**Nazwa przedmiotu:**

Metody translacji

**Koordynator przedmiotu:**

Dr inż. Jan Bródka

**Status przedmiotu:**

Obowiązkowy

**Poziom kształcenia:**

Studia I stopnia

**Program:**

Informatyka i Systemy Informacyjne

**Grupa przedmiotów:**

Wspólne

**Kod przedmiotu:**

1120-IN000-ISP-0363

**Semestr nominalny:**

6 / rok ak. 2020/2021

**Liczba punktów ECTS:**

5

**Liczba godzin pracy studenta związanych z osiągnięciem efektów uczenia się:**

**Liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich:**

**Język prowadzenia zajęć:**

polski

**Liczba punktów ECTS, którą student uzyskuje w ramach zajęć o charakterze praktycznym:**

**Formy zajęć i ich wymiar w semestrze:**

|  |  |
| --- | --- |
| Wykład:  | 30h |
| Ćwiczenia:  | 15h |
| Laboratorium:  | 0h |
| Projekt:  | 0h |
| Lekcje komputerowe:  | 0h |

**Wymagania wstępne:**

Biegła znajomość więcej niż jednego języka programowania wysokiego poziomu, znajomość wyrażeń regularnych i gramatyk bezkontekstowych, znajomość podstawowych struktur danych. Przydatna znajomość języka zorientowanego maszynowo (asemblera).
Przedmioty poprzedzające: Teoria automatów i języków formalnych, Programowanie 2 – obiektowe, Programowanie 3 – zaawansowane, Algorytmy i struktury danych 1

**Limit liczby studentów:**

Ćwiczenia – 30 os/grupa

**Cel przedmiotu:**

Celem przedmiotu jest zdobycie wiedzy na temat przebiegu procesu kompilacji, zrozumienie wpływu cech języka źródłowego (wysokiego poziomu) na wydajność generowanego kodu maszynowego oraz nabycie umiejętności przetwarzania tekstów o sformalizowanej strukturze metodami stosowanymi w kompilatorach z wykorzystaniem popularnych narzędzi. Po ukończeniu kursu studenci powinni:
- znać podstawowe fazy procesu kompilacji (analiza leksykalna, składniowa i semantyczna, generowanie kodu pośredniego i docelowego, optymalizacja) i rozumieć ich znaczenie,
- znać podstawy teoretyczne procesu analizy kodu źródłowego,
- rozumieć wpływ różnorodnych konstrukcji języków wysokiego poziomu na wydajność generowanego na ich podstawie kodu wynikowego,
- rozumieć znaczenie kodu pośredniego i znać podstawowe metody jego optymalizacji,
- posiadać podstawową wiedzę na temat kodu CIL i LLVM,
- umieć stosować wybrane narzędzia wspomagające analizę leksykalną i składniową.

**Treści kształcenia:**

Wykład:
Ogólne pojęcia i fazy procesu kompilacji. Przegląd własności języków programowania. Modele środowiska wykonawczego. Analiza leksykalna. Analiza składniowa i gramatyki bezkontekstowe, w tym metoda zejść rekurencyjnych, analizatory LL(1), analizatory LR(1). Analiza semantyczna i gramatyki atrybutywne. Diagnostyka i obsługa błędów. Generowanie kodu pośredniego i wynikowego, na przykładzie kodu CIL i LLVM. Optymalizacja.
Ćwiczenia:
Ilustracja materiału z wykładu, przykłady użycia generatorów i narzędzi wspomagających tworzenie kompilatorów, w tym FLEX, Bison, Gardens Point, LLVM

**Metody oceny:**

Zaliczenie odbywa się łącznie dla wykładu i ćwiczeń. Podstawą zaliczenia jest pisemne kolokwium końcowe. Dodatkowo na ocenę pozytywny wpływ mają nieobowiązkowe zadania domowe i aktywność na ćwiczeniach, a negatywny nieobecności (obecność bez aktywności jest neutralna). Studenci wykazujący się zdecydowanie ponadprzeciętną wiedzą i zaangażowaniem (aktywnością na ćwiczeniach) mogą być zwolnieni z kolokwium końcowego (każdorazowa indywidualna decyzja wykładowcy).

**Egzamin:**

nie

**Literatura:**

1. A.V. Aho, R. Sethi, J.D. Ullman, Kompilatory: Reguły, metody, narzędzia, WNT 2002.
2. K.C. Louden, Compiler Construction: Principles and Practice, PWS 1997.
3. Dostępna w internecie dokumentacja projektów FLEX, Bison, Gardens Point, LLVM i innych.
4. Materiały z wykładów na stronie internetowej http://www.mini.pw.edu.pl/~brodka.

**Witryna www przedmiotu:**

e.mini.pw.edu.pl

**Uwagi:**

## Charakterystyki przedmiotowe

### Profil ogólnoakademicki - wiedza

**Charakterystyka W01:**

Ma uporządkowaną, podbudowaną teoretycznie wiedzę ogólną w zakresie języków i paradygmatów programowania

Weryfikacja:

kolokwium końcowe

**Powiązane charakterystyki kierunkowe:** K\_W06

**Powiązane charakterystyki obszarowe:**

**Charakterystyka W02:**

Zna podstawowe metody, techniki i narzędzia stosowane przy implementacji języków programowania

Weryfikacja:

kolokwium końcowe

**Powiązane charakterystyki kierunkowe:** K\_W10

**Powiązane charakterystyki obszarowe:**

### Profil ogólnoakademicki - umiejętności

**Charakterystyka U01:**

Ma umiejętność formułowania algorytmów i ich programowania z użyciem popularnych narzędzi

Weryfikacja:

kolokwium końcowe, zadania domowe

**Powiązane charakterystyki kierunkowe:** K\_U14

**Powiązane charakterystyki obszarowe:**