# Równoległe i funkcyjne techniki programowania - opis przedmiotu

## Informacje ogólne

| | |
|---|---|
| Nazwa przedmiotu | Równoległe i funkcyjne techniki programowania |
| Kod przedmiotu | 11.3-WI-INFD-RówniFunkcTechProg-S15 |
| Wydział | Wydział Nauk Inżynieryjno-Technicznych |
| Kierunek | Informatyka |
| Profil | ogólnoakademicki |
| Rodzaj studiów | drugiego stopnia z tyt. magistra inżyniera |
| Semestr rozpoczęcia | semestr zimowy 2021/2022 |

## Informacje o przedmiocie

| | |
|---|---|
| Semestr | 3 |
| Liczba punktów ECTS do zdobycia | 4 |
| Typ przedmiotu | obieralny |
| Język nauczania | polski |
| Sylabus opracował | • dr hab. inż. Marek Sawerwain, prof. UZ |

## Formy zajęć

| Forma zajęć | Liczba godzin w semestrze (stacjonarne) | Liczba godzin w tygodniu (stacjonarne) | Liczba godzin w semestrze (niestacjonarne) | Liczba godzin w tygodniu (niestacjonarne) | Forma zaliczenia |
|---|---|---|---|---|---|
| Wykład | 15 | 1 | 9 | 0,6 | Zaliczenie na ocenę |
| Laboratorium | 15 | 1 | 9 | 0,6 | Zaliczenie na ocenę |
| Projekt | 15 | 1 | 9 | 0,6 | Zaliczenie na ocenę |

## Cel przedmiotu

- Zapoznanie studentów z podstawowymi informacjami o równoległych i funkcyjnych technikach programowania,
- ukształtowanie wśród studentów zrozumienia i świadomości roli równoległych technik programowania, a także uwypuklenia zwiększającej się roli programowania funkcyjnego,
- nauka podstawowych umiejętności w zakresie tworzenia programów równoległych w systemach wieloprocesorowych opartych o tradycyjne uniwersalne procesory (CPU) a także o graficzne wieloprocesorowe układy ogólnego zastosowania (GPU),
- ukształtowanie podstawowych umiejętności w zakresie paradygmatu programowania funkcyjnego, a w szczególności roli funkcji i rekurencji, programowania bez efektów ubocznych oraz nabycie umiejętności używania techniki obliczeń leniwych.

## Wymagania wstępne

Metody programowania, Algorytmy i struktury danych, Teoretyczne podstawy informatyki, Logika dla informatyków

## Zakres tematyczny

Równoległy model obliczeniowy, klasy złożoności obliczeń równoległych.

Dostępne narzędzia pomagające realizować programy działające w środowiskach równoległych: MPI, OpenMP, CUDA, OpenCL.

Rodzaje prymitywnych operacji równoległych.

Zależność i podział danych, modele równoległych środowisk wykonawczych dla CPU oraz GPU.

Podstawowe konstrukcje funkcyjnego języka programowania na przykładzie języków OCaml, F#, Scala.

Typy danych, wyjątki, pojęcie obiektu.

Funkcje wyższego rzędu, model obliczeń programów funkcyjnych (w postaci uproszczonego opisu operacyjnego).

System typów, leniwe obliczenia oraz konstrukcje programowania równoległego.

Konstrukcje imperatywne w programowaniu funkcyjnym.

## Metody kształcenia

Wykład: wykład konwencjonalny/tradycyjny.
Laboratorium: ćwiczenia laboratoryjne, wg listy zadań.
Projekt: praca w grupach, metoda projektu.

## Efekty uczenia się i metody weryfikacji osiągania efektów uczenia się

| Opis efektu | Symbole efektów | Metody weryfikacji | Forma zajęć |
|---|---|---|---|
| Potrafi wykorzystywać dostępne rozwiązania programowania funkcyjnego, aby skracać czas implementacji wybranych rozwiązań informatycznych. | • K_K05 | • sprawdzian z progami punktowymi | • Wykład |
| Umie wykorzystać równoległe i rozproszone rozwiązania CPU i GPU we własnych programach. | • K_U14 | • projekt<br>• sprawozdanie z projektu | • Laboratorium<br>• Projekt |
| Rozumie podstawowe pojęcia programowania funkcyjnego oraz rolę instrukcji imperatywnych. | • K_W11 | • sprawdzian z progami punktowymi | • Wykład<br>• Laboratorium |
| Jest świadom dynamicznego rozwoju równoległych technik programowania i rosnącej roli programowania funkcyjnego. | • K_K01 | • sprawdzian z progami punktowymi | • Wykład |
| Zna model obliczeń równoległych stosowany w nowoczesnych rozwiązaniach sprzętowo-programowych. | • K_W09 | • sprawdzian z progami punktowymi | • Wykład<br>• Laboratorium |

## Warunki zaliczenia

Wykład - warunkiem zaliczenia jest uzyskanie pozytywnej oceny z egzaminu przeprowadzonego w formie pisemnej.
Laboratorium - warunkiem zaliczenia jest uzyskanie pozytywnych ocen ze wszystkich sprawdzianów pisemnych z ćwiczeń laboratoryjnych, przewidzianych do realizacji w ramach programu laboratorium.
Projekt - warunkiem zaliczenia jest wykonanie wszystkich zadań projektowych, przewidzianych do realizacji w ramach zajęć projektowych oraz przygotowanie pisemnego raportu ze zrealizowanego projektu.
Składowe oceny końcowej = wykład: 40% + laboratorium: 30% + projekt: 30%.

## Literatura podstawowa


1. Mattson G.T., He Y., Koniges E.A.:The OpenMP Common Core: Making OpenMP Simple Again, MIT Press, 2019.
2. Han J., Sharma B.: Learn CUDA Programming: A beginner's guide to GPU programming and parallel computing with CUDA 10.x and C/C++, Packt Publishing, 2019.
3. Syme D., Granicz A., Cisternino A.: F# 4.0 dla zaawansowanych, Wydanie IV, Helion 2017.
4. Sanders J., Kandrot E.: CUDA w przykładach. Wprowadzenie do ogólnego programowania procesorów GPU, Helion, 2012.
5. Michaelson G.: An Introduction to Functional Programming Through Lambda Calculus, Dover Publications Inc., 2011.
6. Gaster B., Howes L., Kaeli D. R., Mistry P., Schaa D.: Heterogeneous Computing with OpenCL, Morgan Kaufmann, 2011.
7. Pacheco P.: An Introduction to Parallel Programming, Morgan Kaufmann, 2011.
8. Sanders J., Kandrot E.: CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley Professional, 2010, english edition.
9. Czech Z.: Wprowadzenie do obliczeń równoległych, Wydawnictwo Naukowe PWN, 2010.
10. Herlihy M., Shavit N.: Sztuka programowania wieloprocesorowego, Wydawnictwo Naukowe PWN, 2010, edycja polska.
11. Smith C.: Programming F#, O'Reilly Media, Inc.,Sebastopol, USA, 2010.
12. Pickering R.: Foundations of F#, Apress,USA, 2007.


## Literatura uzupełniająca


1. Syme D., Granicz A., Cisternino A.: Expert F# 4.0, Apress, 2015.
2. Farber R.: CUDA Application Design and Development, Morgan Kaufmann, 2011.
3. Wen-mei W. Hwu, eds: GPU Computing Gems, Emerald Edition and Jade Edition, Morgan Kaufmann, 2011.
4. Harrop J.: F# for Scientists, John Wiley & Sons, Inc., Hoboken, New Jersey, USA, 2008.
5. Thomspon S.: Haskell - The Craft of Functional Programming, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1999.


## Uwagi

–

Zmodyfikowane przez dr hab. inż. Marek Sawerwain, prof. UZ (ostatnia modyfikacja: 26-04-2021 21:44)