**Конспект**

**Лекция 1**

**Язык программирования —** это набор инструкций и синтаксиса, используемый для создания программных продуктов.Некоторые из основных языков программирования включают:

**Синтаксис:** особые правила и структура, используемые для написания кода на языке программирования.

**Типы данных:** тип результатов, которые могут храниться в программе, такие как числа, строки и логические значения.

**Переменные:** именованные ячейки памяти, в которых могут храниться значения.

**Операторы:** символы, используемые для выполнения операций над значениями, такие как предложение, вычитание и сравнение.

**Управляющие структуры:** инструкции, использование для управления потоком программ, такие как инструкции if-else, циклы и функции вызова.

**Библиотек и платформеры:** коллекция предварительно написанного кода, который можно использовать для выполнения обычных задач и ускорения разработки.

**Парадигмы :** стиль или философия программирования, используемые в языке, например процедурный, объектно-ориентированный или функциональный.

Примеры популярных языков программирования включают Python, Java, C++, JavaScript и Ruby. Каждый язык имеет свои сильные и слабые стороны и подходит для разных типов проектов.

**Язык программирования —** это формальный язык, который определяет набор инструкций для компьютера для выполнения определенных задач. Он используется для написания программ и приложений, а также для контроля и управления компьютерными системами. Существует множество различных языков программирования, каждый из которых имеет свой собственный синтаксис, структуру и набор команд. Некоторые из наиболее часто используемых языков программирования включают Java, Python, C++, JavaScript и C#. Выбор языка программирования зависит от конкретных требований проекта, включая используемую платформу, целевую аудиторию и желаемый результат. Языки программирования продолжают развиваться и меняться с течением времени: разрабатываются новые языки и обновляются старые для удовлетворения меняющихся потребностей.

**Компьютер —** это устройство, которое может принимать команды человека, обрабатывать их и реагировать на них, или компьютер — это вычислительное устройство, которое используется для обработки данных под управлением компьютерной программы. **Программа —** это последовательность инструкций вместе с данными.

**Лекция 2**

**Массив как структура данных**

**Массив —** это структура данных, хранящая набор значений (элементов массива), идентифицируемых по индексу или набору индексов.

**Объекты** могут быть любые нет индексации всегда может что-то ещё поместиться

**Формула** поиска n-ого элемента массива: aₙ = start + (n-1) \* cellSize

1. Начало массива aₙ — искомый адрес в памяти n-ного элемента, start — адрес начала массива n-1 — количество элементов, на которое нам необходимо отступить от начала массива cellSize — размер одного элемента   (сколько ячеек памяти он занимает)

**Параметры массива:**

1. Начало массива(адрес первой ячейки с элементами массива.)

2. Размер каждого элемента массива (сколько ячеек памяти занимает каждый элемент массива)

3. Количество элементов массива

**Формула скалярного произведения :** (a, b) \* (c, d) = a \* c + b \* d

**Как компьютер считывает наш код**

Языки высокого уровня — есть огромное количество посредников; максимально похожи на человеческую речь.

Языки низкого уровня — небольшое количество посредников; команды ближе к командам, которые будет выполнять компьютер

Трансляторы (программы-переводчики) — переводят написанный человеком файл и преобразуют в код для машины.

**Трансляторы бывают двух типов:**

**1. Интерпретаторы** Python, выполняет программу построчно

**2. Компиляторы** Java, cначала переводит всю программу, потом выполняет

Типы ошибок:

**Синтаксические:** Например, неправильно написанные команды, пропущенные знаки или несоблюдение отступов и тд.

**Ошибки выполнения**:Например, деление на 0

**Логические ошибки:**Когда программа работает, но выводит не то, что от неё требуется. Или работает с одним количеством чисел, а с другим уже не работает

**Лекция 3**

**Как научить программу (алгоритм) исключать наибольшее число?**

Для этого нужно вводить в алгоритм индекс максимального элемента. Как научить программу (алгоритм) исключать наибольшее число? Ранее мы заостряли внимание на массу (величину) самой гири, но для решения задачи нужно сконцентрироваться на индексе. Алгоритм надо выстроить так, чтобы на определенном этапе он сравнивал индекс числа с индексом числа, которое надо пропустить. Тем самым можно исключить наибольшее по величине значение и выявить второе по величине.

**Именование переменных**

Самое сложное в программировании — именование переменных. Значения, которые хранятся в переменных должны быть отражены в названии переменных. Читатель должен понимать по названию, что хранится в переменной.

Наименование переменных

**Cтили наименования переменных:** snake case: max\_number\_index CamelCase: MaxNumberIndex

ВАЖНО

*Принцип резиновой уточки — если   что-то не получается, надо взять в руку резиновую утку и просто объяснить ей шаг за шагом действие программы.*

**Структуры в программировании**

**Программу любой сложности можно представить в виде комбинации из трёх структур:**

1. Следование - последовательное выполнение инструкций.
2. Ветвление - условия, которые позволяют перенаправить выполнение нашей программы по одной из имеющихся веток.
3. Цикл - конструкция, которая позволяет задавать многократное повторение операторов

**Сложности**

Сложности бывают разными. Например, мы завязываем шнурки

1. Необходимая сложность. Минимальная сложность, без которой решить задачу нельзя. Нужно уметь завязать узелок и бантик.
2. Необязательная сложность.Помимо решения основной задачи, нужно сделать что-то ещё Если вы попробуете завязать шнурки в варежках.
3. Случайная сложность. Связана с выбором решения. Если вы попробуете завязать шнурки, которые тянутся.