# Course Analysis: SF2565, FSF3565, Program Construction in C++ for Scientific Computing, 2021

**Course Data**

- Program Construction in C++ for Scientific Computing, SF2565, FSF3565, 7.5 ECTS
- Period 1/2, 2021/2022
- Responsibility: Michael Hanke
- Teaching hours:
  - Lectures/exercises: 24+8 h
  - Computer labs: 2 h
- Registered students: 26 + 8 PhD students (including 2 students from SU)
- Literature: Lippman/Lajoie/Moo, C++ Primer, 5th ed., Skansholm, C++ direkt, 3:e uppl, lecture slides
- Credits:
  - homework: 3.5 ECTS
  - Written examination: 4 ECTS
- Graduation rate (undergraduate students): 76.9%

**Aim** The course provides an introduction to the C++ language both for users and developers of classes with a special emphasis on problems in Scientific Computing. Special care is put on efficient programming. The language features are developed using examples from the numerical solution of partial differential equations.

**Changes compared to the last year** The most important change for this and the previous years has been that the course was given completely and partly, respectively, remotely because of the Covid-19 pandemic. This includes even exercises and lab sessions.

Additionally, a section on callables including functors and lambda expressions has been added.

**Conclusions** The questionnaire has been answered by 5 out of 34 participants. So the conclusions based on them should be taken with a grain of salt.

Generally, the course was considered interesting and meaningful.

The most important observation in the answers compared to the last years is that the shift to remote lectures and exercises changed the kind the students are comprehending the information. The lectures and exercises where previously mainly organised (at least in important parts) as having a dialogue during the sessions. This includes even live programming experiments. The remote situation, however, made all of them mostly to a one-way kind of communication. Even if live quizzes are used at certain points, a real discussion was almost impossible to start. As a lecturer, I was missing live feedback. This can clearly seen in the estimation of the lectures and the lecture slides which went down from very good to good-acceptable. Accordingly, the percentage of listeners to lectures went down. This is in sharp contrast to live lectures which were highly estimated and heartily recommended by the former participants.

These are some of the opinions of the questionnaire (Note: in this edition, both remote and on-campus lectures have been used):

- "I think the professor tried his best to be available and keep the lectures good, but it is still very difficult to give an interesting lectures remotely. It became quite dry to just go through a bunch of slides and hard to stay focused"
- "the in person lectures were way more fun and interesting"
- "Michael Hanke did a good job on delivering the material remotely – I preferred in person though; Michael seemed to enjoy teaching in person more, and it helped me focus on the task at hand, better!"

Despite the more requiring situation for both students and teacher, the examination rate is above the long-term average.

The numerical parts (structured grids and finite difference operators on structured grids) are usually considered as hard to understand. In the present edition, the answers did not show this. Instead, language feature like rvalue references and move operators as well as templates have been considered hardest to understand.

It it was emphasized in the answers that the running example was a very good motivation to show the benefits of C++ and the benefits of using it. This is emphasized in the homework evaluation. In order to cite one representative answer: "I think the last one [homework] was the most rewarding because it was so nice when all the puzzles worked together".

In the most advanced parts, a moderate fluency in programming was required which led to widely varying quality of the implementations.

In a programming course, where many code snippets are shown, it is unavoidable to use slides extensively. They will be commented on heavily by the teacher and discussed with the students. The lecture slides shall be considered as a skeleton for the notes taken by the students during the lectures. Therefore, they are published well in advance such that they can be downloaded for making notes. However, for pedagogical reasons, the information on them was not exhaustive. As indicated previously, in a remote situation, these ideas do not seem to be sufficient. Since there is no course book in the traditional sense available, comprehensive lecture notes should be provided. For the time being, the internet is the most important source of information.

**Teaching** The teaching was done by lectures, excercises, and one computer lab. The latter was intended for students not compfortable with the linux operatinmg system and the GNU Compiler Suite to get started. An IDE was introduced (Code-Blocks). Homeworks have been evaluated during lectures or excercises. According to the answers, the course activities where definitely of help to reach the learning outcomes.

**Examination** The examination based on homework problems and a written examination. A successfully solved project 4 gave bonus credits for the written examination. According to the students' questionnaire, homework and examination reflected the course's goals very well. The level of the homeworks was estimated very different: ranging from too simple in the beginning to too advanced in the later papers. This indicates that the level of programming skills was rather wide.

**Prerequisites** With the exception of certain programming skills, no problem. This concerns, in particular, experiences with developing more complex programs.

**Planned changes** For the future, comprehensive lecture notes should be prepared. It should be considered if the final examination should be changed to a project.

**Grading** No problems.