SFINAE

type_traits, serialization, iostreams

Цели и задачи занятия

- Попрактиковаться в использовании SFINAE для реализации собственного алгоритма выбора перегрузки шаблонной функции
- Реализовать библиотеку для сериализации/десериализации контейнеров и POD типов с возможностью раширить этот набор пользовательскими типами

Сериализация

- В программе имеется некоторый объект
- Состояние объекта сохраняется в блок байтов/текста/… (сериализация)
- Из этого блока можно восстановить эквивалентный объект (десериализация)
- Эквивалентность, как и оператор == определяются каждым классом самостоятельно

Тесты и стартовый код

https://drive.google.com/folderview?id=0B-KXDlig5kwVU0ZWMW9xSTEyWjA&usp=sharing

Задача 1

- В пространстве имен **serialization** реализуйте функции **serialize**(std:: ostream& out, T& obj) и **deserialize**(std::istream& in, T &obj), которые могут сериализовать и десериализовать любые POD типы
- Если T - не POD тип, то покажите ошибку компиляции “Type is not serializable”
- Пример вызова serialize, deserialize смотрите в тестах
- Старайтесь по ходу работы сохранять свой код компактным. Очень легко будет сделать много функций, в которых сложно ореинтироваться

Задача 2

- Поддержите в функциях **serialize** и **deserialize** возможность сериализовать контейнеры POD типов
- Контейнер - класс, внутри которого определен тип iterator

Задача 3

Добавьте в **serialize** и **deserialize** поддержку контейнеров, состоящих из элементов, уже поддерживаемых любой из имеющихся реализаций **serialize** и **deserialize**

Задача 4

Попробуйте сериализовать std::map с не POD ключом. Добавьте поддержку сериализации std::map и std::unordered_map в вашу библиотеку сериализации

Задача 5

Добавьте возможность сериализовывать произвольные пользовательские типы с помощью написания для него функции serialize, которая сама выберет какие поля пользовательского типа сериализовывать, а какие нет

Ссылка на код

TODO