Opis przedmiotu		
PROGRAMMING 1 - FUNDAMENTALS		
Kod przedmiotu	1120-IN000-ISA-0116	
Nazwa przedmiotu	Programming 1 – Fundamentals	
Nazwa przedmiotu w jęz. polskim	Programowanie 1 – podstawy	
A. Usytuowanie przedmiotu w systemie studiów		
Poziom kształcenia	Studia pierwszego stopnia	
Forma i tryb prowadzenia studiów	Stacjonarne	
Kierunek studiów	Informatyka i Systemy Informacyjne	
Profil studiów	Profil ogólnoakademicki	
Specjalność	-	
Jednostka prowadząca	Wydział Matematyki i Nauk Informacyjnych	
Jednostka realizująca	Wydział Matematyki i Nauk Informacyjnych	
Koordynator przedmiotu	dr inż. Rafał Jóźwiak, mgr inż. Paweł Aszklar	
B. Ogólna charakterystyka przedmiotu		
Blok przedmiotów	Kierunkowe	
Grupa przedmiotów	Obowiązkowe	
Status przedmiotu	Obowiązkowy	
Język prowadzenia zajęć	Angielski	
Semestr nominalny	1	
Usytuowanie realizacji w roku akademickim	Semestr zimowy	
Wymagania wstępne /przedmioty poprzedzające	-	
Limit liczby studentów	Ćwiczenia – 30 os. /grupa Laboratoria (ćwiczenia komputerowe) – 15 os. / grupa	
C. Efekty kształcenia i sposób prowadzenia zajęć		
Cel przedmiotu	Studenci powinni nauczyć się tworzyć programy strukturalne w języku C. Powinni zaznajomić się z opracowywaniem rozwiązań prostych problemów programistycznych, formułowaniem algorytmów i implementacją ich w języku C. Powinni stosować zasady języka C w poprawnym stylu programowania. Students should learn to write structured programs in the C language. They should become familiar working out solutions to simple programming problems, formulate algorithms and implement them in the C language. They should apply the C language rules in a good programming style.	
Efekty kształcenia	Patrz TABELA 4.	
Formy zajęć i ich wymiar	Wykład	30
	Ćwiczenia	15
	Laboratorium	30
	Projekt	0

Treści kształcenia	Kurs obejmuje następujące zagadnienia: • Struktura programu w C. Schematy blokowe. Analiza przykładowego programu. • Podstawowe typy danych. Definicje stałych (literałów). Deklaracja zmiennych i tablic. Operatory i wyrażenia. • Kontrola przepływu. Instrukcje warunkowe. instrukcja goto i etykiety. Pętle: while, for, do-while. instrukcje break, continue i return. instrukcja switch. • Tablice jednowymiarowe. Adresy i wskazówki. arytmetyka wskaźników, • Funkcje: definicje, deklaracje i wywołania. • Tablice dwuwymiarowe. Arytmetyka wskaźników (cd.) • Alokacja pamięci: malloc, calloc, free. • Typ znakowy. Stałe i zmienne znakowe. Tablice znaków i ciągi znaków. Inicjalizacja. Przetwarzanie ciągów znaków: funkcje niestandardowe. • Ciągi znaków cd.: Funkcje biblioteczne. • Struktury i unie. Przykładowe definicje. Deklaracja typedef. Inicjalizacja. • Tworzenie i przetwarzanie list: podstawowe operacje. rekursja. • Operacje I/O na plikach. • Operatory cd.: Operatory bitowe. Literały całkowite: dziesiętne, szesnastkowe, ósemkowe. • Zmienne funkcje argumentów. Deklaracja i użycie wskaźników funkcji. • Programy wielomodułowe. *The course covers the following topics:* • *Program structure in C. Flowcharts. Analysis of a sample program.* • *Basic data types. Constant (literals) definitions. Declaration of variables and arrays. Operators and expressions.* • *Flow control. if statement. goto statement and labels. Loops: while, for, do-while. break, continue and return statements. switch statement.* • *One-dimensional arrays. Addresses and pointers. Pointer arithmetic,* • *Functions: definitions, declarations and calls.* • *2-dimensional arrays. Pointer arithmetics (contd.)* • *Memory allocation: malloc, calloc, free.* • *The char type. Character constants and variables. Character arrays and strings. Initialization. Characterstring processing: custom functions.* • *Character strings contd.: Library functions.* • *Structures and unions. Sample definitions. typedef declaration. Initialization. Member access.* • *List creation and processing: Basic operations. Recursion.* • *I/O operation on files.* • *Operators contd.: Bitwise operators. Integer literals: decimal, hexadecimal, octal.* • *Variable argument functions. Declaration and usage of function pointers.* • *Multi-module programs.*

Metody oceny	Zasady oceny ćwiczeń • Każde laboratorium instruktażowe, z wyjątkiem pierwszego, rozpocznie się od krótkiego 5-minutowego testu. • Każdy test jest oceniany do 2 punktów, w sumie maksymalnie 12 punktów. • Wyniki każdego krótkiego testu zostaną zaprezentowane na kolejnych zajęciach laboratoryjnych. • Usprawiedliwiona nieobecność spowoduje proporcjonalne przeskalowanie innych wyników. • Końcowy test pisemny odbędzie się w 14. tygodniu. • Studenci będą mieli 90 minut na udzielenie odpowiedzi na zestaw pytań. • Będzie oceniane do 40 punktów. • Wyniki będą dostępne w USOS w ciągu tygodnia od testu końcowego. • W razie potrzeby w ostatnim tygodniu semestru zaplanowano poprawkę do kolokwium końcowego. • Korzystanie z notatek, dodatkowych materiałów i/lub urządzeń jest zabronione podczas jakichkolwiek testów. • Do zaliczenia ćwiczeń potrzeba 26 punktów. Zasady oceniania laboratorium • Każde ocenione laboratorium będzie punktowane do 8 punktów. • Na rozwiązanie zadanego zadania uczniowie będą mieli 90 minut. • Każde zadanie będzie oceniane podczas zajęć na podstawie zrozumienia problemu przez studenta, postępów w pisaniu rozwiązania oraz jakości kodu źródłowego. • Dodatkowo poszczególne rozwiązania zostaną porównane po zajęciach przez program antyplagiatowy. • Rozwiązania należy pisać na komputerach udostępnionych w sali. • Podczas zajęć studenci mogą korzystać ze slajdów z wykładów, własnych notatek z wykładów i ćwiczeń, rozwiązań poprzednich zadań oraz dokumentacji języka C. • Używanie jakichkolwiek innych materiałów i/lub urządzeń jest zabronione. • Dopuszczalne są maksymalnie 3 nieobecności w semestrze. • Niestawienie się w ciągu pierwszych 15 minut traktowane jest jako nieobecność. • Zamiast laboratorium powtórkowego, najniższa ocena zostanie odrzucona. • Studenci mogą uzyskać maksymalnie 48 punktów. • Do zaliczenia laboratoriów na ocenę potrzebne są 24 punkty. Zasady ogólne i ocena kursu • Użycie jakichkolwiek niedozwolonych materiałów lub urządzeń podczas testów i laboratoriów ocenianych skutkuje przyznaniem 0 punktów za dany test lub zadanie. • Wielokrotne naruszanie powyższej zasady skutkuje niezdaniem przedmiotu. • Usprawiedliwiona nieobecność na ocenianych laboratoriach lub podczas kolokwium spowoduje proporcjonalne przeskalowanie innych wyników. • Warunkiem zaliczenia przedmiotu jest zaliczenie ćwiczeń i laboratoriów. • Ocena końcowa będzie uzależniona od sumy punktów z ćwiczeń i laboratoriów: • 0-50 punktów - 2,0 (D - Niezaliczony) • 51-60 punktów - 3,0 (C - Dostateczny) • 61-70 punktów - 3,5 (C+ - Dość dobry) • 71-80 punktów - 4,0 (B - Dobra) • 81-90 punktów - 4,5 (B+ - powyżej dobrego) o 91-100 punktów - 5,0 (A - bardzo dobry) Wcześniejsze zaliczenie przedmiotu • Studenci zaznajomieni z tematem mogą próbować zdać wcześniejszy test końcowy. Zostanie on zaplanowany w pierwszym miesiącu semestru i będzie się składał z dwóch części: •Test pisemny:

	• Uczniowie będą mieli 90 minut na udzielenie odpowiedzi na zestaw pytań. • Będzie oceniane do 50 punktów, • Aby przejść, potrzeba 35 punktów. • Zadanie laboratoryjne: • Wstęp mają tylko studenci, którzy zdali część pisemną. • Studenci będą mieli 90 minut na napisanie programu rozwiązującego zadane zadanie. • Zostanie oceniony do 8 punktów na podstawie zrozumienia problemu przez studenta, postępów w pisaniu rozwiązania oraz jakości kodu źródłowego. • Do zaliczenia tej części potrzeba 6 punktów. • Studenci, którzy zaliczą obie części automatycznie otrzymają ocenę końcową 5,0 (A - Very Good). Uczniowie, którzy nie zdadzą egzaminu, będą musieli normalnie uczęszczać na zajęcia. *Tutorial Assessment Rules* • *Each tutorial laboratory, except the first one, will begin with a short 5 minutes test.* • *Each test is graded for up to 2 points, for a total of maximum 12 points.* • *Results of each short test will be presented during next laboratory class.* • *Excused absence will cause other results to be scaled proportionally.* • *Final written test will take place during 14th week.* • *Students will have 90 minutes to answer a set of questions.* • *It will be graded for up to 40 points.* • *Results will be available in USOS within a week from the final test.* • *If necessary, a retake of final test will be scheduled during the last week of the semester.* • *Use of notes, additional material and/or devices is disallowed during any of the tests.* • *26 points are needed to pass the tutorials.* *Laboratory Assessment Rules* • *Each graded lab will be scored for up to 8 points.* • *Students will have 90 minutes to solve a given task.* • *Each task will be graded during class based on student's understanding of the problem, progress towards writing a solution and quality of the source code.* • *Additionally individual solutions will be compared after class by anti-plagiarism software.* • *Solutions need to be written on computers provided in the classroom.* • *During the class students are allowed to use lecture slides, own lecture and tutorial notes, previous tasks solutions and C reference page.* • *Use of any other material and/or devices is prohibited.* • *Maximum of 3 absences are allowed in the semester.* • *Failure to arrive within the first 15 minutes is considered an absence.* • *Instead of a retake laboratory, the lowest score will be dropped.* • *Students can gain up to 48 points.* • *24 points are needed to pass the graded laboratories.* *General Rules and Course Assessment* • *Use of any unauthorized material or devices during tests and graded laboratories will result in 0 points for a given test or task.* • *Repeat offenders will fail the subject.* • *Excused absence on graded labs or during short test will cause other results to be scaled proportionally.* • *To pass the subject student needs to pass both tutorials and laboratories.* • *Final grade will depend on the sum total of points from tutorials and laboratories:* • *0-50 points - 2.0 (D - Fail)* • *51-60 points - 3.0 (C - Satisfactory)*

	• *61-70 points - 3.5 (C+ - Fairly Good)*
	• *71-80 points - 4.0 (B - Good)*
	• *81-90 points - 4.5 (B+ - Above Good)*
	  ○ *91-100 points - 5.0 (A - Very Good)*
	
	Passing Course Early
	
	• *Students familiar with the subject may attempt to pass an early final test. It will be scheduled during the first month of the semester and will consist of two parts:*
	• *Written test:*
	  ○ *Students will have 90 minutes to answer a set of questions.*
	  ○ *It will be graded for up to 50 points,*
	  ○ *35 points are needed to pass.*
	• *Laboratory task:*
	  ○ *Only students who pass the written part are allowed.*
	  ○ *Students will have 90 minutes to write a program solving a given task.*
	  ○ *It will be graded for up to 8 points based on student's understanding of the problem, progress towards writing a solution and quality of the source code.*
	  ○ *6 points are needed to pass this part.*
	
	Students who pass both parts will automatically receive final grade of 5.0 (A - Very Good). Students who fail will have to attend regular classes as normal.

Metody sprawdzania efektów kształcenia	Patrz TABELA 1.
Egzamin	Nie
Literatura	1. The C Programming Language, B.W. Kernighan, D.M. Ritchie, Second Edition, Prentice Hall Inc., 1988. 2. C Primer Plus, Stephen Prata, 6th Edition, Addison-Wesley, 2014. 3. Modern C, Jens Gustedt, First Edition, Manning, 2019, (https://modernc.gforge.inria.fr/ — 2nd Edition, CC BY-NC-ND 4.0) 4. The C Answer Book, C.L. Tondo, S.E. Gimpel, Second Edition, Prentice Hall Inc., 1988.
Witryna www przedmiotu	http://pages.mini.pw.edu.pl/~aszklarp/p1.php
D. Nakład pracy studenta	
Liczba punktów ECTS	6

Liczba godzin pracy studenta związanych z osiągnięciem efektów kształcenia	godziny kontaktowe – 80 h; w tym a. obecność na wykładach – 30 h b. obecność na ćwiczeniach – 15 h c. obecność na laboratoriach – 30 h d. konsultacje – 5 h praca własna studenta – 75 h; w tym a. przygotowanie do wykładów – 5 h b. przygotowanie do ćwiczeń – 20 h c. przygotowanie do sprawdzianów pisemnych (kolokwiów) – 20 h d. przygotowanie do zajęć laboratoryjnych – 30 h Razem 155 h, co odpowiada 6 pkt. ECTS
Liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich:	obecność na wykładach – 30 h obecność na ćwiczeniach – 15 h obecność na laboratoriach – 30 h konsultacje z prowadzącymi zajęcia – 5 h Razem 80 h, co odpowiada 3 pkt. ECTS
Liczba punktów ECTS, którą student uzyskuje w ramach zajęć o charakterze praktycznym	obecność na ćwiczeniach – 15 h obecność na laboratoriach – 30 h przygotowanie do ćwiczeń – 20 h przygotowanie do sprawdzianów pisemnych – 20 h przygotowanie do zajęć laboratoryjnych – 30 h Razem 115 h, co odpowiada 4 pkt. ECTS
E. Informacje dodatkowe	
Uwagi	-

TABELA 1. LEARNING OUTCOMES OF THE COURSE

1. Learning outcomes and their reference to the learning outcomes in the area of technical sciences and for the field of study Computer Science and Information Systems

Learning outcomes of the module	DESCRIPTION OF THE LEARNING OUTCOMES A graduate of first level studies in the field *Computer Science and Information Systems*:	Odniesienie do efektów uczenia się dla kierunków	Sposób weryfikacji **Verification method**
	KNOWLEDGE		
W01	Posiada usystematyzowaną wiedzę nt. konstruowania prostych algorytmów i rozdzielania ich na funkcje. *Possesses systematic knowledge pertaining the construction of simple algorithms and separating them into functions.*	I1A_W05	graded classroom activity, test, computer programs (written in C language)
W02	Zna podstawowe elementy języka C: wbudowane i zdefiniowane przez użytkownika typy danych, wyrażenia, instrukcje, funkcje, obsługę wejścia i wyjścia, wskaźniki, pamięć dynamicznie alokowaną. *Knows core elements of the C language: built-in and user-defined data types, expressions, statements, functions, input and output handling, pointers, dynamically allocated memory.*	I1A_W05 I1A_W10 I1A_W11	graded classroom activity, test, computer programs (written in C language)
W03	Zna podstawowe komponenty środowiska programistycznego: IDE, edytor, kompilator, linker, biblioteki. *Knows the essential components of a development environment: IDE, editor, compiler, linker, libraries.*	I1A_W04	writing computer programs (in C language)
	SKILLS		

U01	Potrafi rozwiązywać proste problemy algorytmiczne i implementować rozwiązanie w języku programowania wysokiego poziomu (C). *Is able to solve simple algorithmic problems and implement the solution in a high-level programming language (C).*	I1A_U02, I1A_U18	graded classroom activity, test, computer programs (written in C language)
U02	Potrafi napisać kod źródłowy, skompilować, uruchomić i sprawdzić poprawność wykonania programu w języku C. *Is able to produce the source code, compile, run and check the correct execution of a program in C language.*	I1A_U17	writing computer programs (in C language)
SOCIAL COMPETENCIES			
K01	Rozumie potrzebę przestrzegania standardów programowania i najlepszych praktyk. *Understands the need to adhere to programming standards and best practices.*	I1A_K02	discussion
K02	Rozumie potrzebę pisania uniwersalnego i przenośnego kodu, który można łatwo wykorzystać ponownie. *Understands the need to write universal and portable code which can be easily reused.*	I1A_K06	discussion