Module Name **Programming in a Scientific Environment with C++**						
Type of Module Advanced Module			**Module Code** AM-PSE			
Identification Number	**Workload**	**Credit Points**	**Term**	**Offered Every**	**Start**	**Duration**
MN-CS-PSE	270 h	9 CP	1. - 3. Semester	SuSe	Summer Term only	1 Semester

1	**Course Types**	**Contact Time**	**Private Study**	**Planned Group Size**
	a) Lecture	30 h	90 h	<20 Students
	b) Seminar	15 h	30 h	
	c) Exercise	15 h	90 h	

2	**Module Objectives and Skills to be Acquired**
	Object Oriented Programming, Design Principles Classical Computers, Molecular Dynamics/Hartree-Fock Implementation

3	**Module Content** Part I (Lecture/Exercise) - Introduction - Capabilities: »computers vs. humans« - Artificial »Intelligence«, Quantum Computing vs. classical programming - Programming paradigms, programming languages, Turing completeness - Interpreter, compiler, just in time compilation - Efficiency: language vs. algorithm vs. math - Why C++? - First steps - »Hello world« - Error messages... - Variables and (builtin) types - Loops and if clauses - Expressions - Type safety, type propagation - Subroutines - Templates: variable types (static polymorphism) - Function and class templates - Standard template library (STL) containers - vector, array, list, set, map, ... - Access, insert/delete complexities - Explicit and implicit instantiation - Algorithms and their complexities - Nesting of templates - Idea of object orientation (OO) - 3 steps: - Group builtin types together to create new abstract types (data structures) - Group abstract types with methods - Introduce life cycle governing methods (constructors, destructors, ...) - Separation of interface and implementation - Introduction of custom operators - Access control - Inheritance - Virtual functions (dynamic polymorphism) - UML diagrams - Programming 5. Technical issues - Organizing »large« projects - File splitting (header and implementation files) - Automatizing build process (make) - »Compiler« pipeline - Preprocessor, compiler, assembler, linker - Involved stages, files, and linker symbols - Interoperability with other languages - Floating point issues - Limits of computational power - Latency, bandwidth, caches - Basis linear algebra subroutines and their levels BLAS1-3 - Cache-aware...

4	**Teaching Methods** Lecture, Practical exercises, autonomous experiments, journal creation
5	**Prerequisites (for the Module)** Formal: None With respect to the contents: Basic Programming Skills
6	**Type of Examination** Passed oral exam, testified written report; the quality of the report is also included in the grading of the oral exam
7	**Credits Awarded** The module is passed by passing an oral examination. The grade given for the module is equal to the grade of the oral examination.
8	**Compatibility with other Curricula** The course is part of the Master of Science Chemistry
9	**Proportion of Final Grade**
10	**Module Coordinator** M. Hanrath
11	**Further Information** teaching language: English The Module is based on the Experimental Module of the MSc Chemistry. As such, it is designed as a seven-week course with daily contact times.