вопросы по курсу ООП **0x7E8**

1. **Ручное управление памятью.** Выделение памяти на стеке. Опасности при возвращении адресов локальных переменных. Отличие указателей от массивов. Адресная арифметика. Динамическое выделение памяти на куче.
2. **Ссылки**. Lvalue и rvalue ссылки. Отличие указателей и ссылок. Операции с ссылками. Константные ссылки и константные указатели. Move семантика.
3. **Исключения в C++.** std::exception и std::exception_ptr. std::current_exception и std::rethrow_exception. Способы перехвата исключений в catch. Модификатор noexcept. Плюсы и минусы использования исключений.
4. **Объект и его свойства**. Классы объектов. Пространства имен. Жизненный цикл объектов. Конструкторы и деструкторы. Константные классы и функции.
5. **Понятие объектно-ориентированного языка.** Инкапсуляция. Жизненный цикл классов и объектов. Конструкторы и деструкторы. Статические методы и члены классов.
6. **Понятие объектно-ориентированного языка**. Наследование в C++. Полиморфизм. Таблица виртуальных функций. Вызов конструктора и методов родительского класса. Множественное наследование. Плюсы и минусы полиморфизма.
7. **Указатели и ссылки**. LValue и RValue ссылки. Move-семантика. Константы в C++ (переменные и функции), мутанты.
8. **Перегрузка операторов.** Синтаксис. Возможные к перегрузке операции. Перегрузка ++ и --. Перегрузка бинарных операторов.
9. **Перегрузка операторов.** Перегрузка оператора =. Move семантика. Запрет операторов копирования. Перегрузка (). Функтор.
10. **Шаблоны классов и функций.** Параметры шаблонов. Специализация шаблонов.
11. **Шаблоны классов и функций.** Метафункции. Метафункция возвращающая разный тип данных в зависимости от условия.
12. **Шаблоны классов и функций.** SFINAE. std::enable_if. Type traits.
13. **Шаблоны классов и функций.** Concept. Variadic template. Хвостовая рекурсия на стадии компиляции.

14. **Идиома RAII.** Умные указатели. Шаблон std::unique_ptr. Пример реализации.
15. **Идиома RAII.** Умные указатели. std::shared_ptr. Наследование и std::shared_ptr. Проблемы при использовании std::shared_ptr. Шаблон std::weak_ptr.
16. **Итераторы.** Концепция, требования к forward итератору. Последовательные контейнеры в STL.
17. **Итераторы.** Концепция, требования к forward итератору. Ассоциативные контейнеры в STL
18. **Аллокаторы памяти.** Перегрузка оператора new. Простой аллокатор памяти на массиве. Allocator traits.
19. **Проектирование структуры классов**. Характеристики. Coupling vs cohesion. Виды cohesion. Способы улучшения структуры классов.
20. **Способы улучшения структуры классов**. Pimpl. TDA. CQS. Закон Джона Постеля.
21. **SOLID**: Принцип единой ответственности. Пример.
22. **SOLID**: Принцип открытости/закрытости. Пример.
23. **SOLID**: Принцип подстановки Барбары Лисков.
24. **SOLID**: Принцип разделения интерфейсов.
25. **SOLID**: Dependency Inversion Principle.
26. **Шаблон** factory method. Шаблон abstract factory. Шаблон builder.
27. **Шаблон** visitor, шаблон memento. Шаблон Observer.
28. **Лямбда выражения.** Функция как параметр, функтор, std::function. Списки захвата лямбда функций. Использование лямбда-выражений в стандартных алгоритмах (std::transform, std::for_each …).
29. **std::thread.** Функции и функторы как параметры. Использование семантики перемещения в std::thread. Функциии из пространства имен std::this_thread. Идиома RAII и std::thread.
30. **Потокобезопасность**. Реентерабельность. Race condition. Взаимное исключение. std::mutex и std::recursive_mutex. std::lock_guard и std::unique_lock. Реализация потокобезопасного стека. dead_lock. Просачивание данных за пределы lock_guard
31. **std::future и std::async std::promise**.Условные переменные. Закон Амдала.
32. **Ассинхронное программирование**. Понятие цикла обработки сообщений. Coroutine.