(faculty stamp)

COURSE DESCRIPTION

1. Course title: CONCURRENT PROGRAMMING	**2. Course code:** CCP
3. Validity of course description: 2016/2017	
4. Level of studies: 2nd cycle of higher education	
5. Mode of studies: intramural studies	
6. Field of study: MACROFACULTY	
7. Profile of studies: general academic	
8. Programme: COMPUTER SCIENCE (INFORMATICS)	
9. Semester: 1	
10. Faculty teaching the course: Institute of Informatics	
11. Course instructor: dr inż. Jacek Widuch	
12. Course classification: common courses	
13. Course status: compulsory	
14. Language of instruction: English	
15. Pre-requisite qualifications: It is assumed that the student has the basic knowledge of computer programming in the C/C++ language and problems presented in subjects of 1st cycle of higher education: Computer Programming, Algorithms and Data Structures.	
16. Course objectives: The course introduces students into the basic subjects of parallel computing and concurrent programming. The fundamental concepts of parallel computing, models of parallel computations and architectures of parallel computers, designing of parallel algorithms are discussed. Some libraries and programming languages supporting parallel computing are discussed. The lecture provides basic information that is then used in practice in laboratory and classes.	

17. Description of learning outcomes:

Nr	Learning outcomes description	Method of assessment	Teaching methods	Learning outcomes reference code
1	Student possesses advanced knowledge of models of parallel computations, basic parallel algorithms and designing of parallel algorithms.	Written exam, test on classes, laboratory exercises	Lectures, Classes, Laboratory exercises	K2A_W23, K2A_W29
2	Student possesses detailed knowledge of OpenMP standard.	Written exam, laboratory exercises	Lectures, Laboratory exercises	K2A_W23, K2A_W29
3	Student is able to use the library for thread management.	Written exam, laboratory exercises	Lectures, Laboratory exercises	K2A_U18
4	Student is able to use the methods for solving synchronization of parallel processes in the model with shared memory.	Written exam, test on classes, laboratory exercises	Lectures, Classes, Laboratory exercises	K2A_U08

5	Student is able to run parallel processes and designing of parallel algorithms and analyzing them.	Written exam, test on classes, laboratory exercises	Lectures, Classes, Laboratory exercises	K2A_U02, K2A_U10, K2A_U17

18. **Teaching modes and hours**
Lecture: 30 h., **Class:** 30 h., **Laboratory:** 30 h.

19. Syllabus description:

Lectures:

1. Definitions of parallel algorithm and concurrent process. Parameters of parallel algorithm (time complexity, speed-up, cost of the algorithm, efficiency).
2. Models of parallel computations and architectures of parallel computers. Super-computers with high performance.
3. Expressing concurrency: fork-join-quit statements, cobegin-coend block, parfor statement.
4. Correctness of parallel algorithms: deadlock, starvation, critical section, mutual exclusion.
5. Communication and synchronization of parallel processes in the model with shared memory. Synchronization objects: mutex, semaphore, monitor, conditional variable.
6. Communication and synchronization of parallel processes in the model with distributed memory: sending and receiving messages, synchronous and asynchronous communication, buffered communication, selective communication (guarded statements).
7. Fundamental problems of concurrent programming: the producer-consumer problem, the dinning philosophers problem, the readers-writers problem, the barrier synchronization.
8. Multithreading in C++11 standard.
9. The OpenMP standard.
10. The MPI standard.

Class:

During classes tasks with the following topics are solved:

1. Expressing the concurrency.
2. Correctness of parallel algorithms: deadlock, starvation, critical section, mutual exclusion.
3. The communication of parallel processes in the model with shared memory, the synchronization using mutexes and semaphores.
4. The communication of parallel processes in the model with shared memory, the synchronization using monitors.
5. The communication and the synchronization of parallel processes in the model with distributed memory.

Laboratory:

Laboratory exercises presents the practical related to communication and synchronization of parallel processes and threads. The following standards supporting parallel programming are presented:

1. Multithreading in C++11 standard.
2. The OpenMP standard.
3. The MPI standard.

20. Examination: yes

21. Primary sources:

1. T. Wittwer: *An Introduction to Parallel Programming*. VSSD, 1st edition, 2006.
2. P. Pacheco: *An Introduction to Parallel Programming*. Morgan Kaufmann; 1st edition, 2011.
3. A. Williams: *C++ Concurrency in Action: Practical Multithreading*. Manning Publications; 1st edition, 2012.
4. P.S. Pacheco: *Parallel programming with MPI*. Morgan Kaufman, 1997.
5. B. Chapman, G. Jost, R. van der Pas: *Using OpenMP*. MIT Press, 2008.
6. R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, J. McDonald: *Parallel programming In OpenMP*. Morgan Kaufamnn, 2001.
7. M. Ben-Ari: *Principles of Concurrent and Distributed Programming*. Pearson, 2nd edition, 2006.
8. Z. Czech: *Wprowadzenie do obliczeń równoległych*. Wydawnictwo Naukowe PWN, Warszawa 2010 (in Polish).
9. Z. Weiss, T. Gruźlewski: *Programowanie współbieżne i rozproszone*. WNT, Warszawa 1993 (in Polish).
10. M. Ben-Ari: *Podstawy programowania współbieżnego i rozproszonego*. WNT, Warszawa 1996 (in Polish).

22. Secondary sources:

1. I. Parberry: *Parallel Complexity Theory (Research notes in theoretical computer science)*. Financial Times Prentice Hall, 1987.
2. T. Tauber, G. Rünger: *Parallel Programming for Multicore and Cluster Systems*. Springer-Verlag Berlin Heidelberg, 2013.
3. G. Em Karniadakis, R.M. Kirby II: *Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and their Implementation*. Cambridge University Press; PAP/CDR edition, 2003.
4. B. Parhami: *Introduction to Parallel Processing: Algorithms and Architectures*. Springer US, 1999.
5. M. Herlihy, N. Shavit: *The Art of Multiprocessor Programming*. Morgan Kaufmann; 1st edition, 2012.
6. W. Gropp, E. Lusk, N. Doss, A. Skjellum: *A high-performance, portable implementation of the MPI message passing interface standard*. Parallel Computing, vol. 22, no 6, pp 789-828, 1996.
7. W. Gropp, E. Lusk: *User's Guide for mpich, a Portable Implementation of MPI*. ANL-96/6, Mathematics and Computer Science Division, Argonne National Laboratory, 1996.
8. Z. Weiss, T. Gruźlewski: *Programowanie współbieżne i rozproszone w przykładach i zadaniach*. WNT, Warszawa 1993 (In Polish).

23. Total workload required to achieve learning outcomes

Lp.	Teaching mode :	Contact hours / Student workload hours
1	Lecture	30 / 15
2	Classes	30 / 30
3	Laboratory	30 / 30
4	Project	
5	BA/ MA Seminar	
6	Other	0 / 15
	Total number of hours	90 / 90

24. Total hours: 180

25. Number of ECTS credits: 6

26. Number of ECTS credits allocated for contact hours: 1

27. Number of ECTS credits allocated for in-practice hours (laboratory classes, projects): 2

26. Comments:

Approved:

…….. ……..

(date, Instructor's signature) (date, the Director of the Faculty Unit signature)