вопросы по курсу ООП **0x7E3**

1. **Ручное управление памятью.** Выделение памяти на стеке. Опасности при возвращении адресов локальных переменных. Объект и его свойства. Классы объектов. Пространства имен. Жизненный цикл классов. Конструкторы и деструкторы.
2. **Указатели и ссылки**. LValue и RValue ссылки. Move-семантика. Константы в C++ (переменные и функции), мутанты.
3. **Перегрузка операторов.** Синтаксис. Возможные к перегрузке операции. Перегрузка ++ и --. Перегрузка бинарных операторов.
4. **Перегрузка операторов.** Перегрузка оператора =. Запрет операторов копирования. Перегрузка (). Функтор. constexpr. Пользовательские литералы.
5. **Понятие объектно-ориентированного языка**. Объектно-ориентированная декомпозиция. Базовые понятия ООП: абстрагирование. Абстракция сущности, поведения, виртуальной-машины, произвольная. Наследование в C++. Полиморфизм.
6. **Понятие объектно-ориентированного языка.** Инкапсуляция. Жизненный цикл классов и объектов. Конструкторы и деструкторы.
7. **Понятие объектно-ориентированного языка.** Протечка абстракции. Примеры, способы устранения.
8. **Исключения в C++.** std::exception и std::exception_ptr. std::current_exception и std::rethrow_exception. Модификатор noexcept. Метод terminate
9. **Шаблоны классов и функций.** Сравнение наследования и шаблонов. Параметры шаблонов. Специализация шаблонов.
10. **Шаблоны классов и функций.** Метафункции. Метафункция возвращающая разный тип данных в зависимости от условия.
11. **Шаблоны классов и функций.** SFINAE. std::enable_if.
12. **Шаблоны классов и функций.** Variadic template. Хвостовая рекурсия на стадии компиляции.
13. **Шаблоны классов и функций.** Реализация tuple.
14. **Лямбда выражения.** Функция как параметр, функтор, std::function. Списки захвата лямбда функций. Использование лямбда-выражений в стандартных алгоритмах (std::transform, std::for_each …).
15. **Идиома RAII.** Умные указатели. Недостатки «обычных» указателей. Шаблон std::unique_ptr. Семантика перемещения. std::move
16. **Идиома RAII.** Умные указатели. Недостатки «обычных» указателей. std::shared_ptr. Наследование и std::shared_ptr. Проблемы при использовании std::shared_ptr. Шаблон std::weak_ptr.
17. **Контейнеры в C++.** std::array, std::vector. Итераторы.
18. **Контейнеры в C++.** std::stack, std::queue, std::deque. Итераторы.
19. **Контейнеры в C++.** std::forward_list, std::list. Итераторы.
20. **Контейнеры в C++.** std::unordered_set, std::unordered_map. Итераторы.
21. **Контейнеры в C++.** std::unordered_multiset, std::unordered_multimap. Итераторы.
22. **Аллокаторы памяти.** Перегрузка оператора new. Простой аллокатор памяти на массиве.
23. Проектирование структуры классов. Характеристики (жесткость, хрупкость, повторное использование. Способы улучшения структуры классов. SOLID: Принцип единой ответственности. Пример.
24. **Виды связанности.** Метрика cohesion. Метрика coupling. SOLID: Принцип открытости/закрытости. Пример.
25. Шаблон проектирования template method (обычный и CRTP). Шаблон проектирования strategy.
26. **SOLID**: Принцип подстановки Барбары Лисков.
27. Принцип TDA. Принцип Command Query Separation.Закон Постеля.
28. **SOLID**: Принцип разделения интерфейсов.
29. **SOLID**: Dependency Inversion Principle.

материалы лекций: https://github.com/OOP4MAI

30. **Мультипроцессирование и мультипрограммирование**. SMP, MPP и NUMA. Мультипрограммирование. Вытесняющая и не вытесняющая многозадачность. Планировщик задач. Жизненный цикл потока.
31. **std::thread**. Функции и функторы как параметры. Использование семантики перемещения в std::thread. Функциии из пространства имен std::this_thread.
32. **Потокобезопасность**. Реентерабельность. Race condition. Взаимное исключение. std::mutex и std::recursive_mutex.
33. std::lock_guard и std::unique_lock. Реализация потокобезопасного стека. dead_lock. Просачивание данных за пределы lock_guard
34. std::future и std::async std::promise.Условные переменные. Закон Амдала.
35. **Неблокирующие алгоритмы**. Атомарные типы. CAS операции. Потокобезопасный стек на CAS-операциях.