Dr. Ole Klein, Stefan Meggendorfer
IWR, Heidelberg University

Exercise Sheet 12

Note: This is the last exercise sheet. It should give you a (non-exhaustive) overview over the lecture and may serve as a starting point for your preparation for the exam. You do not have to submit your solution to this exercise sheet and it will not be discussed in the tutorials.

Exercise 1: C++ Quiz 6 (0 points)

Here are some additional questions from https://cppquiz.org for you to answer:

Question 1: https://cppquiz.org/quiz/question/126 (unqualified lookup)

Question 2: https://cppquiz.org/quiz/question/184 (method shadowing)

Question 3: https://cppquiz.org/quiz/question/18 (private virtual function)

Question 4: https://cppquiz.org/quiz/question/264 (consequences of defaulted constr.)

Question 5: https://cppquiz.org/quiz/question/112 (default arg of virtual function)

Question 6: https://cppquiz.org/quiz/question/250 (overloaded variadic functions)

Question 7: https://cppquiz.org/quiz/question/109 (deduced and nondeduced context)

These are questions that shine some light on the hidden depths of C++. Neither the questions nor their answers are in any way relevant for the exam, and the correct answer may range from surprising to confusing. This means you don't have to look at them, unless you are interested, of course. Here be dragons.

Exercise 2: Exam Preparation (0 points)

The exam will contain several different types of exercises, ranging from programming exercises where you have to provide a complete implementation, to code snippets which you have to explain, discuss or correct, to smaller one-shot questions and longer writing prompts about features of C++ and programming language constructs.

The following list of questions and tasks can be seen as an invitation to revisit the constructs discussed in the different parts of the lecture and make sure that you have a solid grasp of both the theoretical underpinnings and the concrete implementation in C++. Most questions are answered in the lecture notes, but in some cases it may be helpful to look online.

Fundamentals:

1. What are generic functions, and how is the corresponding construct in C++ called?
2. Write an example code using a template template parameter.
3. What exactly is object oriented programming, how does it differ from functional programming?
4. Discuss benefits and downsides of object oriented programming and functional programming using the example of numerical integration.
5. What is the difference between a class and a struct in C++?
6. What are the main differences between C and C++?
7. What does the keyword `static` mean, both in functions and in classes?

8. When do you need the keyword `explicit`, and why?
9. What are the “rule of zero” and the “rule of five”? Are they sensible?
10. Explain what inheritance is and why it is useful.
11. Write an example code using encapsulation and inheritance.

Standard Library:

12. Discuss pros and cons of different STL container types.
13. What is an iterator, and why is this concept important for STL algorithms?
14. What is the purpose of exceptions, and how do they work?

Advanced Topics:

15. Explain the purpose of template specializations through an example.
16. What are function overloads? How do they interact with template specializations?
17. What is a memory leak, and what are ways to find it?
18. Give a simple example of a situation where a memory leak will occur, and how it can be resolved.
19. What does the acronym RAII stand for, and why is RAII important?
20. Explain template metaprogramming and give a code example.
21. What are virtual methods, and what are they used for?
22. What is an abstract base class, and what is a derived class? Where are these concepts useful?
23. Discuss dynamic vs. static polymorphism and their use cases.
24. What does the abbreviation SFINAE stand for?
25. Explain how SFINAE works by providing a simple example.
26. What is `std::enable_if` and where is it useful, and what are `decltype` and `declval`?

C++11:

27. What are the key features introduced by C++11?
28. What does the keyword `auto` do, and when should it be used?
29. What is a trailing return type? When are they needed?
30. Where could the keyword `constexpr` be useful? Provide a code example.
31. What are lvalues and rvalues, and why is there a distinction?
32. Explain move semantics, and what `std::move` does.
33. What are smart pointers, and why were they introduced?
34. Explain the differences between `std::unique_ptr`, `std::shared_ptr` and `std::weak_ptr`.
35. Are there any benefits in using raw pointers instead of smart pointers?
36. What are lambda expressions? What are they useful for?
37. Give an example of using a capture default inside a lambda expression.
38. What is a variadic template, and what is a parameter pack?
39. Discuss the different ways of initialization.
40. What are range-based loops, and why were they introduced?

41. What is a type trait, and where is it used? Give an example.

C++14 and Beyond:

42. What was introduced by C++14 and C++17, and what does C++20 introduce?
43. Why are the changes to `constexpr` in C++14 important?
44. Print the first few prime numbers using C++14 `constexpr` functions, i.e., non-recursively.
45. In what ways do generic lambdas improve on normal templates?
46. What is a potential use case for variable templates?
47. What does guaranteed copy elision mean? And what are the consequences?
48. How can `if constexpr` serve as a replacement for SFINAE?
49. What are concepts, and why were they introduced? Discuss applications.

Assorted Questions:

50. What are possible ways to implement a number sequence like the Fibonacci numbers? Discuss benefits and downsides.
51. What would be a good design for storing a matrix in C++?
52. How would you implement a numerical matrix class? Discuss the advantages and drawbacks of the approaches that were introduced.
53. How can time measurements be done, using Linux tools and from within your program?
54. What are use cases of threads and MPI, and how do they differ?
55. Try to solve 5 random C++ Quiz questions in a row without giving up or guessing wrong.