Контрольные вопросы:

- ☐ (5 б.) В чем заключается концепция встраивания вызовов функций?
- ☐ (5 б.) Какие аргументы функции могут иметь значения по умолчанию?
- ☐ (5 б.) На основании чего разрешается выбор перегруженной функции?
- ☐ (5 б.) Как обеспечить состояние в функциях и лямбда-выражениях?
- ☐ (5 б.) По каким причинам макросы считаются опасным инструментом?

Упражнения:

- ☐ (25 б.) На семинаре я упоминал проблему неоднозначности выбора при перегрузке функций. Воспроизведите эту проблему при выборе между функциями `void f(int i) {}` и `void f(double d) {}`. Для этого добавьте что-то в сигнатуру данных функций таким образом, чтобы попытка вызова f привела к неоднозначности выбора перегруженной функции. Предложите минимум 3 принципиально разных варианта решения.
- ☐ (25 б.) Реализуйте алгоритм сортировки слиянием массива чисел. Потренируйтесь использовать рекурсию, передавать данные по ссылке и оптимизировать алгоритм. Псевдокод алгоритма можете посмотреть здесь.
- ☐ (25 б.) Реализуйте алгоритм сортировки массива чисел по возрастанию и по убыванию, используя алгоритм `std::sort` из стандартной библиотеки и пользовательское лямбда-выражение для задания типа сортировки.
- ☐ (25 б.) Реализуйте функцию типа `calculate`, которая принимает два числа с плавающей точкой и лямбда-выражение, и возвращает результат, вычисленный лямбда-выражением на основе переданных аргументов.

  Например: `double result = calculate(1.23, 4.56, [](double x, double y) { return x + y; });`

  В результате выполнения данного фрагмента кода в `result` должно получиться значение 5.79. Для работы с лямбда-выражением используйте обертку на основе `std::function`. Создайте массив на основе контейнера `std::vector`, в котором будут храниться различные лямбда-выражения, и организуйте цикл, в котором к заданной паре чисел посредством функции `calculate` применяется каждое лямбда-выражение из массива.