# Parallel and Functional Programming Techniques - course description

## General information

| | |
|---|---|
| Course name | Parallel and Functional Programming Techniques |
| Course ID | 11.3-WI-INFD-RówniFunkcTechProg-S15 |
| Faculty | Faculty of Computer Science, Electrical Engineering and Automatics |
| Field of study | Computer Science |
| Education profile | academic |
| Level of studies | Second-cycle studies leading to MSc degree |
| Beginning semester | winter term 2019/2020 |

## Course information

| | |
|---|---|
| Semester | 3 |
| ECTS credits to win | 4 |
| Course type | optional |
| Teaching language | polish |
| Author of syllabus | • dr hab. inż. Marek Sawerwain, prof. UZ |

## Classes forms

| The class form | Hours per semester (full-time) | Hours per week (full-time) | Hours per semester (part-time) | Hours per week (part-time) | Form of assignment |
|---|---|---|---|---|---|
| Lecture | 15 | 1 | 9 | 0,6 | Credit with grade |
| Laboratory | 15 | 1 | 9 | 0,6 | Credit with grade |
| Project | 15 | 1 | 9 | 0,6 | Credit with grade |

## Aim of the course

- 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

## Prerequisites

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

## Scope

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.

## Teaching methods

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

## Learning outcomes and methods of theirs verification

| Outcome description | Outcome symbols | Methods of verification | The class form |
|---|---|---|---|
| Potrafi wykorzystywać dostępne rozwiązania programowania funkcyjnego, aby skracać czas implementacji wybranych rozwiązań informatycznych. | • K_K05 | • a test with score scale | • Lecture |

| Outcome description | Outcome symbols | Methods of verification | The class form |
|---|---|---|---|
| Umie wykorzystać równoległe i rozproszone rozwiązania CPU i GPU we własnych programach. | • K_U14 | • a project<br>• sprawozdanie z projektu | • Laboratory<br>• Project |
| Rozumie podstawowe pojęcia programowania funkcyjnego oraz rolę instrukcji imperatywnych. | • K_W11 | • a test with score scale | • Lecture<br>• Laboratory |
| Jest świadom dynamicznego rozwoju równoległych technik programowania i rosnącej roli programowania funkcyjnego. | • K_K01 | • a test with score scale | • Lecture |
| Zna model obliczeń równoległych stosowany w nowoczesnych rozwiązaniach sprzętowo-programowych. | • K_W09 | • a test with score scale | • Lecture<br>• Laboratory |

## Assignment conditions

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%

## Recommended reading


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.: 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, edycja polska.
5. Sanders J., Kandrot E.: CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley Professional, 2010, english edition.
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. Czech Z.: Wprowadzenie do obliczeń równoległych, Wydawnictwo Naukowe PWN, 2010.
9. Herlihy M., Shavit N.: Sztuka programowania wieloprocesorowego, Wydawnictwo Naukowe PWN, 2010, edycja polska.


## Further reading


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.


## Notes

Modified by dr hab. inż. Marek Sawerwain, prof. UZ (last modification: 05-05-2019 10:43)