# Parallel and functional programming techniques - opis przedmiotu

## Informacje ogólne

| | |
|---|---|
| Nazwa przedmiotu | Parallel and functional programming techniques |
| Kod przedmiotu | 11.3-WE-INFD-PaFPT-Er |
| Wydział | Wydział Informatyki, Elektrotechniki i Automatyki |
| Kierunek | WIEiA - oferta ERASMUS / Informatyka |
| Profil | - |
| Rodzaj studiów | Program Erasmus drugiego stopnia |
| Semestr rozpoczęcia | semestr zimowy 2018/2019 |

## Informacje o przedmiocie

| | |
|---|---|
| Semestr | 3 |
| Liczba punktów ECTS do zdobycia | 6 |
| Typ przedmiotu | obieralny |
| Język nauczania | angielski |
| 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 | - | - | Zaliczenie na ocenę |
| Laboratorium | 15 | 1 | - | - | Zaliczenie na ocenę |
| Projekt | 15 | 1 | - | - | Zaliczenie na ocenę |

## Cel przedmiotu

- Familiarize students with basic information about parallel and functional programming techniques.
- To shape understanding and awareness of the role of parallel programming techniques as well as highlight the increasing role of functional programming.
- To give basic skills in creating parallel programs for multi-core systems based on traditional processors (CPU) as well as graphics multi-core processors of general use.
- Learning of the basic skills in the functional programming paradigm, and in particular: the role of functions and recursion, programming without side effect and the use of the lazy computations method.

## Wymagania wstępne

Methods of Programming, Algorithms and Data Structures, Theoretical Foundations of Computer Science, Logic for Computer Scientists

## Zakres tematyczny

Theory of computation models: models of parallel computations and complexity classes.

Programmer tools: available tools for parallel programming for CUDA and OpenCL technologies.

Basic operations: Parallel primitive operations.

Data Dependency: dependency and division of data, models of execution of parallels environments for CPU and GPU.

Programming paradigm: Functional paradigm and basic constructions in selected functional languages e.g. OCaml, F#, Scala.

Basic data types: Data types in functional programming, exceptions and objects.

High-class function: first-class and high-order functions, functional model of computations (in a form of simplified operational description).

Type system and imperative control flow instructions: type systems, and lazy-computations, imperative features in functional programming languages.

## Metody kształcenia

Lecture: conventional lecture
Laboratory: laboratory exercises, group work
Project: project method, discussions and presentations

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

| Opis efektu | Symbole efektów | Metody weryfikacji | Forma zajęć |
|---|---|---|---|

| Opis efektu | Symbole efektów | Metody weryfikacji | Forma zajęć |
|---|---|---|---|
| Knows and understands the basics of functional paradigm and the role of imperative constructions in functional languages. | | • sprawdzian z progami punktowymi | • Wykład<br>• Laboratorium |
| Knows model of parallel programming used in modern hardware -- software computation systems. | | • sprawdzian z progami punktowymi | • Wykład<br>• Laboratorium |
| Can use available functional programming features to reduce implementation time of selected problems in computer science. | | • sprawdzian z progami punktowymi | • Wykład |
| Can use existing libraries which supports parallel programming techniques for CPU and GPU. | | • projekt<br>• sprawozdanie z projektu | • Laboratorium<br>• Projekt |
| Is aware of dynamic development of parallel programming techniques and rising role of functional programming. | | • sprawdzian z progami punktowymi | • Wykład |

## Warunki zaliczenia

Lecture - obtaining a positive grade in written exam.
Laboratory - the main condition to get a pass are sufficient marks for all exercises and tests conducted during the semester.
Project - a condition of pass is to obtain positive marks from all project tasks and preparation written report of project.
Calculation of the final grade: = lecture 40% + laboratory 30% + project 30%.

## Literatura podstawowa


1. Pickering R.: Foundations of F#, Apress, USA, 2007.
2. Smith C.: Programming F#.: O'Reilly Media, Inc., Sebastopol, USA, 2010.
3. Syme D., Granicz A., Cisternino A.: Expert F# , Apress, USA, 2015.
4. Sanders J., Kandrot E.: CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley Professional, 2010.
5. Gaster B., Howes L., Kaeli D. R., Mistry P., Schaa D.: Heterogeneous Computing with OpenCL, Morgan Kaufmann, 2011.
6. Pacheco P.: An Introduction to Parallel Programming, Morgan Kaufmann, 2011.
7. Rauber T., Rünger G.: Parallel Programming for Multicore and Cluster Systems, Springer Berlin Heidelberg, 2013.
8. Herlihy M., Shavit N.: The Art of Multiprocessor Programming, Morgan Kaufmann, 2012.
9. Farber R.: Parallel Programming with OpenACC, Elsevier Science & Technology, 2015.


## Literatura uzupełniająca


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


## Uwagi

Zmodyfikowane przez dr hab. inż. Marek Sawerwain, prof. UZ (ostatnia modyfikacja: 29-03-2018 13:00)