

# Алгоритмизация и основы объектно-ориентированного программирования

**тема урока:  
«Алгоритм и виды»**

# Понятие алгоритма

**Алгоритм** - это точная конечная система правил, определяющая содержание и порядок действий исполнителя над некоторыми объектами (исходными и промежуточными данными) для получения (после конечного числа шагов) искомого результата.

# Программа и алгоритм

**Программа** - это алгоритм записанный на языке исполнителя.

Алгоритм и программа не отличаются по содержанию, но могут отличаться по форме. Для алгоритма строго не определяется форма его представления.

Один и тот же алгоритм можно изобразить графически, словесно или какими-нибудь специальными знаками, понятными только его автору.

Программа (алгоритм для исполнителя) должна быть записана *только на языке исполнителя.*

---

# Виды алгоритмов

1. Последовательные(линейные)
2. Ветвящиеся
3. Циклические
4. Рекурсивные

---

Линейные, ветвящиеся и циклические алгоритмы являются **базовыми** структурами. Для них характерен *один вход и один выход*.

# Виды алгоритмов.

## Последовательные(линейные)

Алгоритм  $P$  реализован через последовательную алгоритмическую структуру, если каждый шаг алгоритма  $P$  выполняется один раз, причем после каждого  $i$ -го шага выполняется  $(i+1)$ -й шаг, если  $i$ -й шаг не конец алгоритма.

# Виды алгоритмов.

## Ветвящиеся алгоритмы

**Алгоритм Р реализован через ветвящуюся алгоритмическую структуру, если от входных данных зависит, какие шаги алгоритма будут выполнены (последовательность выполнения шагов алгоритма зависит от входных данных).**

---

**При каждом конкретном наборе входных данных ветвящаяся алгоритмическая структура сводится к последовательной алгоритмической конструкции.**

# Виды алгоритмов.

## Циклические алгоритмы

Алгоритм Р реализован с использованием циклической алгоритмической структуры, если некая, подряд идущая группа шагов алгоритма может выполняться несколько раз в зависимости от входных данных.

---

Любая циклическая алгоритмическая конструкция содержит в себе элементы ветвящейся алгоритмической конструкции.

Виды алгоритмов.

## Рекурсивные алгоритмы

Алгоритм  $R$  называется **рекурсивным**, если на каком-либо шаге он прямо или косвенно обращается сам к себе.



# Ошибки в алгоритмах

1. Синтаксические
2. Отказы
3. Логические

# Ошибки в алгоритмах.

## Синтаксические ошибки

**Синтаксические** - ошибки, возникающие в следствии неправильной записи команд исполнителя алгоритма.

---

О синтаксической ошибке исполнитель сообщает до начала выполнения алгоритма.

# Ошибки в алгоритмах.

## Отказы

**Отказы** - ситуация, когда исполнитель прекращает выполнение алгоритма.

Причиной отказа служит команда алгоритма, не входящая в систему команд исполнителя.

---

Отказы возникают в ходе выполнения алгоритма исполнителем.

# Ошибки в алгоритмах. Логические ошибки

Данный вид ошибок не обнаруживается исполнителем ни до, ни во время выполнения алгоритма.

Причина - логическая ошибка, допущенная автором при составлении алгоритма или при постановке задачи, или при выборе метода ее решения.

---

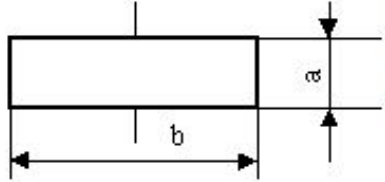
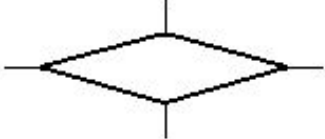
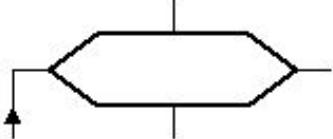
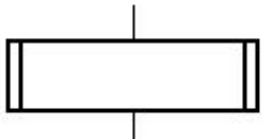

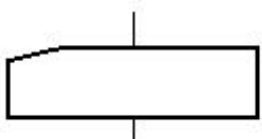
Логические ошибки выявляются в процессе тестирования алгоритма.

Задание алгоритмов с помощью блок-схем оказалось очень удобным средством изображения алгоритмов и получило широкое распространение.

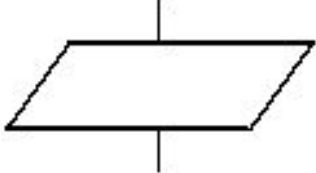

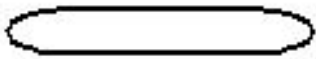

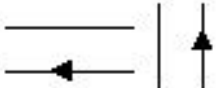
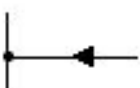

**Блок-схема** алгоритма – графическое изображение алгоритма в виде связанных между собой с помощью стрелок (линий перехода) и **блоков** – графических символов, каждый из которых соответствует одному шагу алгоритма. Внутри блока дается описание соответствующего действия. Блоки и элементы связей называют *элементами* блок-схем. При соединении блоков следует использовать только *вертикальные* и *горизонтальные* линии **ПОТОКОВ**.

Конфигурация и размеры блоков, а также порядок графического оформления блок-схем регламентированы ГОСТ 19002-80 и ГОСТ 19003-80 "Схемы алгоритмов и программ"

# Блок-схема и ее элементы

<i>Название</i>	<i>Элемент</i>	<i>Комментарий</i>
Процесс		Вычислительное действие или последовательность вычислительных действий
Решение		Проверка условия
Модификация		Заголовок цикла
Предопределенный процесс		Обращение к процедуре
Документ		Вывод данных, печать данных
Перфокарта		Ввод данных

# Блок-схема и ее элементы

Ввод/Вывод		Ввод/Вывод данных
Соединитель		Разрыв линии потока
Начало, Конец		Начало, конец, пуск, останов, вход и выход во вспомогательных алгоритмах
Комментарий		Используется для размещения надписей
Горизонтальные и вертикальные потоки		Линии связей между блоками, направление потоков
Слияние		Слияние линий потоков
Межстраничный соединитель		Нет

В рамках структурного программирования задачи, имеющие алгоритмическое решение, могут быть описаны с использованием следующих алгоритмических структур:

- ▶ **Следование.** Предполагает последовательное выполнение команд сверху вниз. Если алгоритм состоит только из структур следования, то он является линейным.
- ▶ **Ветвление.** Выполнение программы идет по одной из двух, нескольких или множества ветвей. Выбор ветви зависит от условия на входе ветвления и поступивших сюда данных.
- ▶ **Цикл.** Предполагает возможность многократного повторения определенных действий. Количество повторений зависит от условия цикла.
- ▶ **Функция (подпрограмма).** Команды, отделенные от основной программы, выполняются лишь в случае их вызова из основной программы (из любого ее места). Одна и та же функция может вызываться из основной программы сколь угодно раз.



# Описание алгоритмических структур на языке блок-схем

## Ветвление if

Это самый простой тип ветвления. Если результат вычисления выражения-условия возвращает true (правда), то выполнение алгоритма идет по ветке «Да», в которую включены дополнительные выражения-действия.

Если условие возвращает false (ложь), то выполнение алгоритма идет по ветке «нет», т.е. продолжает выполняться основная ветка программы.



## Ветвление if-else

Если выражение-условие возвращает true (правда), то выполнение алгоритма идет по ветке «Да», если условие не выполняется (false), то выполнение идет по ветке «Нет». При любом результате выражения-условия нельзя вернуться в основную ветку программы, минуя дополнительные действия.



## Ветвление if-elif-else

Количество условий может быть различно. Если выполняется первое, то после выполнения действий, программа переходит к основной ветке, не проверяя дальнейшие условия. Если первое условие возвращает ложь, то проверяется второе условие. Если второе условие возвращает правду, то выполняются действия, включенные в вторую ветку конструкции. Последнее условие проверяется лишь в том случае, если ни одно до него не дало в результате true. Данную алгоритмическую конструкцию (if - elif - else) не следует путать с алгоритмической конструкцией «Выбор».



## Цикл while

Пока условие выполняется (результат логического выражения дает true), будут выполняться действия тела цикла. После очередного выполнения вложенных действий условие снова проверяется. Для того чтобы выполнение алгоритма не зациклилось, в теле цикла (помимо прочих действий) должно быть выражение, в результате выполнения которого будет изменяться переменная, используемая в условии. Тело цикла может ни разу не выполниться, если условие с самого начала давало false.

Цикл "Пока"  
(цикл с предусловием)



## Цикл do

В этом цикле первый раз условие проверяется лишь после выполнения действий тела цикла. Если условие возвращает true, то выражения-действия повторяются снова. Каким бы ни было условие, тело данного цикла хотя бы раз, но выполнится.

### Цикл с постусловием



## Цикл for

Данный цикл также называют циклом «Для» (for). В его заголовке указывается три параметра: начальное значение переменной (от), конечно значение (до) и ее изменение с помощью арифметической операции на каждом «обороте» цикла (шаг).

### Арифметический цикл



