



## KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu

Języki formalne i kompilatory [S1Inf1>JFK]

### Przedmiot

Kierunek studiów  
Informatyka

Rok/Semestr  
2/4

Studia w zakresie (specjalność)  
–

Profil studiów  
ogólnoakademicki

Poziom studiów  
pierwszego stopnia

Język oferowanego przedmiotu  
polski

Forma studiów  
stacjonarne

Wymagalność  
obieralny

### Liczba godzin

Wykład  
16

Laboratorium  
16

Inne (np. online)  
0

Ćwiczenia  
0

Projekty/seminaria  
0

### Liczba punktów ECTS

3,00

### Koordynatorzy

dr inż. Wojciech Complak  
wojciech.complak@put.poznan.pl

### Wykładowcy

### Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z algorytmiki i programowania w językach imperatywnych. Powinien posiadać umiejętność rozwiązywania podstawowych problemów z zakresu projektowania, sprawdzenia poprawności i implementowania algorytmów w języku C oraz umiejętność pozyskiwania informacji ze wskazanych źródeł. Powinien również rozumieć konieczność poszerzania swoich kompetencji / mieć gotowość 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

1. Przekazanie studentom podstawowej wiedzy z zakresu praktycznych aspektów teorii języków formalnych oraz budowy translatorów i środowisk czasu wykonania w zakresie zasad, technik i narzędzi wykorzystywanych współcześnie do budowy kompilatorów i innych narzędzi do automatycznego przetwarzania tekstu, takich jak: edytory tekstu, systemy wyszukiwania informacji, systemy składu elektronicznego i weryfikatory programów. 2. Rozwijanie u studentów umiejętności rozwiązywania prostych problemów za pomocą języków programowania ogólnego przeznaczenia, jak i z wykorzystaniem do tego celu specjalistycznych narzędzi. Poszerzenie wiedzy na temat wcześniej wykorzystywanych środowisk programistycznych i języków programowania w wyniku spojrzenia na nie z punktu widzenia projektanta i implementatora, a nie tylko użytkownika.

## Przedmiotowe efekty uczenia się

Wiedza:

1. ma rozszerzoną i pogłębioną wiedzę z matematyki przydatną do formułowania i rozwiązywania złożonych zadań informatycznych dotyczących formalnej specyfikacji i weryfikacji oprogramowania
2. ma uporządkowaną i podbudowaną teoretycznie wiedzę ogólną w zakresie języków i paradygmatów programowania
3. zna podstawowe techniki, metody i narzędzia wykorzystywane w procesie rozwiązywania zadań informatycznych w zakresie analizy algorytmów i implementacji języków programowania

Umiejętności:

1. potrafi właściwie zaplanować i wykonać testy funkcjonalne i pozafunkcjonalne oprogramowania
2. potrafi zaprojektować oraz zrealizować zadanie informatyczne stosując odpowiednio dobrane metody analityczne i eksperymentalne
3. ma umiejętność specyfikowania i implementowania analizatorów z wykorzystaniem poznanych narzędzi

Kompetencje społeczne:

1. ma świadomość znaczenia wiedzy z zakresu języków formalnych i kompilatorów w rozwiązywaniu problemów inżynierskich

## Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Ocena formująca:

a) w zakresie wykładów:

- na podstawie odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach

b) w zakresie laboratoriów:

- na podstawie oceny bieżącego postępu realizacji zadań

Ocena podsumowująca:

Sprawdzanie założonych efektów kształcenia realizowane jest przez:

- ocenę przygotowania studenta do poszczególnych sesji zajęć laboratoryjnych (sprawdzian 'wejściowy') oraz ocenę umiejętności związanych z realizacją ćwiczeń laboratoryjnych,

- ocenianie ciągle, na każdych zajęciach (odpowiedzi ustne) - premiowanie przyrostu umiejętności postępowania się poznanymi zasadami i metodami,

- ocenę wiedzy i umiejętności związanych z realizacją zadań projektowych/laboratoryjnych poprzez kolokwium pod koniec semestru, kolokwium obejmuje zadania o charakterze praktycznym dotyczące poszczególnych narzędzi i zagadnień omawianych w ramach przedmiotu (AWK, lex, yacc, SLR); zadania mają zarówno charakter konstrukcyjny (np. napisz program) jak i analityczny (np. jaka będzie odpowiedź danego programu)

- ocenę i 'obronę' przez studenta sprawozdania z realizacji projektu.

Uzyskiwanie punktów dodatkowych za aktywność podczas zajęć, a szczególnie za:

- omówienia dodatkowych aspektów zagadnienia,

- efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,

- uwagi związane z udoskonaleniem materiałów dydaktycznych,

- wskazywanie trudności percepcyjnych studentów umożliwiające bieżące doskonalenia procesu dydaktycznego.

## Treści programowe

Pierwszy wykład poświęcony jest omówieniu organizacji zajęć (zakresu przedmiotu, środowiska i narzędzi, literatury i zasad zaliczania) oraz wprowadzeniu do tematyki przetwarzania tekstu na przykładzie języka AWK.

Na drugim wykładzie przedstawiany jest model analiza-synteza translatora, podział procesu translacji na etapy oraz faza analizy leksykalnej i zasady prowadzenie jej z wykorzystaniem generatora analizatorów leksykalnych lex.

Trzeci wykład, otwierający cykl poświęcony analizie składniowej, zawiera omówienie ogólnych zasad prowadzenia analizy składniowej i pojęć związanych z gramatykami bezkontekstowymi (takich jak: terminale i nieterminale, produkcje, wywody, typy rekurencji, niejednoznaczność i równoważność gramatyk) oraz wstęp do metody wstępującej. Wykład obejmuje charakterystykę generatora yacc, składnię specyfikacji analizatora składniowego, zasady współpracy z analizatorem leksykalnym oraz wykrywania i obsługi błędów składniowych.

W trakcie kolejnego wykładu przedstawiana jest koncepcja translacji sterowanej składnią.

Przedstawiane są pojęcia atrybutów, definicji sterowanych składnią, schematów translacji oraz definicji S-atrybutowych i L-atrybutowych. Omawiane są także zasady implementacji translacji sterowanej składnią w metodzie wstępującej w generatorze yacc (atrybuty syntetyzowane i dziedziczone, typy atrybutów, akcje wielokrotne).

Kolejny wykład z cyklu dotyczącego analizy składniowej poświęcony jest posługiwaniu się gramatykami niejednoznaczными w metodzie wstępującej w generatorze yacc. Przedstawiane są zalety i typowe, praktyczne przykłady gramatyk niejednoznacznych oraz zasady wykorzystywania ich w generatorze yacc. W ramach szóstego wykładu przedstawiana jest analiza semantyczna: różne typy kontroli zależności kontekstowych, takie jak sprawdzenie przepływu sterowania, unikalności deklaracji nazw, powtórzeń nazw oraz kontrola typów.

Wykład kończący cykl dotyczący analizy składniowej poświęcony jest porównaniu wad i zalet różnych metod tworzenia translatorów działających w oparciu o metodę wstępującą oraz demonstracji sposobu generowania kodu parsera.

Ostatni wykład poświęcony jest etapowi syntezy kodu pośredniego: omawiane są różne rodzaje kodów pośrednich i maszyny wirtualne, a jako przykład konkretnej implementacji, szczegółowo przedstawiany jest kod trójadresowy. W ramach wykładu dokonywany jest również przegląd zagadnień dotyczących generacji kodu wynikowego i jego optymalizacji oraz budowy środowiska wykonawczego, takich jak dostęp do nazw nielokalnych, dynamiczny przydział pamięci i przekazywanie parametrów do podprogramów.

Pierwsze zajęcia laboratoryjne poświęcone są zagadnieniom organizacyjnym: zaznajomieniu się ze środowiskiem i narzędziami, uruchamianiem skryptów do kompilacji oraz nauce wykorzystywania języka AWK do przetwarzania tekstu.

W trakcie kolejnych zajęć studenci przechodzą do zapoznawania się z projektowaniem i implementowaniem prostych filtrów tekstu z wykorzystaniem generatora analizatorów leksykalnych lex.

Pozostałe zajęcia laboratoryjne poświęcone są metodzie bottom-up i generatorom lex i yacc: implementacji prostych filtrów tekstu, zasadom łączenia analizatorów składniowych i leksykalnych, implementacji analizatora prostego języka programowania imperatywnego oraz translatora tłumaczącego kod pomiędzy wybranymi, znanymi językami imperatywnymi.

Podczas ostatnich zajęć laboratoryjnych studenci rozliczają wykonanie projektów translatorów oraz piszą test końcowy.

## Metody dydaktyczne

1. wykład: prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy, rozwiązywanie zadań, demonstracja narzędzi programistycznych,
2. ćwiczenia laboratoryjne: rozwiązywanie zadań, dyskusja.

## Literatura

Podstawowa:

1. Kompilatory. Reguły, metody i narzędzia, A. V. Aho, R. Sethi, J. D. Ullman, WNT, Warszawa, 2002
2. Automatyczne przetwarzanie tekstów, J. Cybulka, B. Jankowska, J. Nawrocki, Nakom, Poznań, 2002
3. Wprowadzenie do przetwarzania tekstów w języku AWK, J. Nawrocki, W. Complak, Nakom (Pro Dialog), Poznań, 1994
4. Wprowadzenie do generatora Lex, J. Nawrocki, A. Czajka, Nakom (Pro Dialog), Poznań, 1998

Uzupełniająca:

1. lex & yacc, 2nd Edition, D. Brown, J. Levine, T. Mason, O'Reilly Media, 1992
2. The Definitive ANTLR Reference: Building Domain-Specific Languages, T. Parr, The Pragmatic Bookshelf, 2007
3. Compilers: Principles, Techniques, and Tools, 2. Ed., A.V. Aho, M. S. Lam, R. Sethi, J.D. Ullman, Addison-Wesley, 2007

### Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	75	3,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	32	1,50
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwii/egzaminu, wykonanie projektu)	43	1,50