COURSE SYLLABUS STEM Ed Abroad Program

Course Title: ***C/C++ programming***

Course Semester: Fall

University and Country: Adam Mickiewicz University; Poznan, Poland

Number of ECTS: 6 (lecture)

Course Designations for Transfer Credit: ECE209 (NCSU)

Content: Introduction to C/C++ programming. Students should get acquainted with all elementary language structures, should be able to write simple programs using input/output operations, pointers, dynamic memory allocation as well as working with objects and use of inheritance. Afterwards more emphasis would be put on improvement of programming skills. Learning of using exceptions, template library, creating and using dynamically linked libraries, system API calls and creating multithreaded programs.

Pre-requisites: Minimum knowledge of mathematics at the level of secondary school (lyceum). Preferable completed basic mathematics course (including algebra).

Aims: Module learning outcomes in terms of knowledge, skills and social competences and their reference to programme learning outcomes

Learning outcomes:

Learning outcomes symbol	Upon completion of the course, the student will:	Reference to programme learning outcomes
cpp_01	Know basic C keywords, build in data types, flow control instructions. Be able to write simple programs.	cpp_wu_01
cpp_02	Be able to write program using tables, stuctures, pointers, dynamic memory allocation, file i/o operations.	cpp_wu_02
cpp_03	Be able to define classes. Use user defined types in programs. Be able to use more complex data types. Use inheritance and virtual functions. C++ input output classes. Be able to use exceptions.	cpp_wu_03
cpp_04	Be able to use templates of functions and classes in programs. Know all C++ keywords.	cpp_wu_04
cpp_05	Know concepts of STL. Know basic notions, data structures and algorithms used in STL. Be able to write program using STL .	cpp_wu_05
cpp_06	Know how to write multithreaded programs.	cpp_wu_06
cpp_07	Know how to call system services directly from their own programs	cpp_wu_07

COURSE SYLLABUS STEM Ed Abroad Program

1. Learning content

Module title		
Learning content symbol	Learning content description	Reference to module learning outcomes #
TK_01	Compilation. Identifiers. Simple (build in) data types. Integral types, logical types, enumerated types, substitution of enumerated types, floating point types.	cpp_01
TK_02	Binary representation of data types. Errors, precision and stability. Integral types. Floating point types. Operators. Scope and visibility of variables	cpp_01
TK_03	Flow control instructions. Conditional instructions. Loops. Functions scanf and printf. Formatting strings in c. Operators ++ and —. Calling functions with arguments.	cpp_01
TK_04	Pointers. Tables. Structures. Anonymous structures. User defined types in C. Structure fields (members) access. Structures and function calls. Bit fields. Bit fields declaration. Limits of bit field use. Unions.	cpp_02
TK_05	Pointers declarations. Pointers arithmetic. Pointer conversion. Pointers to functions. Constant and pointers.	cpp_02
TK_06	Variable initialization. Input/output operations in C. Examples/Exercises: Matrix multiplication. Dynamic string table. „Pascal" style table allocation. Toss up simulation. Basic run time library string functions. List. Stack. Queue	cpp_02
TK_07	Declaration modifiers.	cpp_02
TK_08	Classes. Objects. Class declarations: class keyword, struct keyword, union keyword	cpp_03
TK_09	Inheritance. Class fields/members access. Access to parent and child class members. Virtual base classes. Friend functions	cpp_03
TK_10	Input output operations in C++. Classes: ofstream, ifstream, fstream. Operator >>. Operator << . Manipulators. Basic IOS.	cpp_03
TK_11	Constructors and destructors properties. Constructors: Default constructor, Copy constructor, Constructor overloading, Constructor call order. Class initialization. Single argument constructors. Destructors. Destructor calling, Destructors and atexit function and compiler directive #pragma exit. Destructors and exit function. Destructors and abort function. Virtual destructors.	cpp_03
TK_12	Keyword this. Static class members. Inline functions. Operator overloading. Unary operator overloading. Binary operator overloading. Substitution operator = overloading. Function call operator (). Overloading. Index operator [] overloading. Overloading of arrow -> operator. Definitions of increment and decrement operators. Changing definition of &&, || i , operators	cpp_03
TK_13	Virtual functions. Abstract classes. Polymorphic classes. Polymorphism can be treacherous. Exceptions. Missed destructor calls. Exceptions in constructors. Implicit type conversion in functions. new and delete operators: Operator new. Placement new operator. Operator delete. Tables.	cpp_03
TK_14	Macros. Nested parenthesis and commas. Token insertion using ##.	cpp_04

	Changing of variable identifier to string using #. Conditional compilation	
TK_15	Templates. Function templates. Function template overriding. Explicit and implicit function templates. Class templates. Templates arguments. Angle brackets in templates. Type safe generic lists. Pointer elimination in templates.	cpp_04
TK_16	Standard Template Library. STL history. STL and C++ ANSI/ISO standard. Nice classes. Functional object.	cpp_05
TK_17	STL overview: Containers. Vector. List. Double ended queue. Set. Map.	cpp_05
TK_18	Iterators. Input / output iterators. Unidirectional iterators. Bidirectional iterators. Random access iterators. Algorithms and functional object. How to create generic algorithm?	cpp_05
TK_19	Other STL related topics. Adaptors. Stack. Queue. Priority queue. Iterator Adaptors. Reverse Iterators. Insert Iterators. Raw Storage Iterator. Function Adaptors. Negators. Binders. Adaptors for pointers to functions. Allocators and memory use. Iterator Tags. Associative containers	cpp_05
TK_20	Mutex. Semaphore. Smart pointers.	cpp_06
TK_21	Threads. Thread created using Windows API. Thread created using VCL. Thread synchronization. Highert priority thread. Thread with critical section	cpp_06
TK_22	Directory search using Shell API. Wininet API. cURL library	cpp_07

Lecture component:

Recommended Books:

a. Kernigham Brain, Ritchie Dennis, „Język ANSI C" WNT Warszawa 1998
b. Stroustrup, Bjarne „Język C++" wydanie 4 WNT Warszawa 1998. Oryginał „The C++ programming language" -- 2nd ed. Addison-Wesley 1993
c. Eckel Bruce, "Thinking in C++. Edycja polska." Helion. Gliwice. Na stronie internetowej http://www.ibiblio.org/pub/docs/books/eckel/ dostępne są (za darmo) oryginalne wersje tej oraz innych książek autora „Thinking in C.", „Thinking in Java".
d. Vandevoorde David, Josuttis Nicolai M., „C++ Szablony." Helion Gliwice 2003. Oryginał "C++ templates: The complete Guide." Pearson Education Inc. część Addison Wesley 2003.
e. Meyers Scott, „Język C++ bardziej efektywny" Wydawnictwa Naukowo-Techniczne, Warszawa1998. Oryginał „More Effective C++. 35 New Ways to Improve Your Programs and Designs" Addison-Wesley 1996.
f. Josuttis Nicolai M., „C++ biblioteka standardowa. Podręcznik programisty." Helion Gliwice 2003. Oryginał „The C++ Standard Library: A Tutorial and Reference" Pearson Education Inc. część Addison-Wesley 1999.
g. Andrew Koenig, Barbara E. Moo, „Accelerated C++. Practical Programing by Example", Pearson Education część Addison-Wesley Professional 2000. Polskie tłumaczenie „C++ potęga języka" Helion Gliwice 2004
h. Grębosz Jerzy, „Symfonia C++" tomy 1 -- 3, Oficyna Kallimach. Oraz inne książki tego autora poświęcone językom C i C++.

i. Hansen Tony, „C++ zadania i odpowiedzi" wydanie drugie WNT Warszawa 1994. Oryginał „The C++ Answer Book" Addison-Wesley 1990
j. Plauger P. J., „Biblioteka standardowa C++" WNT Warszawa 1997. Oryginał „The Standard C++ Library" Prentice Hall, Inc.
k. Lee Meng, Stepanov Alexander "The Standard Template Library" HP Labaratories, 1501 Page Mill Road, Palo Alto, CA 94304 February 1995
l. Compiler documentation

Instructors: Michał Mucha PhD with consultation by Dr. Greg Byrd (NC State University).

Grading System and Percentage Contribution

A. Lecture assessment

Homework	12%
Problem Sessions	10%
Exam (3 @16% each)	48%
Final Exam, Cumulative	30%
Total	100%

AMU Grading system and scale

The grading system used at Adam Mickiewicz University, whose name is abbreviated as AMU or UAM, is as follows:

Tests, exams, homework assignments grading scale

5 100%-91%
4+ 90%-86%
4 85%-76%
3+ 75%-71%
3 70%-60%
2 59% and less

This translates into the following ECTS (European internationally recognized system) grading scale:

ECTS Grade	AMU grade	Definition
A	5.0	EXCELLENT – outstanding performance with only minor errors
B+	4+ / 4.5	VERY GOOD – above the average standard but with some errors
B	4.0	GOOD – generally sound work with a number of notable errors
C+	3+ / 3.5	SATISFACTORY – fair but with significant shortcomings
C	3.0	SUFFICIENT – performance meets the minimum criteria
F	2.0	FAIL – considerable further work is required

Hours: 3 Lecture hours per week.

I. Additional information

1. Reference of learning outcomes and learning content to teaching and learning methods and assessment methods

Module title

Symbol of module learning outcome*	Symbol of module learning content#	Methods of teaching and learning	Assessment methods of LO achievement&
cpp_01	TK_01, TK_02, TK_03	Lecture, computer lab.	Inspection of exercises completion
cpp_02	TK_04, TK_05, TK_06, TK_07	Lecture, computer lab.	Inspection of exercises completion
cpp_03	TK_08, TK_09, TK_10, TK_11, TK_12, TK_13	Lecture, computer lab.	Inspection of exercises completion
cpp_04	TK_14, TK_15	Lecture, computer lab.	Inspection of exercises completion
cpp_05	TK_16, TK_17, TK_18, TK_19	Lecture, computer lab.	Inspection of exercises completion
cpp_06	TK_20, TK_21	Lecture, computer lab.	Inspection of exercises completion
cpp_07	TK_22	Lecture, computer lab.	Inspection of exercises completion
All	All	Final exam	Program/project + oral exam

It is advisable to include assessment tasks (questions).

2. Student workload (ECTS credits)

Module title:	
Activity types	Mean number of hours* spent on each activity type
Contact hours with the teacher as specified in the programme	Lectures 30, practical classes 30
Reading	40
preparation for classes	Highly individual. Impossible to asses >= 5
Writing programs	Highly individual. Impossible to asses >= 15
Exam preparation	Highly individual. Impossible to asses
Total hours	60 + not less then 60
Total ECTS credits for the module	5

** Class hours – 1 hour means 45 minutes*

#Independent study – examples of activity types: (1) preparation for classes, (2) data analysis, (3) library-based work, (4)writing a class report, (5) exam preparation, etc.

3. Assessment criteria
   Knowledge and understanding of language constructs. Ability to use them in (student's) own programs. Ability to solve problems.