1. Problems of C language, motivation for C++. Encapsulation. Access modifiers. Classes and Structs. Invariants of a class.
2. Const modifier for types (including pointers and references), functions, class functions.
3. Object lifetime. new and delete operators. Constructors and destructors. Explicit constructors. Member initializer list.
4. lvalue references. Difference from pointers. const references. Temporary objects and const lvalue references to them.
5. Initialization in C++: default, value, direct, copy and aggregate. Comparison with initialization in C language.
6. Copy constructors: its form, use cases and motivation. RVO/NRVO. Copy assign operator. Rule of 3.
7. Value categories in C++. lvalues, prvalues and xvalues. Examples. rvalue references. Typical use cases. std::move: motivation and pitfalls.
8. Move semantics. Move constructors, move assign operators. copy-and-swap idiom. Rule of five.
9. RAII idiom. Motivation (including exceptions). Scoped pointers example, two variants of its semantics. Concept of ownership.
10. Casts in C++: C-style casts, static\_cast, const\_cast, reinterpret\_cast. Overloading conversion operators. Explicit casts operators. Casts vs constructors.
11. Operators overloading. Binary and unary operators overloading example. Indexing operator. \* and -> operators. Operators that can’t be overloaded.
12. Friends: functions and classes. Semantics, motivation and examples.
13. Inheritance in C++. Types of inheritance in C++. Examples of good and bad class hierarchies. Liskov substitution principle. Subtyping polymorphism in C++. Private inheritance as composition.
14. Virtual functions in C++, pure virtual functions, abstract classes. Virtual destructors. Early and late binding conceptions.
15. Virtual function implementation. VMT. Virtual calls in constructors and destructors. Adjusting for pointers, static\_casts for down casts.
16. Multiple inheritance in C++. Motivation. Object layout. Adjusting pointers during casts.
17. Diamond inheritance. Virtual inheritance. Drawbacks of such a solution. Examples of hierarchies where virtual and nonvirtual diamond inheritance are needed.
18. RTTI, influence on object layout, dynamic\_cast, typeid. Dynamic cast and multiple inheritance. Pros and cons of RTTI.
19. Exceptions: basic syntax and mechanics, throwing by value, by reference or by pointer, standard exceptions. Stack unwinding. Exceptions in constructors and destructors. Safe exceptions philosophy.
20. Generic programming and templates in C++. Syntax, basic mechanics and implementation. Monomorphization. Template specialization.
21. Template classes. Lazy instantiation of class templates. non-type argument. Templates vs inheritance. Static polymorphism. CRTP.
22. Templates implementation. 2 phase names lookup. SFINAE. enable\_if.
23. SFINAE based metaprogramming: intergral\_constant, compile time and modifiers checkers. Compile-time execution (without constexpr or consteval), examples.
24. Iterators in C++; Ranged-based for; initializer\_list;
25. Types deduction for template arguments. Example of ambiguous declarations. Cutting of references and const. Class template argument deduction.
26. auto modifier. Rules of deduction for auto. Examples: ranged-based for with auto. decltype. auto return type. auto arguments types.
27. Reference collapsing. Universal (forwarding) references. Examples of usage and perfect forwarding.
28. Lambdas in C++: basic syntax and implementation. Closure and capturing this. Applications: comporators, generators, functional-style work with collections.
29. Variadic templates. Parameter packs. Recursive functions for handling variadic templates, specialization and constexpr-if. Folding, and expanding the packs. Perfect forwarding for variadic templates and emplace\_back.
30. Constexpr: expression, function and constexpr-if. Difference from const. Consteval. Custom literal types. Compile-time generation of arrays.
31. Requires (modifier and expression) and concepts. Example of constraint for random-access container.
32. Smart pointers. Problems of raw pointers and motivation. Unique pointers and RAII. Custom deleters. Using unique\_ptr in fields instead of raw pointers.
33. Smart pointers: shared pointers. Motivation, syntax and implementation. Difference between make\_shared and just creating shared\_ptr. enable\_share\_from\_this. Cyclic references and weak\_ptr.