Контрольные вопросы:

- ☐ (5 б.) Из каких сегментов состоит структура памяти процесса?
- ☐ (5 б.) Каким образом связаны встроенные указатели и массивы?
- ☐ (5 б.) Почему низкоуровненвая работа с памятью небезопасна?
- ☐ (5 б.) Что такое ссылка и чем она отличается от указателя?
- ☐ (5 б.) Какими способами можно передать данные в функцию?

Упражнения:

- ☐ (25 б.) (автор – Никита Попов) Правительство планеты Коулун в рамках программы импортозамещения в IT-сфере разместило новый заказ: Вам необходимо разработать программу, которая сможет конвертировать исходные тексты коулунских писателей в требуемый коулунскими издательствами формат. Каждый исходный текст хранится в строке std::string и состоит из слов и знаков препинания, которые могут быть разделены произвольным количеством пробельных символов. Издательства требуют текст в следующем формате: все слова и знаки препинания должны быть записаны по очереди в отдельные ячейки контейнера std::vector. Составные слова считаются одним словом. Знаки препинания, которые принято писать слитно со словом, например, запятые или точки, также следует помещать в отдельные ячейки контейнера std::vector.
- ☐ (25 б.) Реализуйте алгоритм сортировки вставками массива чисел. Потренируйтесь использовать контейнер `std::vector` без итераторов и функцию обмена `std::swap`. Псевдокод алгоритма можете посмотреть здесь.
- ☐ (25 б.) Реализуйте алгоритм бинарного поиска в отсортированном массиве чисел. Достаточно установить факт наличия или отсутствия искомого числа в массиве. Потренируйтесь использовать контейнер `std::vector` без итераторов и оптимизируйте цикл поиска. Псевдокод алгоритма можете посмотреть здесь.
- ☐ (25 б.) Реализуйте алгоритм поиска наибольшей общей подпоследовательности (longest common subsequence) двух последовательностей объектов. Чтобы получить наибольшую общую подпоследовательность, необходимо удалить некоторые объекты из обеих последовательностей так, чтобы они стали одинаковыми и имели бы при этом максимальную длину. Используйте стандартный алгоритм на основе динамического программирования без дополнительной оптимизации расхода памяти. Описание алгоритма можете посмотреть здесь.