programowanie strukturalne. Wykorzystanie podprogramów w strukturalnej dekompozycji problemów.
11. Budowanie złożonych programów z wykorzystaniem podprogramów i modułów.
12. Podstawowe struktury danych i wykonywane na nich operacje.
13. Dynamiczny przydział pamięci oraz rekurencja i jej implementacja w językach wysokiego poziomu.
14. Omówienie metod weryfikacji poprawności programów.

- **Metody dydaktyczne:**

W ramach ćwiczeń laboratoryjnych zakłada się realizację bloków tematycznych, obejmujących spójne treściowo przykłady oraz zadania do indywidualnego wykonania. Każdy blok ćwiczeniowy obejmuje syntezę programu przykładowego, opartego na materiale wykładowym, a pogłębionego o aspekty praktyczne, wprowadzone przez osoby prowadzące ćwiczenia. Po zakończeniu prac nad przykładem, studenci realizują zadania indywidualne pod nadzorem osoby prowadzącej ćwiczenia, konsultując uzyskane rozwiązania. Każdy blok ćwiczeniowy obejmuje zadania dodatkowe, które mogą być realizowane w ramach zajęć własnych studentów oraz mogą być dedykowane studentom wykazującym wyjątkowo dobre opanowanie omawianych zagadnień.

- **Forma i warunki zaliczenia:**

Ocena jest wypadkową (zgodnie z pkt. 8) oceny sprawdzianów, prac kontrolnych oraz sprawozdania z projektu. Zaliczenie projektu odbywa się na podstawie oceny zrealizowanego programu. Ocenie podlega jego zgodność z założeniami oraz poziom realizacji programowej, adekwatność wybranych rozwiązań, stopień wykorzystania wybranej technologii i narzędzi programowania.

- **Wykaz literatury podstawowej:**

1. P. Van Roy, Seif Haridi, Programowanie koncepcje techniki i modele, 2004, HELION.
2. S. Prata, Szkoła programowania. Język C++, 2006, Helion.
3. Gaddis T.: Projektowanie oprogramowania dla zupełnie początkujących. Gliwice: Helion, 2020.
4. Coldwind G.: Zrozumieć programowanie. Warszawa: Wydawnictwo Naukowe PWN, 2017.
5. N. Wirth, Algorytmy + struktury danych = programy wydanie siódme, 2004, wydanie najnowsze.
6. Martin R.C.: Czysta architektura. Struktura i design oprogramowania. Przewodnik dla profesjonalistów. Gliwice: Helion, copyright 2018.

- **Wykaz literatury uzupełniającej:**

1. A.V. Aho, J.E. Hopcroft, Jeffrey D. Ullman, Projektowanie i analiza algorytmów, 2003, HELION.
2. R. Hyde, Profesjonalne programowanie, część 1, Zrozumieć komputer, 2005, HELION.
3. R. Hyde, Profesjonalne programowanie, część 2, Myśl niskopoziomowo pisz wysokopoziomowo, 2005, HELION.
4. J. Robbins, Debugger usuwanie błędów z programów, 2001, READ ME.
5. Wróblewski P.: Algorytmy, struktury danych i techniki programowania. Gliwice: HELION, cop. 2019..

4. Opis sposobu wyznaczania punktów ECTS

a. forma stacjonarna

Forma zajęć	Formy aktywności studenta	Średnia ilość godzin na zrealizowanie aktywności
Wykład	kontakt z nauczycielem	30
	studiowanie literatury oraz zasobów internetowych	20
	przygotowanie do egzaminu	10
Laboratorium	kontakt z nauczycielem	25
	projektowanie i tworzenie programów	20
	sporządzenie dokumentacji	20

Całkowita ilość godzin aktywności studenta	125
Liczba punktów ECTS dla modułu	5

b. forma niestacjonarna

Forma zajęć	Formy aktywności studenta	Średnia ilość godzin na zrealizowanie aktywności
Wykład	kontakt z nauczycielem	20
	studiowanie literatury oraz zasobów internetowych	25
	przygotowanie do egzaminu	15
Laboratorium	Kontakt z nauczycielem	15
	projektowanie i tworzenie algorytmów	25
	przygotowanie do zajęć	25

Całkowita ilość godzin aktywności studenta	125

Liczba punktów ECTS dla modułu	5

5. Wskaźniki sumaryczne

a. forma stacjonarna

a) liczba godzin dydaktycznych (tzw. kontaktowych) i liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich
   - Liczba godzin kontaktowych – 55
   - Liczba punktów ECTS – 2,2

b) liczba godzin dydaktycznych (tzw. kontaktowych) i liczba punktów ECTS na zajęciach o charakterze praktycznym.
   - Liczba godzin kontaktowych – 25
   - Liczba punktów ECTS – 2,6

b. forma niestacjonarna

a) liczba godzin dydaktycznych (tzw. kontaktowych) i liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich
   - Liczba godzin kontaktowych – 35
   - Liczba punktów ECTS – 1,4

b) liczba godzin dydaktycznych (tzw. kontaktowych) i liczba punktów ECTS na zajęciach o charakterze praktycznym.
   - Liczba godzin kontaktowych – 15
   - Liczba punktów ECTS – 2,6

6. Zakładane efekty uczenia się

Numer (Symbol)	Efekty uczenia się dla modułu	Odniesienie do efektów uczenia się dla kierunku
PPR_01	… zna pojęcie algorytmu i programu komputerowego, metody zapisu algorytmów, paradygmaty języków programowania oraz główne metody i techniki programowania: programowanie proceduralne, programowanie obiektowe, programowanie strukturalne.	K_W04, K_W07 K_W12
PPR_02	… rozumie podstawowe konstrukcje programistyczne, zasady ich translacji oraz zna reprezentację wewnętrzną danych.	K_W04, K_W07 K_W12
PPR_03	… ma podstawową wiedzę dotyczącą niezawodności i optymalizacji programów oraz ich dokumentowania.	K_W04, K_W07 K_W12, K_U23
PPR_04	… potrafi skonstruować algorytm rozwiązujący podany problem algorytmiczny.	K_U11, K_U19 K_U23
PPR_05	… potrafi zastosować podstawowe konstrukcje programistyczne.	K_U11, K_U19 K_U23
PPR_06	… rozumie konieczność podnoszenia swoich kompetencji zawodowych.	K_K01
PPR_07	… potrafi pracować w zespole projektowo-programistycznym.	K_K01, K_K02

7. Odniesienie efektów uczenia się do form zajęć i sposób oceny osiągnięcia przez studenta efektów uczenia się

Numer (Symbol)	Forma zajęć		Sposób sprawdzenia osiągnięcia efektu
	wykład	ćwiczenia	
PPR_01	x		Egzamin
PPR_02	x		Sprawdzian
PPR_03	x		Sprawdzian
PPR_04		x	Sprawdzian
PPR_05		x	Sprawdzian
PPR_06		x	Dyskusja + obserwacja pracy
PPR_07		x	Dyskusja + obserwacja pracy

8. Kryteria uznania osiągnięcia przez studenta efektów uczenia się

Numer (Symbol)	Efekt jest uznawany za osiągnięty gdy:
PPR_01	Student poprawnie rozwiąże zadania egzaminacyjne sprawdzające wiedzę o podejściu strukturalnym w programowaniu, algorytmach, metodach zapisu algorytmów oraz głównych metodach i technikach programowania.
PPR_02	Sprawdzian zawiera poprawny kod programu realizującego rozwiązanie zadania określonego przez prowadzącego z zakresu podstawowych konstrukcji programistycznych.
PPR_03	Sprawdzian zawiera poprawny kod programu spełniającego wskazane kryteria optymalności i niezawodności.
PPR_04	Sprawdzian zawiera poprawny algorytm i kod programu rozwiązującego zadany problem programistyczny.
PPR_05	Sprawdzian zawiera poprawny kod programu wykorzystujący wskazane konstrukcje programistyczne.
PPR_06	… aktywnie poszukiwał informacji wspomagających rozwiązywanie postawionych zadań, świadomie rozwijając swoje kompetencje.
PPR_07	… pracuje w grupie realizując programistyczne zadania projektowe i implementacyjne.