# KARTA OPISU MODUŁU KSZTAŁCENIA

| Nazwa modułu/przedmiotu | Kod |
|---|---|
| **Języki formalne i kompilatory** | **1010514361010510088** |

| Kierunek studiów | Profil kształcenia (ogólnoakademicki, praktyczny) | Rok / Semestr |
|---|---|---|
| **Informatyka** | **ogólnoakademicki** | **3 / 6** |
| Ścieżka obieralności/specjalność: **-** | Przedmiot oferowany w języku: **polski** | Kurs (obligatoryjny/obieralny) **obieralny** |

| **Stopień studiów:** | **Forma studiów** (stacjonarna/niestacjonarna) |
|---|---|
| **I stopień** | **niestacjonarna** |

| Godziny | Liczba punktów |
|---|---|
| Wykłady: **16** Ćwiczenia: **-** Laboratoria: **16** Projekty/seminaria: **-** | **2** |

| Status przedmiotu w programie studiów (podstawowy, kierunkowy, inny) | (ogólnouczelniany, z innego kierunku) |
|---|---|
| **kierunkowy** | **z danego kierunku** |

| **Obszar(y) kształcenia i dziedzina(y) nauki i sztuki** | **Podział ECTS (liczba i %)** |
|---|---|
| **nauki techniczne** | **2 100%** |

## Odpowiedzialny za przedmiot / wykładowca:

Dr inż. W. Complak
email: Wojciech.Complak@cs.put.poznan.pl
tel. (0-61) 665-2983
Instytut Informatyki
ul. Piotrowo 2, 60-965 Poznań

## Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

| | | |
|---|---|---|
| 1 | **Wiedza:** | Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z algorytmiki i programowania w językach imperatywnych. |
| 2 | **Umiejętności:** | Powinien posiadać umiejętność rozwiązywania podstawowych problemów z zakresu zaprojektowania, sprawdzenia poprawności i zaimplementowania algorytmów w języku C oraz umiejętność pozyskiwania informacji ze wskazanych źródeł. |
| 3 | **Kompetencje społeczne** | 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 języku 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 spojrzeniu na nie z punktu widzenia projektanta i implementatora a nie tylko użytkownika.
3. Kształtowanie u studentów umiejętności pracy zespołowej poprzez realizację projektów programistycznych w grupach wymagającą umiejętności dzielenie się zadaniami i kompetencjami.

## Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia

### Wiedza:

1. ma uporządkowaną, podbudowaną teoretycznie wiedzę ogólną w zakresie algorytmów i złożoności, architektury systemów komputerowych, systemów operacyjnych, języków i paradygmatów programowania oraz inżynierii oprogramowania - [K_W4]
2. zna podstawowe metody, techniki i narzędzia stosowane przy rozwiązywaniu prostych zadań informatycznych z zakresu analizy złożoności obliczeniowej algorytmów i problemów, budowy systemów komputerowych, systemów operacyjnych, implementacji języków programowania oraz inżynierii oprogramowania - [K_W8]
3. ma szczegółową wiedzę nt. algorytmiki - [K_W5]
4. zna i rozumie zasady doboru narzędzi programistycznych i metod przetwarzania tekstu do specyfiki rozwiązywanego zadania - [-]
5. ma wiedzę niezbędną do zrozumienia, modyfikowania, dostosowania, udokumentowania i utrzymania istniejących rozwiązań programistycznych - [-]

### Umiejętności:

1. potrafi pozyskiwać informacje z literatury, baz danych oraz innych źródeł (w języku ojczystym i angielskim), integrować je, dokonywać ich interpretacji i krytycznej oceny, wyciągać wnioski oraz formułować i wyczerpująco uzasadniać opinie - [K_U1]
2. potrafi ocenić złożoność obliczeniową algorytmów i problemów - [K_U13]
3. ma umiejętność systematycznego przeprowadzania testów funkcjonalnych - [K_U17]
4. potrafi wybrać język programowania odpowiedni do danego zadania programistycznego - [K_U20]
5. potrafi - zgodnie z zadaną specyfikacją - zaprojektować oraz zrealizować prosty system informatyczny, używając właściwych metod, technik i narzędzi - [K_U21]
6. ma umiejętność formułowania algorytmów i ich programowania z użyciem przynajmniej jednego z popularnych narzędzi, - [K_U22]

**Kompetencje społeczne:**

1. rozumie, że w informatyce wiedza i umiejętności bardzo szybko stają się przestarzałe - [K_K1]
2. potrafi inspirować i organizować proces uczenia się innych osób - [K_K2]
3. potrafi współdziałać i pracować w grupie, przyjmując w niej różne role - [K_K5]

## Sposoby sprawdzenia efektów kształcenia

Efekty kształcenia 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 ćwiczeń:

- 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 ciągłe, na każdych zajęciach (odpowiedzi ustne) ? premiowanie przyrostu umiejętności posługiwania się poznanymi zasadami i metodami,
- ocenę wiedzy i umiejętności związanych z realizacją zadań projektowych / laboratoryjnych poprzez kolokwium na koniec semestru, kolokwium obejmuje 11 zadań o charakterze praktycznym dotyczących poszczególnych narzędzi i zagadnień omawianych w ramach przedmiotu (2 x AWK, 2 x lex, 2 x LLgen, 2 x yacc, 3 x 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, ocena ta obejmuje także umiejętność pracy w zespole,

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

- omówienia dodatkowych aspektów zagadnienia,
- efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,
- umiejętność współpracy w ramach zespołu praktycznie realizującego zadanie szczegółowe w laboratorium,
- 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.

Pierwsze zajęcia laboratoryjne poświęcone są zagadnieniom organizacyjnym: sprawdzeniu poprawności działania kont, zaznajomieniu się ze środowiskiem i narzędziami oraz uruchamianiem skryptów do kompilacji.

Następnie 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.

W trakcie zajęć laboratoryjnych studenci uczą się wykorzystywania języka AWK do przetwarzania tekstu rozwiązując przykładowe zadania z wykorzystaniem interpretera GAWK.

Następnie przedstawiany jest 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 zstępującej.

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

W dalszym ciągu cyklu poświęconego analizie składniowej prezentowany jest generator analizatorów składniowych działających w oparciu o metodę zstępującą ? Llgen. W pierwszym wykładzie poświęconym temu generatorowi prezentowany są ogólne zasady jego działania i konstruowania specyfikacji analizatorów syntaktycznych.

Na zajęciach laboratoryjnych studenci rozpoczynają samodzielne konstruowanie bardziej złożonych analizatorów tekstu wejściowego leksykalnych z wykorzystaniem generatora lex.

Następnie przedstawiany jest koncepcja translacji sterowanej składnią. Przedstawiane są pojęcia atrybutów, definicji sterowanych składnią, schematów translacji oraz definicji S-atrybutowych i L-atrybutowych. Omawiane są również zasady implementacji translacji sterowanej składnią w generatorze LLgen.

W ramach zajęć laboratoryjnych studenci mają za zadanie samodzielne zaprojektowanie, implementację i testowanie analizatora kodu źródłowego języka programowania (np. obliczającego metryki) z pomocą generatora lex.

Następnie przedstawiana jest metoda wstępująca zasadom konstruowania i działania analizatorów działających tą metodą i generatorowi yacc. Wykład obejmuje charakterystykę generatora yacc, składnię specyfikacji analizatora składniowego, zasady współpracy z analizatorem leksykalnym oraz wykrywania i obsługi błędów składniowych.

W trakcie zajęć laboratoryjnych studenci, implementując proste filtry tekstu, zapoznają się z podstawami generatora LLgen i zasadami łączenia analizatorów składniowych i leksykalnych.

W ramach następnego wykładu przedstawiane są zasady implementacji translacji sterowanej składnią w metodzie wstępującej w generatorze yacc (atrybuty syntetyzowane i dziedziczone, typy atrybutów, akcje wielokrotne).

Na zajęciach laboratoryjnych studenci tworzą samodzielne analizatory tekstu z wykorzystaniem LLgena i lexa.

Kolejny wykład z cyklu dotyczącego analizy składniowej poświęcony jest posługiwaniu się gramatykami niejednoznacznymi w metodzie wstępującej w generatorze yacc. Przedstawiane są zalety i typowe, praktyczne przykłady gramatyk niejednoznacznych oraz zasady wykorzystywania ich w generatorze yacc.

W czasie zajęć laboratoryjnych studenci samodzielnie implementują z wykorzystaniem generatorów LLgen i lex translator prostego języka programowania imperatywnego.

Następnie przedstawiany jest analiza semantyczna: różne typy kontroli zależności kontekstowych, takie jak: sprawdzenie przepływu sterowania, unikalności deklaracji nazw, powtórzeń nazw oraz kontrola typów.

Na zajęciach laboratoryjnych studenci, rozwiązując proste zadania, rozpoczynają zapoznawanie się z generatorem yacc.

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.

Na zajęciach laboratoryjnych studenci implementują kompletne analizatory z wykorzystaniem lexa i yacca.

Kolejny 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.

Na laboratoriach studenci rozpoczynają, w dwuosobowych grupach, implementację w yaccu i lexie prototypu translatora dokonującego automatycznego tłumaczenia kodu pomiędzy wybranymi, znanymi im językami imperatywnymi.

Następnym zagadnieniem omawianym na wykładach jest generacja kodu wynikowego i jego optymalizacja.

Na zajęciach laboratoryjnych studenci kontynuują w grupach realizację projektu translatora kodu.

Kolejny wykład dotyczy zagadnień budowy środowiska wykonawczego, takich jak dostęp do nazw nielokalnych, dynamiczny przydział pamięci i przekazywanie parametrów do podprogramów.

Na laboratoriach studenci kończą implementację i przeprowadzają testy przygotowywanych prototypów translatorów.

Ostatnim prezentowanym na wykładach narzędziem jest zintegrowane środowisko nowoczesnego generatora ANTLR.

W czasie zajęć laboratoryjnych rozpoczyna się rozliczanie projektów przez poszczególne zespoły.

Ostatni wykład poświęcony jest podsumowaniu całości przedstawionych w ciągu semestru zagadnień oraz sprawdzeniu wiedzy studentów w formie kolokwium zaliczeniowego.

Cześć wymienionych wyżej treści programowych realizowana jest w ramach pracy własnej studenta.

Metody dydaktyczne:

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

**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

**Literatura 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**

| Czynność | Czas (godz.) |
|---|---|
| 1. udział w zajęciach laboratoryjnych / ćwiczeniach: | 16 |
| 2. przygotowanie do ćwiczeń laboratoryjnych:. | 8 |
| 3. udział w konsultacjach związanych z realizacją procesu kształcenia, w szczególności ćwiczeń laboratoryjnych / projektu | 2 |
| 4. napisanie programu / programów, uruchomienie I weryfikacja (czas poza zajęciami laboratoryjnymi) | 10 |
| 5. przygotowanie do sprawdzianów / kolokwium | 4 |
| 6. udział w wykładach | 16 |
| 7. zapoznanie się ze wskazaną literaturą / materiałami dydaktycznymi (10 stron tekstu naukowego = 1 godz.), 50 stron | 5 |
| 8. przygotowanie do zaliczenia wykładów i udział w kolokwium zaliczeniowym | 4 |

**Obciążenie pracą studenta**

| forma aktywności | godzin | ECTS |
|---|---|---|
| Łączny nakład pracy | 65 | 2 |
| Zajęcia wymagające bezpośredniego kontaktu z nauczycielem | 34 | 1 |
| Zajęcia o charakterze praktycznym | 34 | 1 |