Контрольные вопросы:

- ☐ (5 б.) Приведите примеры ситуаций, в которых удобно использовать вариативные шаблоны.
- ☐ (5 б.) Как можно обработать по очереди все аргументы из пакета аргументов функции?
- ☐ (5 б.) Как вычислить количество параметров в пакете параметров вариативного шаблона?
- ☐ (5 б.) Какие существуют разновидности выражений свертки и когда они применяются?
- ☐ (5 б.) В чем заключается разница между динамическим и статическим полиморфизмом?

Упражнения:

- ☐ (25 б.) Реализуйте вариативный шаблон функции, вычисляющей общий объем памяти в байтах, занимаемый пакетом ее аргументов. Внутри функции используйте выражение свертки, «обычный» оператор sizeof и оператор + в роли op в свертке. Также продемонстрируйте вариант без использования выражения свертки.
- ☐ (25 б.) Реализуйте вариативный шаблон функции, которая создает динамический объект указанного пользователем типа при помощи оператора new и возвращает указатель на него. Тип явно указывается пользователем как параметр шаблона. Пакет аргументов функции должен передаваться конструктору этого типа.
- ☐ (25 б.) Реализуйте вариативный шаблон функции, которая принимает произвольный вызываемый объект, например, лямбда-функцию или обычную функцию, и пакет необходимых для вызова этого объекта аргументов необходимых типов, а затем автоматически вызывает данный объект с переданными аргументами.
- ☐ (25 б.) На семинаре мы использовали паттерн CRTP как средство расширения функциональности отдельного класса независимо от других. Факт того, что мы передаем шаблону базового класса тип производного класса в качестве параметра, может натолкнуть на мысль, что базовый класс будет иметь явно указанную пользователем информацию о том, с каким потомком он работает. Это в свою очередь может позволить определенным образом избавиться от механизма виртуальных функций и использовать статический полиморфизм на основе шаблонов. Ваша задача – продемонстрировать данную возможность. Напишите простейшую иерархию из двух классов с виртуальной функцией типа print. Затем избавьтесь от указанной виртуальной функции при помощи паттерна CRTP. Дополнительно можете ознакомиться с этой статьей.
- ☐ (50 б.) Существует более продвинутый производный от CRTP паттерн – миксин (MixIn). Он позволяет создать класс, реализующий некоторую функциональность, которую можно добавить в другой класс. Рассмотрим следующую ситуацию. Предположим, Вы хотите написать алгоритм, рассчитывающий величину необходимого усердия для успешной сдачи предметов. В таком алгоритме будут общие компоненты, например, учет халявности преподавателя. Однако для разных предметов будут и разные условия, например, в одном случае необходимо сдавать еженедельные задания, в другом – можно сдать тетрадь соседа. Для начала реализуйте несколько классов, каждый из которых по-разному определяет вычисление некоторой компоненты алгоритма. Создайте общий для них базовый класс, который будет задавать структуру алгоритма и вычислять общие компоненты. В данной ситуации Вам потребуется реализовать паттерн шаблонный метод и идиому NVI, которые мы рассматривали ранее. Обратите внимание, здесь речь идет только об алгоритме с корректируемыми компонентами на виртуальных функциях, никаких дополнительных посредников не требуется. Далее переработайте реализованную Вами иерархию при помощи паттерна CRTP – в результате получится «перевернутый» миксин. В качестве подкрепления можете использовать эту статью. Найдите пример про космические корабли, а также объяснение, почему миксин получится перевернутым. Присылайте оба решения, т.е. классический шаблонный метод на виртуальных функциях и миксин на основе паттерна CRTP. В комментариях обязательно прокомментируйте отличия и особенности данных подходов.