ПЕРЕМЕННЫЕ

**Переменные** — одно из важнейших базовых понятий. Переменные устанавливают отношения между именами и значениями. Они связывают имя с адресом в памяти, где хранится значение. Чаще всего переменные объявляются при помощи оператора присваивания ( variable = 1).

**АБСТРАКТНЫЕ ТИПЫ ДАННЫХ**

**Абстрактный тип данных (АТД)** — подробное описание группы операций, применимых к конкретному типу данных.

| **Название** | **Описание** |
| --- | --- |
| **Примитивные типы данных** | Типы данных со встроенной поддержкой в языке программирования, например целые числа, числа с плавающей точкой и универсальные операции с ними (сложение, вычитание, умножение, деление). |
| **Стек** | Представьте стопку кубиков. Вы всегда можете положить сверху еще один кубик, либо забрать верхний. Кубик, который лежит в основании (самый нижний), всегда будет удален последним. Стек *(stack)* как раз и представляет такую стопку и позволяет взаимодействовать только с верхним элементом, причем элемент на вершине стека — это **всегда** элемент, который был добавлен последним. Любая реализация стека должна поддерживать как минимум **две** базовые операции:   * 1. push(element) — добавить элемент на вершину стека.   2. pop() — получить и удалить верхний элемент.   Такая обработка данных называется ***LIFO***(*Last-In-First-Out,* «последним пришел — первым вышел»). |
| **Очередь** | Очередь *(queue)* — противоположность стеку. Она также позволяет хранить и извлекать данные, но элементы всегда берутся из начала очереди. Основные операции с очередями:   * 1. enqueue(element) — добавить элемент в конец очереди.   2. dequeue() — удалить элемент из начала очереди.   Очередь работает по принципу ***FIFO***(*First-In-First-Out,* «первым пришел — первым вышел»). |
| **Очередь с приоритетом** | Аналогично очереди, с одной лишь разницей — помещаемым элементам присваивается приоритет:   * 1. push(element, priority)— добавить элемент в очередь согласно уровню приоритетности.   2. dequeue() — вернуть элемент из начала очереди и удалить его. |

**СТРУКТУРЫ**

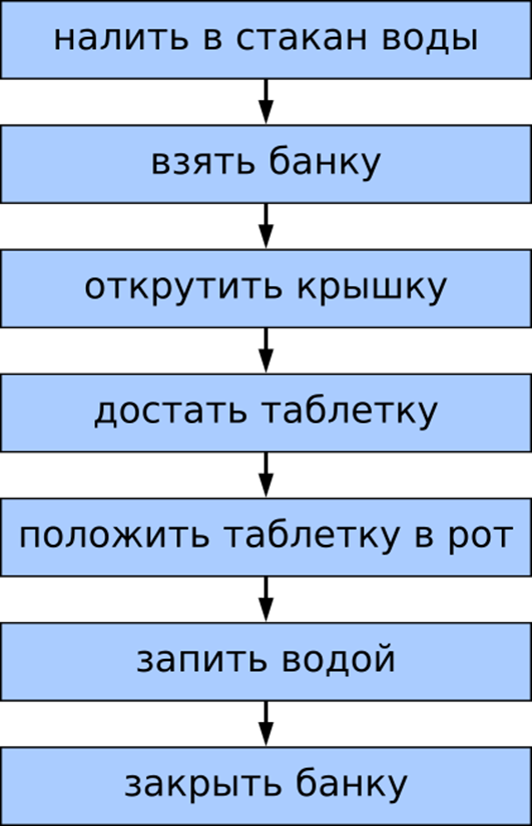
| **Структура** | **Описание** |
| --- | --- |
| **Массив** | Это самый простой способ хранения набора элементов в памяти компьютера — происходит выделение в памяти и последовательная запись ваших элементов (т.е это просто объект, который позволяет хранить не одно значение, а несколько). Если представить, что переменная — это одноэтажный отдельно стоящий дом, то массив — это многоэтажный дом.  **Важно,** что каждый этаж должен быть пронумерован и нумерация начинается с нуля.  **массив[индекс] = значение** |
| **Объект** | Это некая сущность в виртуальном пространстве, которая обладает определенным состоянием и поведением, имеет заданные значения свойств и операций над ними (методов). |

**ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ**

**ЛИНЕЙНЫЕ КОНСТРУКЦИИ**

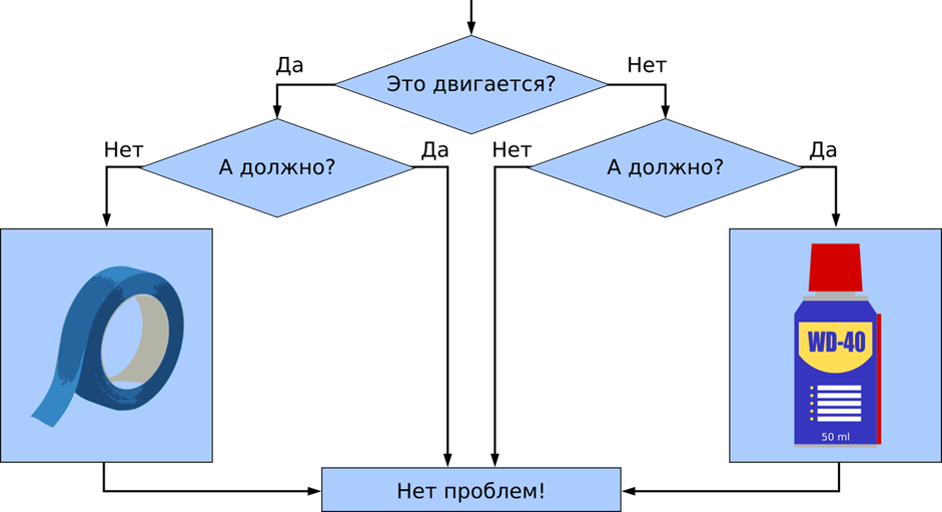
**Алгоритм** — последовательность шагов, схема действий, приводящих к желаемому результату.

**Линейные конструкции или следование** — конструкции, которые отображают последовательный порядок действий. Алгоритмы, использующие только следование, называются *линейными алгоритмами*.



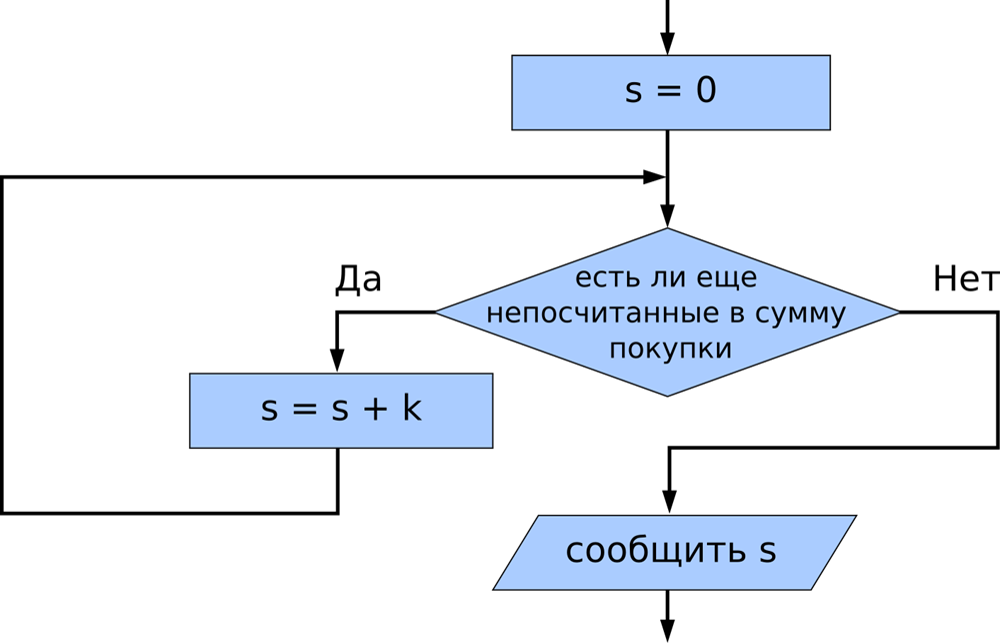
**ВЕТВЛЕНИЕ**

**Ветвление** — конструкции, в которых в зависимости от условия происходит выбор последовательности действий. Алгоритмы, основанные на ветвлении, называются *разветвляющимися (алгоритмами с условием).*



**ЦИКЛ**

**Цикл или повторения** — конструкции, в которых происходит многократное повторение последовательности действий. Алгоритмы, основанные на повторении, называются *циклическими.* Последовательность многократно повторяющихся действий называется *телом цикла.*



**ПОДПРОГРАММА**

**Подпрограмма или функция** — именованная часть программы (именованная подпрограмма), которая может использоваться (вызываться) в других частях программы столько раз, сколько это необходимо. Функции всегда возвращают какое-либо значение.