

# **ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

## **9 КЛАСС**

Учебник для 9 класса школ общего среднего образования

Рекомендовано Министерством народного  
образования Республики Узбекистан.

ИЗДАТЕЛЬСКИЙ ДОМ «TASVIR»

**ТАШКЕНТ – 2020**

ИО'К 004(075.3)

Файзиева М. Р.

КВК 32.81я72

«Информатика и информационные технологии»: учебник

Ф 17

для 9 класса школ общего среднего образования/ Файзиева М. Р.,

Сайфуров Д. М., Хайтуллаева Н. С. – Ташкент: Тасвир, 2020. – 112 с.

Под общей редакцией профессора Закировой Ф.М.

**Рецензенты:**

**Тешабаев Т.З.** – ректор Ташкентского университета информационных технологий им. Мухаммада аль-Хорезми, доктор экономических наук;

**Муминов Б.Б.** – Ташкентский университет информационных технологий им. Мухаммада аль-Хорезми, заведующий кафедрой «Основы информатики», доктор технических наук;

**Турсунова Ф.Р.** – старший преподаватель кафедры «Методика точных и естественных наук» РЦППКРНО г. Ташкента;

**Маматкулов У.Б.** – учитель информатики и информационных технологий школы №9 Касанского района Кашкадаръинской области;

**Тиловова М.М.** – учитель информатики и информационных технологий школы № 7 Сергелийского района г. Ташкента;

**Ипатова Л.В.** – учитель информатики и информационных технологий школы № 257 Юнусабадского района г. Ташкента.



Издано на средства Республиканского  
целевого книжного фонда.

ISBN 978-9943-5800-8-4

© Министерство народного  
образования  
Республики Узбекистан, 2020.  
© Издательский дом «TASVIR», 2020.  
© ООО «KolorPak», 2020.

## **ГЛАВА I. ПРИНЦИП ЛОГИЧЕСКОЙ РАБОТЫ КОМПЬЮТЕРА**

|  |    |
|--|----|
| Основы логики .....                                      | 4  |
| Логические операции и выражения .....                    | 6  |
| Составление таблиц истинности логических выражений ..... | 10 |
| Логические схемы.....                                    | 13 |
| Практическое занятие.....                                | 16 |

## **ГЛАВА II. ПРОЕКТИРОВАНИЕ И МОДЕЛИРОВАНИЕ ЗАДАЧ НА КОМПЬЮТЕРЕ**

|                                      |    |
|--------------------------------------|----|
| Этапы решения задач на компьютере... | 17 |
| Модель и ее виды.....                | 20 |
| Практическое занятие.....            | 26 |

## **ГЛАВА III. ОСНОВЫ АЛГОРИТМИЗАЦИИ**

|  |    |
|--|----|
| Понятие алгоритма и его свойства .....           | 27 |
| Виды алгоритмов и способы их представления ..... | 30 |
| Практическое занятие.....                        | 34 |
| Контрольная работа.....                          | 35 |
| Линейные алгоритмы .....                         | 35 |
| Разветвляющиеся алгоритмы.....                   | 37 |
| Практическое занятие.....                        | 39 |
| Повторяющиеся алгоритмы.....                     | 40 |
| Практическое занятие.....                        | 43 |
| Смешанные (комбинированные) алгоритмы .....      | 44 |

## **ГЛАВА IV. ОСНОВЫ ПРОГРАММИРОВАНИЯ**

|  |    |
|--|----|
| О программе и программировании .....             | 47 |
| Языки программирования .....                     | 48 |
| Установка среды программирования Python.....     | 51 |
| Переменные в Python .....                        | 54 |
| Обработка ошибок с помощью Python                | 57 |
| Типы данных в Python .....                       | 59 |
| Практическое занятие.....                        | 61 |
| Контрольная работа.....                          | 62 |
| Выполнение арифметических операций в Python..... | 62 |

|  |     |
|--|-----|
| Практическое занятие.....  | 64  |
| Работа со строками в Python.....                                       | 65  |
| Практическое занятие.....  | 67  |
| Операторы и выражения в Python .....                                   | 68  |
| Практическое занятие.....  | 69  |
| Программирование простых задач на Python .....                         | 70  |
| Практическое занятие.....  | 72  |
| Программирование логических задач в Python .....                       | 73  |
| Практическое занятие.....  | 75  |
| Программирование разветвляющихся алгоритмов. Оператор if... else ..... | 76  |
| Практическое занятие.....  | 78  |
| Программирование разветвляющихся алгоритмов. Оператор elif .....       | 79  |
| Практическое занятие.....  | 81  |
| Программирование повторяющихся алгоритмов. Оператор for .....          | 82  |
| Практическое занятие.....  | 85  |
| Контрольная работа.....  | 86  |
| Программирование повторяющихся алгоритмов. Оператор while.....         | 86  |
| Практическое занятие.....  | 88  |
| Управление циклами: операторы continue, break.....                     | 89  |
| Подпрограммы: функции и процедуры ...                                  | 91  |
| Практическое занятие.....  | 95  |
| Функции и переменные .....   | 96  |
| Практическое занятие.....  | 98  |
| Библиотека языка программирования Python .....                         | 100 |
| Практическое занятие.....  | 103 |
| Работа с графическим интерфейсом пользователя в Python .....           | 104 |
| Практическое занятие.....  | 107 |
| Контрольная работа.....  | 108 |
| Список использованной литературы и веб-сайтов .....                    | 110 |

# УРОК 1. ОСНОВЫ ЛОГИКИ

ЗНАЕТЕ ЛИ ВЫ?

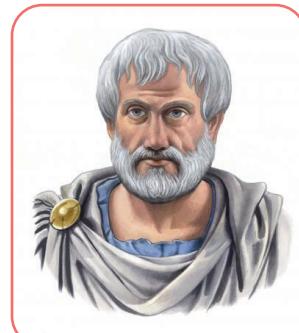
- 1. Что находится между небом и землей? (слово «и»)
- 2. Какого камня не бывает на дне реки? (сухого)
- 3. Какой рукой лучше размешивать чай? (рукой, в которой ложка)

Как вы думаете, к какой категории вопросов относятся эти загадки? Конечно, это были логические вопросы, потому что правильно на них вы смогли ответить только благодаря своей способности мыслить логически. Так что же такое логика, когда она была основана?

Логика имеет свою историю становления и развития. Впервые проблемы логики рассматривались в некоторой степени в учениях Parmenida, Zenona Элейского и Гераклита. Самые ранние учения о логических рассуждениях, формах и методах мышления возникли в странах древнего Востока, в частности, в Индии и Китае, но в древние времена логика была частью философии, которая не сформировалась как самостоятельная наука. В основу современной логики легли учения древнегреческих мыслителей.

ВНИМАНИЕ

- ! Становление логики как отдельной науки приходится на IV век до нашей эры и связано с именем великого греческого ученого Аристотеля. Он первым обозначил круг задач, изучаемых логикой, назвав логику наукой, «определяющей неизвестное с помощью известного», отличающей «истинное от ложного». Ученый первым отделил формы логического мышления от его содержимого, попытался объединить логику и математику, заложил основы теории доказательств.



Также в произведениях Аристотеля проглядываются попытки изображения основ формальной логики и логических процессов с помощью разных математических символов. Сформировать науку логики смог ученый из Центральной Азии Абу Наср аль-Фароби на основе взглядов Аристотеля на общую формальную логику.

В повседневной жизни термин «логика» используется для описания идеи в предложениях, таких как «логика мысли», «логика речи», «логика действия», «логика вещей», «логика событий».

Логика — это изучение форм и проявлений мысли, включая набор правил, регулирующих отношения между идеями.

В настоящее время существуют такие виды логики, как формальная, диалектическая и математическая.

## Запомните!

Термин «логика» соответствует древнегреческому λογικός — науке о мышлении, означает «слово», «мысль», «размышление», «речь», «разум» и неразрывно связан с процессом познания. Объект изучения логики — мышление.

## Запомните!

- **Формальная логика** относится к статической реальности и изучает структуру мышления, независимо от содержания мысли и его развития. В центре внимания лежат правила, связанные с правильным строением обсуждения, и логические действия.
- **Диалектическая логика** относится к динамической реальности и изучает мышление через целостность его содержания и формы.
- **Математическая логика** исследует сознание с помощью математических методов. Она считается одним из основных направлений современной математики.

## ФОРМЫ МЫШЛЕНИЯ

- Понятие** – форма интегрированного выражения основных свойств объектов и событий, важных характеристик.
- Суждение** – подтвержденная или опровергнутая форма мышления о характере объектов и событий, свойствах и связи между ними.
- Умозаключение** – основная логическая форма сознания, выводящая новое утверждение на основании одного или нескольких утверждений.

**Например.** Али старше Вали. А Рано младше Вали. Вывод – Али старше Рано.

Одно из начальных понятий логики – это суждение. Суждение – это повествовательное предложение, об истинности или ложности которого можно рассуждать.

Вопросительные и восклицательные предложения не могут быть суждением. Например, определение «Число, которое без остатка делится на 2, называется четным», не может быть суждением. Но повествовательное предложение «Если целое число делится на 2 без остатка, то это четное число», является суждением. И оно – истинно.

## СУЖДЕНИЯ

- Простые суждения** – неделимые, не связанные никакими условиями или методами и выражающие только одно состояние суждения
- Сложные суждения** – составленные из простых суждений с помощью союзов «и», «или» или частицы «не»

Если суждения правдиво (правильно) отражают свойства, отношения между понятиями, то они являются истиной (правильные), и наоборот, если они противоречат действительности, неправильно излагают, они будут являться ложью (неправильные). Никакое суждение не может быть одновременно и истиной, и ложью. Например, « $7 < 5$ », «А» – гласная буква», «11 – простое число», «Компьютер был изобретен в конце XVI века», «Вирус Covid-19 не опасен для человека». Из этих суждений первое – ложно, второе и третье – истинны, а четвертое и пятое – ложные.

Под значением суждения мы должны понимать его истинность или ложность. Любое суждение может принимать одно «истинное» или «ложное» значение!

**Логические переменные** – это переменные, обозначающие любые суждения и принимающие одно из логических значений «истина» или «ложь». Ниже приведены разные способы обозначений логических значений «истина» или «ложь»:

|        |   |     |       |   |   |
|--------|---|-----|-------|---|---|
| Истина | И | Да  | True  | T | 1 |
| Ложь   | Л | Нет | False | F | 0 |

Обычно суждения принято обозначать большими латинскими буквами (A, B, C, ..., X, Y, Z). Например: A = «Ташкент – красивый город», B = «WWW – всемирная паутина». В этом примере суждения A и B, определенные с помощью логических переменных, являются истиной, и их значение равно 1.



1. Когда появилось понятие логики?
2. Что такое суждение?
3. Какие бывают суждения?
4. Что такое простое суждение и какое значение оно может принимать?
5. Может ли любое повествовательное предложение в прошедшем времени быть суждением? А повествовательное предложение в будущем времени?
6. Что такое логическая переменная? Назовите её значения.



#### 1 задание. Что из нижеследующего является суждением?

Определите истинность или ложность суждений.

- а) Какая длина у этой ленты?
- б) «1234321» – число палиндром.
- в) Делайте утреннюю гимнастику!
- г) Сумма углов треугольника равна  $160^\circ$ .
- д) Назовите устройства ввода информации.
- е) Любое число, делящееся на 5, делится и на 3.

## 2-3 УРОКИ. ЛОГИЧЕСКИЕ ОПЕРАЦИИ И ВЫРАЖЕНИЯ

Логические операции приводят к изменению содержания или размеров суждений. Логическое выражение – это логическое высказывание, записанное с помощью переменных или констант, объединенных логическими функциями, такими как инверсия, конъюнкция, дизъюнкция, импликация и эквиваленция. Обозначаются латинскими буквами A B C. В зависимости от значений участвующих в них переменных, логические выражения могут принимать значения истина (логическая 1) или ложь (логический 0).

Ниже мы ознакомимся с некоторыми операциями, которые можно проделать с простыми суждениями.

Новое суждение, созданное путем связывания двух простых суждений с помощью союза «И», называется **умножением простых суждений**.

## Запомните!

- 1 определение:** Операция создания нового (сложного) суждения, которое является истиной, если оба суждения **A** и **B** одновременно являются истинными, называется **логическим умножением, конъюнкцией** (лат.*conunctio* – связываю).
- Таблица ниже, выражающая логическое умножение, называется **таблицей истинности:**

Имеем следующие простые выражения:

**A** = «Джордж Буль считается основателем математической логики»;

**B** = «Исследования Клода Шеннона дали возможность применения математической логики в вычислительной технике».

**Результат логического умножения.** Джордж Буль считается основателем алгебры логики, и исследования Клода Шеннона дали возможность применения алгебры логики в вычислительной технике.

**Значение окончательного суждения:** Истина.

Новое суждение – «Джордж Буль считается основателем алгебры логики, и исследования Клода Шеннона дали возможность применения алгебры логики в вычислительной технике» – будет истинным, только если оба начальные суждения истинны одновременно. Конъюнкция может применяться не только к двум простым суждениям, но и к нескольким.

В таблицу истинности (в столбцы **A** и **B**) вводятся все возможные варианты для простых суждений. Обычно двоичные значения суждений размещаются в таблице по возрастанию (00, 01, 10, 11). Последний столбец состоит из результата логических действий над соответствующими операндами (элемент, над которым совершается действие).

Посредством одной из следующих форм **A и B, A and B, AΛB, A·B, A∩B, A&B** определяется конъюнкция двух суждений **A** и **B**.

Новое суждение, созданное путем связывания двух простых суждений с помощью союза «ИЛИ», называется **сложением простых суждений**.

## Запомните!

- 2 определение:** Операция создания нового (сложного) суждения, которое является истиной, если по меньшей мере одно суждение является истинным, называется **логическим сложением, дизъюнкцией** (лат.*disjunctio* – разобщаю).
- Таблица истинности логического сложения выглядит следующим образом:

Имеем следующие простые выражения:

**A**=«Идея использования в логике математических символов принадлежит Готфриду Вильгельму Лейбницу»;

**B**=«Лейбниц является основателем бинарной арифметики».

**Результат логического сложения.** Идея использования в логике математических символов принадлежит Готфриду Вильгельму Лейбницу или Лейбница является основателем бинарной арифметики.

**Значение окончательного суждения:** Ложь.

| <b>A</b> | <b>B</b> | <b>A &amp; B</b> |
|----------|----------|------------------|
| 0        | 0        | 0                |
| 0        | 1        | 0                |
| 1        | 0        | 0                |
| 1        | 1        | 1                |

| <b>A</b> | <b>B</b> | <b>A ∨ B</b> |
|----------|----------|--------------|
| 0        | 0        | 0            |
| 0        | 1        | 1            |
| 1        | 0        | 1            |
| 1        | 1        | 1            |

! Посредством одной из следующих форм:  $A \text{ или } B$ ,  $A \text{ or } B$ ,  $A \vee B$ ,  $A + B$ ,  $A \cup B$  определяется дизъюнкция двух суждений А и В.

Создание нового суждения путем добавления к суждению А отрицательной частицы «НЕ» называется **отрицанием простого суждения**.

### Запомните!

- 3 **определение:** Операция, изменяющая значение суждения А на ложь, когда оно истинно, и на истину, если оно ложно, называется **инверсией** (лат.*inversio* – переворачивание, перестановка), **логическим отрицанием**.
- Таблица истинности логического отрицания выглядит следующим образом:

Операция отрицания создает новое суждение, противоречащее начальному. Например, отрицание суждения  $A = \text{«Наш дом расположен в центре города»}$ , выглядит как  $\neg A = \text{«Наш дом не расположен в центре города»}$ .

| A | $\neg A$ |
|---|----------|
| 0 | 1        |
| 1 | 0        |

! Посредством одной из следующих форм,  $\neg A$ ,  $\text{not } A$ ,  $\neg A$ ,  $\bar{A}$  определяется инверсия суждения А.

### Запомните!

- 4 **определение:** Суждение, принимающее значение ложь, только если суждение А истинно и суждение В ложно, во всех остальных случаях принимающее значение истина называется **импликацией** суждений А и В.

Знак « $\Rightarrow$ » называется **знаком импликации**. Логическое выражение  $A \Rightarrow B$  означает: «если А, то будет В» или «Из суждения А следует суждение В».

Таблица истинности импликации выглядит следующим образом:

Импликация (лат.*implicatio*) означает связывание. Например:  
**A** = «Если 72 кратно 9, то это число кратно и 3». Импликация суждения А истинна, потому что оба суждения в составе сложного суждения истинны.

**B** = «Если  $-3 < -1$ , то  $9 < 8$ ». Импликация суждения В ложна. Потому что условие  $-3 < -1$  истинно, а  $9 < 8$  ложно.

| A | B | $A \Rightarrow B$ |
|---|---|-------------------|
| 0 | 0 | 1                 |
| 0 | 1 | 1                 |
| 1 | 0 | 0                 |
| 1 | 1 | 1                 |

! Посредством одной из следующих форм,  $A \Rightarrow B$ ,  $A \rightarrow B$  определяется импликация суждения А.

### Запомните!

- 5 **определение:** Суждение, которое является истинным, только если суждения А и В одновременно истинны или одновременно ложны, называется **эквиваленцией суждений А и В**.

Знак « $\Leftrightarrow$ » называется **знаком эквиваленции**.  $A \Leftrightarrow B$  читается как «Из суждения А следует суждение В, и из суждения В следует суждение А» или «Будет А, только если будет

*В* или «*А эквивалентно В*».

**Таблица истинности** эквиваленции выглядит следующим образом:

Например:

**A** = «Число 972 кратно 9»,

**B** = «Сумма цифр числа 972 кратна 9».

В таком случае эквиваленция *A* и *B* будет выглядеть так: «Число 972 кратно 9, только если сумма цифр числа 972 кратна 9». И эта эквиваленция – истинна.

| <b>A</b> | <b>B</b> | <b>A &lt;=&gt; B</b> |
|----------|----------|----------------------|
| 0        | 0        | 1                    |
| 0        | 1        | 0                    |
| 1        | 0        | 0                    |
| 1        | 1        | 1                    |

Посредством одной из следующих форм **A<=>B**, **A<->B** определяется ! эквиваленция суждения *A*.

Произвольное сложное суждение можно написать в виде логического выражения. Сложные логические выражения состоят из одного или нескольких простых (сложных) логических выражений, связанных между собой логическими операциями. Эти логические выражения состоят из логических переменных, отношений, логических операций и скобок. Например: **(A v̄ B) & (C <=>̄ D)**



### Запомните!

Логические операции в логических выражениях выполняются в следующем порядке: инверсия ( $\neg$ ); конъюнкция ( $\&$ ); дизъюнкция ( $v$ ); импликация ( $\Rightarrow$ ); эквиваленция ( $\Leftrightarrow$ ).



В случае, если выполняются равнозначные или одинаковые операции, они будут выполняться по порядку, слева направо. Если в выражениях участвуют скобки, первым выполняется их содержимое. Во вложенных скобках первым выполняется содержание самых внутренних скобок.

Рассмотрим некоторые примеры:

**1 пример:** если значение *A* принимает значение ложь, установите значение выражения «(*НЕ A*) или *A*».

**Решение:** выражение (*НЕ A*) будет истинным, потому что *A* ложно. В таком случае, мы придем к значению «истина», потому что сложение (операция «ИЛИ») значений «истина» и «ложь» даст нам «истину». Значит, результат – «истина». Ответ: Истина.

**2 пример:** для значений *x*=1,6 и *y*=8,7. *A*=«истина», *B*=«ложь» вычислите значение логического выражения  $(Av̄ B)\&(x>y)$ .

**Решение:** в связи с тем, что *B* ложно,  $\neg B$  будет истиной. А истина, и  $\neg B$  тоже истина, следовательно,  $(Av̄ B)$  тоже истина.  $(1,6>8,7)$  не правда, поэтому это выражение принимает значение «ложь». В таком случае  $(Av̄ B)\&(x>y)$  примет значение «ложь» ( $1\&0=0$ ). Ответ: Ложь.

**3 пример:** Запишите в виде логического выражения следующее суждение: «Если я куплю яблоки или абрикосы, тогда приготовлю фруктовый пирог»

**Решение:** сначала разделим сложное суждение на простые суждения: *A*=«Куплю яблоки», *B*=«Куплю абрикосы», *C*=«Приготовлю фруктовый пирог». В этом



случае сложное выражение «Если я куплю яблоки или абрикосы, тогда приготовлю фруктовый пирог» можно записать в виде следующего логического выражения:  $(A \vee B) \Rightarrow C$

- Какие существуют логические операции, выполняемые над суждениями?
- Как создаются логические выражения?
- Расскажите про операцию логического сложения и её таблицу истинности.
- Расскажите про операцию логического умножения и её таблицу истинности.
- На что надо обращать внимание при создании таблиц истинности логических выражений?

6. Запишите в виде логических выражений следующие суждения: «Ученики выполнили на уроке физики лабораторную работу и показали результаты эксперимента учителю».



- Для значений  $A = \text{истина}$ ,  $B = \text{ложь}$ ,  $C = \text{истина}$  выполните следующие операции:
  - $A \vee B \& C$ ;      c)  $B \vee (C \& A)$ ;
  - $B \vee \neg C$ ;      d)  $\neg (A \& B) \vee (B \Rightarrow C \vee \neg A)$ .
- Если  $A = \text{ложь}$ ,  $B = \text{«Инверсия считается логическим отрицанием»}$ ,  $C=3,14$ ,  $D=7,9$  выполните следующие операции:
  - $(D=C) \& A \& B$ ;      c)  $A \vee (C < D) \& A \vee B$ ;
  - $B <= > (C > D) \& A$ ;      d)  $\neg (A \& B) \Rightarrow ((C+D) > 16)$ .

## 4 УРОК. СОСТАВЛЕНИЕ ТАБЛИЦ ИСТИННОСТИ ЛОГИЧЕСКИХ ВЫРАЖЕНИЙ

В формальном языке, то есть в языке, состоящем из формул, используются специальные символы ( $\&$ ,  $\vee$ ,  $\neg$ ,  $\Rightarrow$ ,  $<= >$ ), называемые логическими союзами. На предыдущих уроках мы ознакомились с этими операциями, их названиями и написанием.

Логические формулы определяются посредством таблиц истинности. Такие таблицы определяются результатом сложных суждений, состоящих из простых и связанных между собой логическими союзами, принимающими значения истина (1) или ложь (0). Используя таблицы истинности логических операций, можно также составить таблицы истинности и для более сложных суждений.

При составлении таблиц истинности нужно следовать правильной последовательности выполнения операций. В последовательности суждений первой выполняется операция отрицания, за ней конъюнкция, дизъюнкция, импликация и в конце эквиваленция.

Если операция отрицания находится перед скобками, то первыми выполняются операции внутри скобок. В случае, если выполняются равнозначные или одинаковые операции, то они будут выполняться по порядку, слева направо. Во вложенных скобках первым выполняется содержание самых внутренних скобок.

## Запомните!

Коллекция суждений и все операции  $\&$ ,  $\vee$ ,  $\neg$ ,  $\Rightarrow$ ,  $\Leftrightarrow$ , выполняемые в ней, вместе

называются **алгеброй суждений**.

Сложные суждения, созданные посредством связывания суждений A, B, C с помощью логических союзов, называются **логическими формулами**.

Рассмотрим последовательность построения таблицы истинности логического выражения:

### 1. Определение количества переменных n в составе логического выражения:

$A \wedge B$ . Мы знаем, что переменные обозначаются большими латинскими буквами.  
 $n = 2$ .

### 2. Определение количества логических операций k:

Участвующие логические операции  $\vee$ ,  $\&$ .  $k = 2$ .

### 3. Определение последовательности выполнения логических операций с учетом скобок и правил выполнения логических операций.

- 1)  $A \& B$ ;
- 2)  $A \vee (A \& B)$ .

### 4. Определение количества столбцов в таблице. Суммируется количество переменных и количество операций: $c = n + k$ :

$$2 + 2 = 4.$$

### 5. Заполнение первой строки таблицы с учетом переменных и последовательности выполнения операций, определенных в 3 пункте:

|   |   |          |                 |
|---|---|----------|-----------------|
| A | B | $A \& B$ | $A \vee A \& B$ |
|---|---|----------|-----------------|

### 6. Количество строк в таблице определяется формулой $r = 2^n$ (строка заголовка в 5 пункте не считается).

$$r = 2^2 = 4.$$

Множество входящих переменных – с 0 до  $2^2 - 1 = 3$ .

Двоичные числа: 00, 01, 10, 11

| A | B | $A \& B$ | $A \vee A \& B$ |
|---|---|----------|-----------------|
|   |   |          |                 |
|   |   |          |                 |
|   |   |          |                 |
|   |   |          |                 |

### 7. Выписываем двоичное множество переменных n-го разряда:

| A | B | $A \& B$ | $A \vee A \& B$ |
|---|---|----------|-----------------|
| 0 | 0 |          |                 |
| 0 | 1 |          |                 |
| 1 | 0 |          |                 |
| 1 | 1 |          |                 |

### 8. Согласно входящим данным выполняем последовательность логических операций в столбцах таблицы и заполняем таблицу. То есть выполняем логические операции по входящим данным.

| A | B | $A \& B$ | $A \vee A \& B$ |
|---|---|----------|-----------------|
| 0 | 0 | 0        | 0               |
| 0 | 1 | 0        | 0               |
| 1 | 0 | 0        | 1               |
| 1 | 1 | 1        | 1               |

Рассмотрим примеры составления таблиц истинности.



**1 задача:** Составьте таблицу истинности суждения  $((A \vee B) \ \& \ (\neg A)) \Rightarrow B$ :

**Решение:**

1. В данном выражении количество переменных равно  $n=2$ , участвуют переменные А и В.
2. Участвующие логические операции:  $\vee$ ,  $\neg$ ,  $\&$ ,  $\Rightarrow$ .  $k = 4$ .
3. Последовательность операций:
- 4)  $(A \vee B) \ \& \ (\neg A)$ ;
- 4)  $((A \vee B) \ \& \ (\neg A)) \Rightarrow B$ .
4. Количество столбцов в таблице  $c = 2 + 4 = 6$ .
5. Первая строка таблицы будет выглядеть так:

| A | B | $A \vee B$ | $\neg A$ | $(A \vee B) \ \& \ (\neg A)$ | $((A \vee B) \ \& \ (\neg A)) \Rightarrow B$ |
|---|---|------------|----------|------------------------------|--|
|---|---|------------|----------|------------------------------|--|

6. Количество строк в таблице:  $r = 2^2 = 4$ .

7. Выпишем двоичное множество переменных  $n$ -го разряда:

| A | B | $A \vee B$ | $\neg A$ | $(A \vee B) \ \& \ (\neg A)$ | $((A \vee B) \ \& \ (\neg A)) \Rightarrow B$ |
|---|---|------------|----------|------------------------------|--|
| 1 | 1 |            |          |                              |  |
| 1 | 0 |            |          |                              |  |
| 0 | 1 |            |          |                              |  |
| 0 | 0 |            |          |                              |  |

8. Заполним таблицу истинности:

| A | B | $A \vee B$ | $\neg A$ | $(A \vee B) \ \& \ (\neg A)$ | $((A \vee B) \ \& \ (\neg A)) \Rightarrow B$ |
|---|---|------------|----------|------------------------------|--|
| 1 | 1 | 1          | 0        | 0                            | 1  |
| 1 | 0 | 1          | 0        | 0                            | 1  |
| 0 | 1 | 1          | 1        | 1                            | 1  |
| 0 | 0 | 0          | 1        | 0                            | 1  |



1. Что такое суждение? Объясните разницу между простыми и сложными суждениями.
2. Перечислите основные логические операции.
3. Расскажите о последовательности выполнения логических операций.
4. Что такое таблица истинности? Из каких элементов она состоит?
5. Расскажите о последовательности составления таблицы истинности.
6. В каких логических операциях должны участвовать по меньшей мере два суждения?



1. Составьте таблицы истинности для следующих логических суждений:
  1.  $B \ \& \ (A \vee B)$ ;
  2.  $A \ \& \ (A \vee B \vee C)$ ;
  3.  $\neg A \ \& \ B \vee \neg C$ ;
  4.  $(A \vee B) \ \& \ \neg A$ ;
  5.  $B \Leftrightarrow (\neg C \vee D) \ \& \ A$ ;
  6.  $\neg (A \vee B \Rightarrow C) \vee (B \Rightarrow C \vee \neg A)$ .
2. Определите истинность логического выражения  $((C \vee B) \Rightarrow B) \ \& \ (A \vee B) \Rightarrow B$ .

## 5-6 УРОКИ. ЛОГИЧЕСКИЕ СХЕМЫ

Алгебра логики считается одним из разделов математики и занимает важное место в проектировании автоматических аппаратов, в производстве аппаратного и программного обеспечения информационно-коммуникационных технологий. Как известно, любая информация может быть представлена в дискретном виде, то есть в виде фиксированного множества отдельных значений.

Если дискретное обрабатывающее устройство после обработки двоичных сигналов выдаёт значение определенной логической операции, то оно называется **логическим элементом**. А устройства, обрабатывающие подобные сигналы, называются **дискретными устройствами**.

Логические элементы являются составной частью компьютера и считаются элементами, предназначенными для выполнения определенных логических операций над двоичными переменными.

Всё вычислительное оборудование современных цифровых технологий (компьютер, мобильные устройства), основано на логических элементах. Любая логическая операция на компьютере выполняется с помощью основных логических элементов. Каждый логический элемент обеспечивает выполнение одной или нескольких логических операций.

Далее мы познакомимся с самими простыми и распространенными логическими элементами.

Сами элементы состоят из простых электрических схем. Входящие в схему сигналы называются **аргументом**, а исходящий сигнал является **функцией аргумента**. Наличие сигнала в определенном месте выражается единицей (1), а отсутствие – нулем (0).

Для выполнения логических функций «И (&)» и «ИЛИ(V)» требуется по меньшей мере два входящих сигнала. А в некоторых случаях их количество может быть и более двух.

Базовые логические элементы компьютера, в основном, выполняют три логические операции:

- 1) **конъюнктор** – (логический элемент «И») выполняет логическое умножение;
- 2) **дизъюнктор** – (логический элемент «ИЛИ») выполняет логическое сложение;
- 3) **инвертор** – (логический элемент «НЕ») выполняет отрицание.

Ниже приведена таблица истинности логического элемента «И» для двух входящих А и В элементов. Видно, что только в случае одновременного обеспечения обоих входящих сигналов входящим сигналом «1» на выходе появляется сигнал «1». В остальных же трех случаях сигнал на выходе равен «0».

**В схеме соответствия конъюнктор** реализует операцию «логическое умножение», есть по меньшей мере два (A, B) входящих и один (A и B) исходящий сигнал. В цифровых схемах логический элемент «И» обозначается, как показано на рисунке 1.

В схемах зарубежных производителей элемент «И» имеет другой вид (см. рисунок 2). Он называется элементом AND.

### Запомните!

**Логическая переменная** – величина, принимающая только одно из двух значений: 0 или 1.

**Логическая функция** – функция, аргументы которой принимают только значения 0 и 1.

**Дизъюнктор** – (элемент ИЛИ) – реализует операцию «логическое сложение», собирающая схема тоже имеет по меньшей мере два (A, B) входящих и один (A или B) исходящий сигнал.

Логический элемент «ИЛИ» для двух (A, B) входящих сигналов работает несколько иначе.

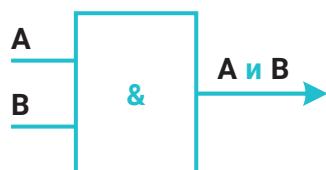


Рисунок 1. Логический элемент «И»

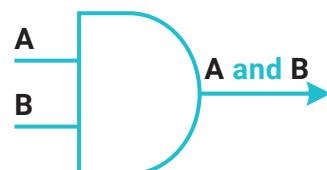


Рисунок 2. Элемент «AND»

| Входящий A | Входящий B | Исходящий (A и B) |
|------------|------------|-------------------|
| 0          | 0          | 0                 |
| 1          | 0          | 0                 |
| 0          | 1          | 0                 |
| 1          | 1          | 1                 |

В цифровых схемах логический элемент «ИЛИ» обозначается, как показано на рисунке 3.

В схемах зарубежных производителей элемент «ИЛИ» имеет вид, показанный на рисунке 4. Он еще называется «OR».

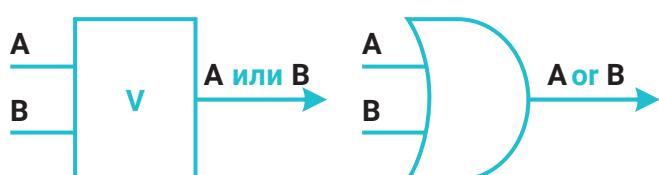


Рисунок 3. Логический элемент «ИЛИ»

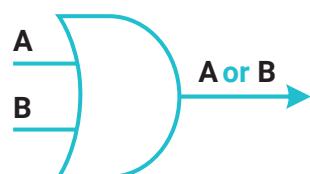


Рисунок 4. Элемент «OR»

| Входящий A | Входящий B | Исходящий (A или B) |
|------------|------------|---------------------|
| 0          | 0          | 0                   |
| 1          | 0          | 1                   |
| 0          | 1          | 1                   |
| 1          | 1          | 1                   |

| Входящий A | Исходящий (не A) |
|------------|------------------|
| 0          | 1                |
| 1          | 0                |

В схеме инвертора существуют только один (A) входящий и один (не A) исходящий сигналы. Инверторная схема также называется «обратной цепью».

В схеме инвертора значение входящего сигнала меняется на противоположный. Например, если входящий сигнал снабжается сигналом

«1», то через исходящий сигнал появляется «0», и наоборот.

В цифровых схемах логический элемент «НЕ» обозначается, как показано на рисунке 5.

В схемах зарубежных производителей элемент «НЕ» имеет вид, показанный на рисунке 6. Он еще называется «NOT».



Рисунок 5. Логический элемент «Не».



Рисунок 6. Элемент «NOT»

При использовании логических элементов создаются сложные цифровые схемы, предназначенные для выполнения арифметических задач и хранения информации. С помощью нескольких логических элементов и различных их комбинаций можно составить схемы, способные выполнять заданные функции.

### Запомните!

**Логическая схема** – электронное устройство, выполняющее любую функцию, описывающую работу компьютера.



**Задача 1:** Нарисуйте схему, подходящую логическому выражению  $A \& B \vee \neg(B \vee A)$ , и вычислите значение исходящего сигнала для входящих сигналов  $A=1$  и  $B=0$ .

**Решение:**

- 1) В выражении есть две переменные: А и В.
- 2) В выражении есть четыре логические операции: конъюнкция, две дизъюнкции и отрицание. Последовательность выполнения операций следующая:

$$\begin{matrix} 3 & 4 & 2 & 1 \\ A \& B \vee \neg(B \vee A) \end{matrix}$$

- 3) Согласно последовательности выполнения логических операций схема будет строиться слева направо (рисунок 7).

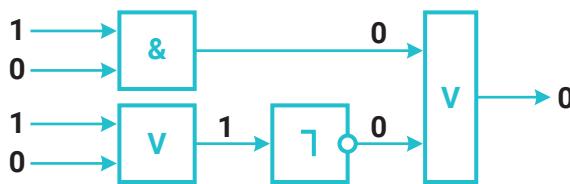


Рисунок 7. Построение логической схемы

- 4) Вычислить значение исходящего сигнала  $1 \& 0 \vee \neg(0 \vee 1) = 0$ .



### Последовательность построения логической схемы:

- 1) Определение количества логических переменных;
- 2) Определение количества основных логических операций и их последовательности;
- 3) Указание соответствующего элемента для каждой логической операции;
- 4) Соединение логических элементов друг с другом согласно последовательности выполнения логических операций.

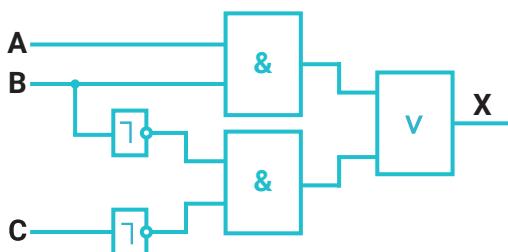


1. Что такое логический элемент?
2. Перечислите основные логические элементы, изобразите их в схемах.
3. Какую задачу выполняет элемент конъюнктор?
4. Какую задачу выполняет элемент дизъюнктор?
5. Какую задачу выполняет элемент инвертор?
6. Для чего строятся логические схемы?
7. Расскажите алгоритм построения логических схем.



- 1.** Постройте схемы, соответствующие следующим логическим выражениям.
- $D \vee \neg B \wedge A \wedge (B \vee \neg A)$ ;
  - $A \wedge B \vee (\neg C)$ ;
  - $A \vee B \wedge (\neg C \vee E)$ ;
  - $A \vee \neg B \wedge C$ ;
  - $\neg (\neg A \wedge C) \vee \neg B$ ;
  - $\neg A \wedge (\neg B \vee C)$ ;

- 2.** Напишите логическое выражение, соответствующее следующей схеме:



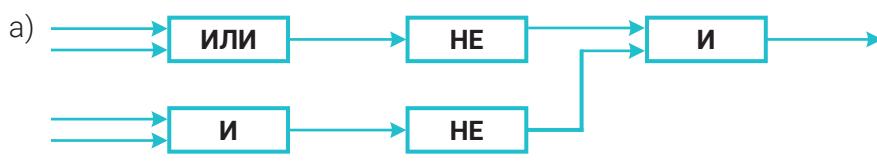
**1. Какие из приведенных предложений являются суждениями?**

- Самым легким металлом считается литий;
- Африка – второй по величине материк после Евразии;
- Первый Президент Республики Узбекистан Ислам Каримов родился 30 января 1938 года в городе Самарканде;
- Летом часто идет снег;
- $5 \cdot 2 < 14,6 + 3$ ;
- Растровая графика – одна из самых интересных тем информатики;
- Уходя в школу, заприте двери.

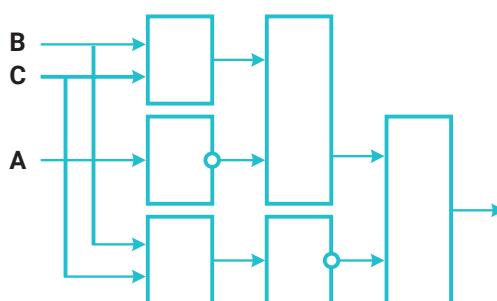
**2. Постройте таблицы истинности и схемы логических выражений:**

- $A \wedge (B = \neg C)$ ;
- $B \wedge \neg (C \vee \neg B = A)$ ;
- $(A \vee C) = \neg (\neg B \wedge A \wedge D)$ ;
- $B = \neg (\neg C \vee D) \wedge A$ ;
- $(A = \neg D) \wedge (C \vee \neg B)$ ;
- $(A \vee C) \Leftrightarrow (\neg B \vee A \vee D)$ ;
- $\neg (A = \neg B \vee C) \wedge (D = \neg B)$ ;
- $\neg (A \wedge B = C) \vee (B = \neg C \vee \neg A)$ .

**3. Какие значения должны быть при входе, чтобы при выходе получить 0 в следующих схемах?**



**4. Найдите логическое выражение, соответствующее данной схеме:**



# 8-9 УРОКИ. ЭТАПЫ РЕШЕНИЯ ЗАДАЧ НА КОМПЬЮТЕРЕ

В процессе практической работы человеку приходится решать множество задач. Одни из них решаются при помощи легких, а другие – сложных вычислений. При решении некоторых задач может потребоваться многоразовое выполнение определенной группы операций. Исходя из этого, могут появиться определенные вопросы.

Сможет ли нам помочь в этом наш безвозмездный и быстрый помощник – компьютер? Если да, то как организовать все это на компьютере? На самом деле, компьютер создан для быстрого решения задач и обработки данных.

**Решение любых задач при помощи компьютера состоит из следующих этапов:**

- 1 этап: постановка задачи;
- 2 этап: составление математической модели задачи;
- 3 этап: алгоритмизация;
- 4 этап: программирование;
- 5 этап: ввод программы в память компьютера;
- 6 этап: получение и анализ результата.

**На 1 этапе** определяется правильность постановки задачи, её цель и содержание. Изучаются все показатели и их свойства. Определяется ожидаемый результат, входящие и исходящие величины. Точность и исчерпывающая ясность задания дает возможность полного решения проблемы.

**На 2 этапе** решение задачи рассматривается с точки зрения достижений науки и математики. То есть посредством взаимных связей всех приведенных величин составляется математическая модель. Выбирается математический метод для точного и ясного решения поставленной задачи. При решении задачи можно воспользоваться различными методами, только выбранный метод обязательно должен привести к точному решению.

**На 3 этапе**, используя модель задачи, составляется алгоритм решения. Определенная последовательность инструкций составляется при помощи одного из методов описания алгоритмов. Например, инструкции можно описать с помощью блок-схемы или словами. После того как алгоритм будет готов, можно переходить к следующему этапу.

**На 4 этапе**, когда последовательность инструкций в алгоритме готова, она переводится на язык, понятный компьютеру.

**На 5 этапе** при помощи языков программирования готовая программа вводится в память компьютера.

**На 6 этапе** программа запускается, результаты анализируются. При помощи тестов определяется корректность программы. При определении ошибок в программе они устраняются. Ошибки могут быть связаны с нарушениями правил программирования на конкретном языке программирования. Процесс сравнивается с исходными данными и останавливается при получении правильного результата.

**1 задача.** На льду лежит хоккейная шайба массой 150 гр. Какое ускорение получит шайба, если хоккеист нанесет по ней удар силой 100 Н?

**Проанализируем задачу:** из курса физики нам известно, что ускорение предмета пропорционально силе, действующей на него, и обратно пропорционально его массе. Это основывается на втором законе Ньютона.

**Дано:**

$$m = 150 \text{ г} = 150:1000 = 0,15 \text{ кг.}$$
$$F = 100 \text{ Н.}$$

**Формула:**

$$a = \frac{F}{m}$$

**Решение:**

$$a = \frac{100}{0,15} \cdot \frac{\text{Н}}{\text{кг}} = 666,67 \frac{\text{м}}{\text{s}^2}$$

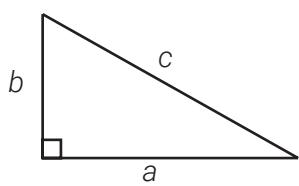
**Найти:**  $a - ?$

**Ответ:**  $666,67 \frac{\text{м}}{\text{s}^2}$ .

**2 задача:** Дильнур синим карандашом нарисовала на белой бумаге прямоугольный треугольник с катетами 12 см и 9 см. Найдите периметр и площадь данного треугольника.

**Проанализируем задачу:** во-первых, определим, что для решения задачи не имеет значения, каким карандашом Дильнур нарисовала треугольник, то есть для нас это не является важной информацией. Во-вторых, прямоугольность треугольника – это уже важная для нас информация. Если мы вспомним теорему Пифагора из курса математики, то решение задачи будет иметь следующий вид:

**Чертеж:**



**Дано:**

$$a = 12 \text{ см.}$$
$$b = 9 \text{ см.}$$

**Найти:**

$$P_{\text{треуг.}} - ?$$
$$S_{\text{треуг.}} - ?$$

**Формулы:**

Гипотенуза:  $c = \sqrt{(a^2 + b^2)}$ .

Периметр:  $P_{\text{треуг.}} = a + b + c$ .

Площадь:  $S_{\text{треуг.}} = \frac{1}{2}ab$ .

**Решение:**

В соответствии с теоремой Пифагора:  $c = \sqrt{(a^2 + b^2)} = \sqrt{((12\text{см})^2 + (9\text{см})^2)} = \sqrt{(225\text{см}^2)} = 15 \text{ см.}$

Значит:  $P_{\text{треуг.}} = 12 \text{ см} + 9 \text{ см} + 15 \text{ см} = 36 \text{ см. } S_{\text{треуг.}} = 1/2(12 \text{ см} * 9 \text{ см}) = 54 \text{ см}^2$ .

**Ответ:** 36 см и 54 см<sup>2</sup>.

**3 задача:** Бунёд подготовил реферат, состоящий из 46 страниц. На каждой странице по 40 строк, каждая из которых содержит в среднем 67 символов. Если у Бунёда есть мобильный интернет-пакет с 51 Мб, то сможет ли он отправить своему учителю подготовленный документ по электронной почте?

Перейдем к анализу задачи.

**Начальные значения задачи:**

- Документ состоит из 46 страниц;
- Число символов в каждой строке равно 67;
- На странице по 40 строк.

**Цель задачи:**

Первым делом надо вычислить размер документа и после выяснить, можно ли его отправить по электронной почте.

**Создание математических соотношений в соответствии с условиями задачи:**

В задаче количество символов на 1 странице равно  $x$ , а количество символов на всех страницах равно  $y$ . Согласно условию задачи, если Бунёд в процессе работы над рефератом набрал по  $x$  символов на каждой странице, то на всех страницах он набрал  $y = x \cdot 46$  символов (46 страниц, содержащих по  $x$  символов). Эти данные нужны для вычисления размера документа. Из курса информатики мы знаем, что один символ равен одному байту. Чтобы Бунёд мог отправить своему учителю подготовленный документ, должно быть уместно неравенство  $y < 51$ . То есть согласно условию задачи мы создали неравенство.

**Последовательность выполняемых действий:**

- 1) Рассчитаем количество знаков на 1 странице:  $x = 67 \cdot 40 = 2680$  знаков;
- 2) Находим общее количество знаков в документе:  $y = 2680 \cdot 46 = 123280$  знаков;
- 3) В связи с тем что объем мобильного интернета в распоряжении Бунёда выражается в Мбайтах, размер документа мы тоже должны перевести в Мб:  
 $123\ 280 : 1024 : 1024$  Мбайт = 0,118 Мбайт;
- 4) Проверяем неравенство  $y < 51$ : получаем  $0,118$  Мб < 51 Мб. Неравенство действительно.

**Анализ результата:**

Неравенство 0,118 Мб < 51 Мб было правильным, поэтому Бунёд сможет отправить преподавателю подготовленный реферат по электронной почте.

**Ответ:** да, может.

Теперь рассмотрим, какие этапы были пройдены в приведенных выше примерах:

1. В каждой задаче есть правильная **постановка задачи**, цель и содержание, начальные и результирующие величины (то, что должно быть найдено).
2. Определяются **формулы**, необходимые для решения задачи, другими словами, **математические соотношения**.
3. Определяется последовательность **действий** (формулы, соотношения) для решения задачи.
4. **Получение и анализ результатов.**

Поскольку приведенные выше этапы «Программирование» и «Ввод программы в память компьютера» требуют дополнительных знаний и навыков, мы изучим их на следующих уроках.

1. Перечислите этапы решения задач на компьютере.
2. Почему необходимо построение алгоритма для каждой задачи?
3. Для чего определяются начальная и конечная величины задачи?
4. Какие ошибки возникают при выполнении расчетов на калькуляторе?
5. Опишите последовательность этапов при решении квадратного уравнения.
6. Для чего анализируется полученный результат?
7. Приведите пример построения уравнения, соответствующего условию задачи.

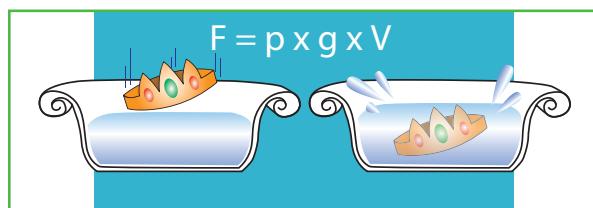




Проанализируйте условия следующих задач и решите их поэтапно:

- Когда напряжение на зажимах лампы составляет 110 В, сила тока в лампе составляет 0,5 А. Если через лампу протекает ток 0,25 А, какое напряжение дано лампе?
- Найдите площадь треугольника, в котором одна сторона равна 24 см, другая на 8 см короче, а угол между ними равен 30 градусам.
- Для поднятия бетонной плиты весом 4000 Н на 10 см использовали рычаг. На рычаг ставится сила в 1200 Н. Другой конец рычага прошел 40 см. Найдите коэффициент полезного действия рычага ( $\eta = A_1/A_2 \cdot 100\%$ )

## 10-11 уроки. МОДЕЛЬ И ЕЕ ВИДЫ



Посмотрите на картинку.

Что вы видите? Какой процесс изображен? Что означают буквы в формуле?

Окружающий нас мир – это мир предметов и событий. Как правило, все, что привлекает внимание человека, что его интересует и что он исследует, можно называть объектом. Всё: растения, животные, реки, горы, страны, птицы и дома – становятся объектами человеческого познания. Знать о какой-то вещи или процессе означает познание какой-либо информации об этом.

Обычно предметы именуются аспектами, привлекающими внимание людей. Объект – это часть окружающей человека реальности, воспринимаемая (предметы), процессы, события).

Каждый объект имеет название, отличающее его от других объектов. Обычно люди, отвечая на вопрос: «**Кто это?**» или «**Что это?**», находят название объекта, отличающее его от других объектов.

Имена можно разделить на два типа: общее имя для обозначения нескольких объектов и имя собственное – используется для обозначения некоторого объекта в определенном наборе. У однотипных объектов есть свои характеристики – **описания**.

Каждый отдельно взятый объект отличается от другого соответствующим **значением данного описания**.

### Запомните!

**Объект** (лат. *objectum* – вещь, инструмент) – философская категория, обозначающая вещь, явление или процесс, на которые направлена предметно-практическая, управляющая и познавательная деятельность субъекта; при этом в качестве объекта может выступать и сам субъект.



Объекты-предметы

Помимо названия в сообщении об объекте могут подробно излагаться такие его характеристики, как: свойство, поведение, условия, состояние.

В следующей таблице приведены объекты, их свойства, а также соответствующие этим свойствам количества и значения.



Объекты-процессы



Объекты-события

| Название объекта | Свойство     | Единица измерения | Значение          |
|------------------|--------------|-------------------|-------------------|
| Человек          | Голубоглазый | Цвет глаз         | Голубой           |
| Человек          | Высокий      | Рост              | 178 см            |
| Файл             | Старый       | Время создания    | 07.09.1979        |
| Файл             | Графический  | Тип               | Рисунок типа .bmp |

Человек посредством изучения вещей или событий, которые не всегда можно воплотить в жизнь, пытается познать мир. Именно поэтому создаются и изучаются модели, воссоздающие эти объекты или события. Модель считается упрощенной версией реального объекта.

Можно говорить не только про свойства объекта, но и про отношения его с другими объектами, про их схожие стороны. Например: «Нафиса – дочь Сахобиддина ака»; «число 21 кратно 3»; «Самарканд такой же древний город, как и Бухара». В каждом из этих примеров называются отношения между двумя объектами.

Отношения могут существовать не только между двумя, но и между несколькими объектами одновременно. Например: «Диск считается носителем информации»; «Камчатка – это полуостров».

В большинстве случаев исследование в определенной области – это не реальный объект, а в некотором смысле его копия. С одной стороны, в силу некоторых причин не бывает возможности изучать объект напрямую (нестабильность молнии, расстояние до Солнца, работа с объектом требует больших затрат или представляет опасность для жизни людей и т.д.), а с другой стороны, для исследований может быть достаточно изучения копии объекта. Конечно, в таких случаях копия объекта должна полностью соответствовать требованиям области исследования.

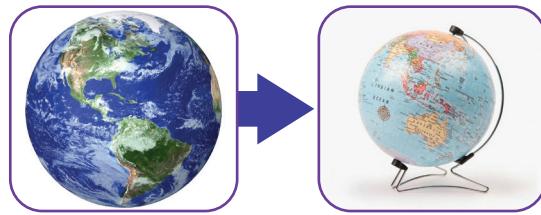
**Модель** (лат. modulus – мера, норма) – это изображение или копия реального объекта или системы объектов, которые соответствуют определенным требованиям области исследования.

**Моделирование** – это процесс построения, изучения и применения моделей, исследование объектов познания (физических явлений и процессов) с использованием их моделей, создание моделей реально существующих предметов и событий с целью их изучения.

Можно привести много реальных примеров моделей объектов. Например, модель Земли – глобус или карта.

Человечество испокон веков было заинтересовано в создании благополучных условий для жизни, предсказании стихийных бедствий. Поэтому вполне естественно изучение различных событий окружающего мира.

Специалисты в области точных наук изучают только те свойства того или иного процесса, которые их интересуют. Например, геологи изучают историю развития Земли, когда, где и какие животные жили, какие растения росли, как менялся климат. Это помогает им найти месторождения полезных ископаемых.



Информация, полученная в результате изучения окружающего нас мира, может быть неточной и неполной. Но полет в космос, разгадывание тайн атомного ядра, овладение законами развития общества не мешают другим. На их основе можно создавать модель изучаемого события или процесса, которая должна максимально полно отражать их характеристики.

Реальный объект и его модель соответствуют требованиям области исследования только в том случае, если они дают одинаковые результаты в экспериментах. Например, самолет и его маленькая модель подчиняются одним и тем же законам аэродинамики. Выведенные для модели результаты также действительны для реальных самолетов. После того как спроектированный реальный самолет построен, он испытывается на специальном оборудовании в лаборатории – стенах, которые направляют поток воздуха на самолет. В этом случае лабораторные стенды служат моделью атмосферы.

Ни одна модель не учитывает все особенности поведения прототипа, поэтому, результат, полученный на основе модели, считается близким к реальности.

Степень приближения зависит от степени совместимости модели. При создании модели человек прежде всего стремится выбрать наиболее важные характеристики объекта и в результате игнорирует те особенности функций, которые не оказывают значительного влияния.

### **А теперь познакомимся с некоторыми типами моделей.**

**1. По области применения:** учебные, экспериментальные, игровые, имитационные, исследовательские модели.

Учебные модели используются в учебном процессе. К ним относятся наглядные пособия, тренажеры, обучающие программы.

Экспериментальные модели используются для изучения объекта и прогнозирования (предсказывания) его будущих характеристик.

Например, модель крыла самолета продувается в аэродинамической трубе для исследования ее сопротивления воздушным потокам.

Научно-технические модели создаются для изучения процессов и событий. К таким моделям можно отнести устройство для получения грозового электрического разряда, модель движения планет Солнечной системы, модель двигателя внутреннего сгорания.

Игровые модели – это разные игры: развивающие, экономические, военные. При помощи таких моделей можно выстраивать разрешение конфликтных ситуаций, продумывать оказание психологической поддержки, прогнозировать поведение объекта в различных ситуациях.

Имитационные модели не только отражают реальность с разной степенью

точности, но и имитируют (повторяют) её. Эксперименты с моделью многократно повторяются в разных стартовых ситуациях или одновременно проводятся с несколькими одинаковыми объектами, помещенными в разные условия. В ходе этих экспериментальных работ изучаются и оцениваются в реальном времени последствия любых действий. По результатам экспериментов подводятся итоги.

## **2. По временному фактору (динамике): статические и динамические модели.**

Статические модели представляют объект без учета изменений, происходящих с ним в определенный период времени. Эти модели не имеют временного фактора. Статической моделью могут служить график изменения средней температуры воздуха за неделю, макет или рисунок молекулы воды, состоящей из атомов водорода и кислорода.

Динамические модели представляют собой процесс изменения объекта с течением времени. Например, карта конкретной местности, результат одного осмотра в клинике.

## **3. По способу изложения: информационные (нематериальные, абстрактные) и материальные модели.**

Материальные модели – это материальные копии объектов моделирования. Например: глобус – модель формы Земли, кукла – модель внешности человека, робот – модель человеческого поведения на вредном производстве.

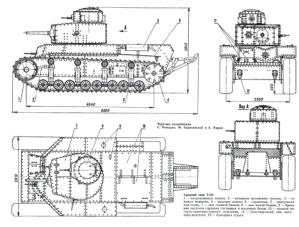
В моделировании материалов используется экспериментальный метод понимания, а в моделировании нематериальных объектов – теоретический метод понимания.

Информационная модель – набор данных, описывающих свойства и состояние объекта, процесса или события.

Информационные модели, в свою очередь, делятся на символические (специальные символы, передаваемые посредством любого формального языка), вербальные (передача мысленных форм в устной речи) и смешанные информационные модели.

Символьные информационные модели строятся с использованием разных языков (систем символов). Символьная информационная модель может выражаться в форме текста на естественном языке или программы на языке программирования, формулы (например, площадь прямоугольника может быть представлена как  $S = ab$ ). Примеры символьных информационных моделей – это географические карты, графики, диаграммы и т. д.

### **Примеры символьных информационных моделей**

|   |   |   |
|---|---|---|
|  |  |  |
| Схема   | План квартиры   | Чертеж  |

Вербальные модели — это информационные модели в мысленной или разговорной форме. Они состоят из визуальных форм объектов, отраженных на любом носителе (бумаге, фото, пленке и т. д.). Эти типы моделей широко используются в сфере образования (картинки в учебниках, обучающие плакаты по различным дисциплинам) и в предметах, в которых необходимо классифицировать объекты по их внешним признакам (ботаника, биология, палеонтология и др.).

В информатике применяются модели, которые можно создавать и исследовать при помощи компьютера. При этом символические модели делятся на компьютерные и некомпьютерные.

Компьютерная модель — это математическая модель, представленная средствами программной среды.

В настоящее время существует два типа компьютерных моделей:

- структурно-функциональные модели — условное выражение объекта, описываемого с использованием компьютерных технологий;
- имитационные модели — это программа или программный комплекс, позволяющий выражать процессы работы объекта в различных условиях.

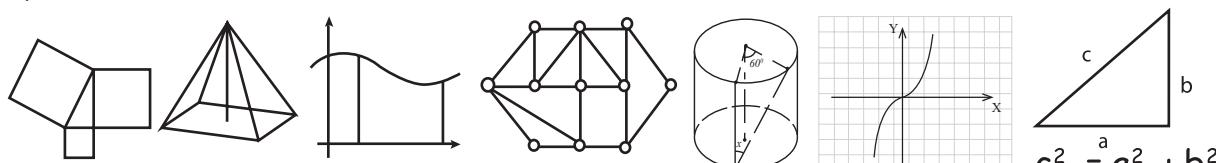
### Примеры компьютерных моделей



#### 4. По средствам представления объектов:

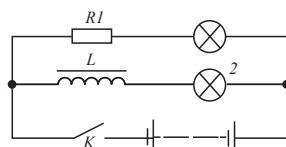
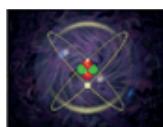
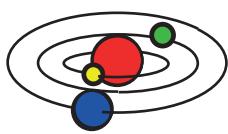
а) **Абстрактные модели**, которые, в свою очередь, делятся на две группы: **математические и экономико-математические модели**.

Математические модели состоят из математических соотношений структуры объекта и законов взаимодействия, формул и математико-логических описаний. Математические модели — это система математических символов, описывающих процесс или событие.



С помощью экономико-математических моделей анализируются наиболее общие законы экономического развития. Для прогнозирования и анализа различных экономических показателей, таких как национальный доход, потребление, занятость, сбережения и инвестиции, используются сложные экономические модели. Для изучения конкретных хозяйственных ситуаций используют малые экономические системы, а для проверки сложных экономических систем обычно используют математические модели.

б) В **физических моделях** характер и структура объекта такие же, как и у оригинала, но отличаются по количеству (размеру, скорости и т. д.). Например, модели самолетов, кораблей, автомобилей, поездов и многое другое.

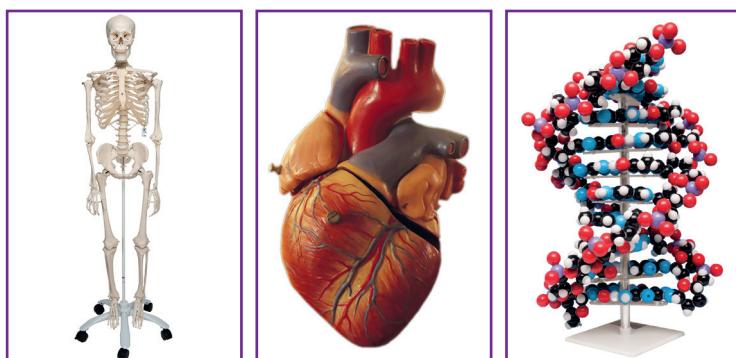


$$F = k \frac{q_1 \cdot q_2}{r^2}$$

в) **Биологическая модель** используется при моделировании биологической структуры, функций и процессов, характерных для различных живых объектов и их частей (клеток, организмов и т. д.)

Биологическая модель даёт возможность тестирования на лабораторных животных различных состояний или болезней, встречающихся у людей и животных. В этом случае экспериментально изучаются механизмы возникновения или течения данного состояния, болезни и их последствия.

На сегодняшний день широко используется компьютерная томография для наблюдения за моделями биологических процессов, то есть для диагностики заболеваний человека с помощью компьютера.



1. Что такое объект?
2. Приведите примеры объектов с общими и специальными названиями.
3. Что такое модель? Сколько существует типов моделей?
4. Приведите примеры объекта и соответствующих ему моделей.
5. Что такое математическая модель? Приведите примеры.
6. Что такое символьная модель? Приведите примеры.
7. Объясните разницу между математической и другими моделями.
8. Что такое информационные модели? Приведите примеры.



ВОПРОСЫ И  
ЗАДАНИЯ

#### Упражнения:

1. Напишите качества и свойства для следующих объектов:



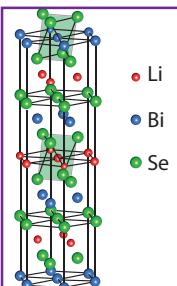
ДОМАШНЕЕ  
ЗАДАНИЕ

| Объект       | Качество | Свойство                  |
|--------------|----------|---------------------------|
| Человек      | Рыжий    | Высокий, открытый, щедрый |
| Книга        |          |                           |
| Жесткий диск |          |                           |
| Монитор      |          |                           |
| Компьютер    |          |                           |
| Файл         |          |                           |

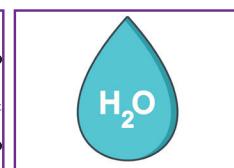
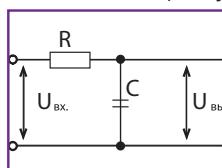
2. Назовите два объекта, обозначающих (выполняющих) движения человека:  
а) сбор; б) пополнение; в) открытие; г) привязка; д) установка; е) измерение;  
ж) захват.

3. Сопоставьте слова в левом и правом столбцах:

|          |                |
|----------|----------------|
| Газета   | Объект-предмет |
| Радуга   | Объект-процесс |
| Прогулка | Объект-событие |
| Стадион  |                |
| Учеба    |                |
| Пир      |                |



4. Расскажите, к какому типу модели относятся рисунки:



## 12 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Ниже приводится анализ задач, связанных с математическими моделями экономических, физических и биологических процессов.

**Задача:** Создайте модель решения проблемы (математический фокус) угадывания задуманного числа.

**Решение:** каждый из вас может придумать произвольное число и сделать следующее:

- 1) умножьте задуманное число на пять;
- 2) добавьте к результату умножения число, соответствующее сегодняшней дате (или любое другое число);
- 3) удвойте полученную сумму;
- 4) прибавьте к результату число текущего года.

Теперь рассмотрим модель, которая соответствует математической цели поиска числа, о котором вы думаете. Для этого прежде всего формализуем задачу: **X** – задуманное число; **U** - результат расчета; **N** - дата; **M** - текущий год.

Итак, приведенные выше инструкции 1–4 можно представить формулой  $U=(X*5+N)*2+M$ . Эта формула служит математической моделью задачи (математический фокус) и представляет собой линейное уравнение относительно переменной X.

Если мы решим уравнение  $X = (U - (M + 2N))/10$ , мы сможем найти задуманное число. А формула  $X = (U - (M + 2N))/10$  – это алгоритм нахождения задуманного числа.



1. На экране размером 2048 x 1366 отображается только 4 разных цвета. Определите объем информации на экране.
2. Если периметр четырёхугольника составляет 54 см, а стороны имеют соотношение 3: 4: 5: 6, найдите длины его сторон.
3. Предмет весом 200 г подбросили вертикально вверх со скоростью 15 м / с. Какова кинетическая энергия объекта и его потенциальная энергия относительно точки выброса через 1 с? Считать g = 10 м / с.
4. Сколько мг воды нужно добавить к 200 мг 12% раствора, чтобы получился 9% раствор?
5. Скоростной поезд «Афросиаб» из Ташкента прошел путь 550 км для прибытия в Бухару. На карте это расстояние равно 11 см. Определите масштаб карты.

# 13-14 УРОКИ. ПОНЯТИЕ АЛГОРИТМА И ЕГО СВОЙСТВА

Каждый день в течение всей жизни человек ставит перед собой цели: решить какие-то задачи, выполнить задание, составить план действий или сделать что-то по плану. Например, записать в блокнот пути решения какой-либо задачи, воспользоваться рецептом при приготовлении какого-нибудь блюда, изучить инструкцию по эксплуатации бытовой техники, объяснить кому-нибудь, как добраться до места назначения и так далее.

1. Какие цели мы ставим, прежде чем начать действовать?
2. Что такое алгоритм? Знаете ли вы историю его возникновения?
3. Что такое исполнитель алгоритма и кто может быть исполнителем алгоритма?



ЗНАЕТЕ ЛИ ВЫ?

Слово и понятие «алгоритм» напрямую связаны с именем великого ученого Абу Абдулла Мухаммад ибн Муса аль-Хорезми, жившего в IX веке (783-850). Слово алгоритм произошло от искаженного произношения имени Аль-Хорезми европейскими учеными. Аль-Хорезми в своем трактате по арифметике «Аль-Китаб аль-Мухтасар фи Хисаб аль-Джабр и аль-Мукаバラ» впервые объяснил правила выполнения четырех операций в десятичной системе счисления. А имя ученого – «аль-Хорезми» в виде «алгоритма» навечно закрепилось в науке.



Каждая инструкция или команда в алгоритме подразумевает выполнение какого-либо действия. Объект, выполняющий действия в алгоритме, можно назвать исполнителем. Любой алгоритм является правилом, определяющим действия, а результат его цепочки – желаемым результатом. Такая цепочка операций называется алгоритмическим процессом, а каждая операция, имеющая общий аспект в составе алгоритма, называется этапом алгоритма.

Алгоритм – это последовательность четких и понятных команд для выполнения исполнителем ради достижения определенных целей.

В качестве примера рассмотрим алгоритм поиска и чтения интересной книги в интернете:

- 1) войти в поисковую систему интернета;
- 2) найти интересную книгу;
- 3) скачать найденную книгу;
- 4) прочитать книгу;
- 5) сделать выводы из прочитанной книги.

## Запомните!

Обычно человек достигает своих целей путем выполнения определенных действий, иначе говоря, выполняя инструкции или последовательность команд, основываясь на своем жизненном опыте или приобретенных знаниях.

Эта последовательность инструкций и действий называется словом «алгоритм».



Алгоритм – это четкая последовательность действий, направленная на достижение поставленной цели или решения задачи (рисунок 1). Алгоритм состоит из последовательности команд, инструкций, действий или движений, то есть шагов, которые необходимо выполнить. Только при правильно составленном алгоритме можно добиться нужного результата.



Например, для достижения своей цели в алгоритме чтения книги человек должен как исполнитель выполнить алгоритм. Другими словами, цель достигается только тогда, когда кто-то выполняет последовательность инструкций или команд.

Среди повседневных правил нашей жизни есть правила, требующие ряд действий, которые приводят к обязательному результату и представлены одним из основных понятий информатики – словом «алгоритм».

Многие алгоритмы повторяются на протяжении всей жизни и поэтому становятся привычками. Например, готовить, есть, аккуратно одеваться, выходить из комнаты, писать, переходить с одного места на другое и так далее.

Итак, алгоритм – это последовательность из определенного числа инструкций или команд (действий), а человек, выполняющий эти алгоритмы, является исполнителем.

В роли исполнителя алгоритма могут выступать природа, человек, автоматизированное устройство (компьютер, техника, робот) и другие. Если для решения какой-либо задачи используются технические устройства, последовательность инструкций или команд должна быть точной и понятной. Чем яснее и понятнее последовательность команд и инструкций, тем быстрее и точнее будет результат. Набор инструкций или команд, которые может выполнить исполнитель, называется системой команд исполнителя (сокращенно СКИ).

**Чтобы разобраться в системе команд исполнителя, рассмотрим следующее задание.**

**1 пример.** Робот может двигаться вправо, вверх или вниз по ячейкам. Он может проходить через одну и ту же ячейку более одного раза. Напишите последовательность необходимых инструкций для перехода от ячейки, где находится робот, к ячейке с аккумулятором. Система команд робота исполнителя (СКР):

**СКР = {направо; вверх; вниз}.** Теперь в качестве решения проблемы можно рассматривать один из следующих алгоритмов:

| Количество шагов | 1 алгоритм | 2 алгоритм | 3 алгоритм | 4 алгоритм | 5 алгоритм |
|------------------|------------|------------|------------|------------|------------|
| 1                |            |            |            | 1) вверх   | 1) вверх   |
| 2                | 1) направо | 1) вниз    | 1) направо | 2) направо | 2) направо |
| 3                | 2) вниз    | 2) направо | 2) направо | 3) направо | 3) вниз    |
| 4                | 3) направо | 3) направо | 3) вниз    | 4) вниз    | 4) вниз    |
| 5                |            |            |            | 5) вниз    | 5) направо |

Это означает, что алгоритм, решающий проблему, может быть не уникальным.

### Теперь познакомимся с основными свойствами алгоритмов.

**Дискретность.** Алгоритм должен выражаться в виде ограниченного числа последовательности простых инструкций для выполнения.

**Определенность.** Инструкции, данные исполнителю в алгоритме, должны иметь однозначный смысл, быть четкими и выполняться только в строго указанном порядке.

Неопределенности в рекомендациях будут препятствовать достижению целей. Например, такие рекомендации как: «Пройдите немного вправо» (в слове «немного» какое расстояние имеется в виду – метров 100 или 50?), «Добавьте достаточное количество сахара» (сколько сахара имеется в виду – 1 чайная или 1 столовая ложка?), «Запустить программу» (какую?) – приводят к разным (часто ненужным) результатам.

**Понятность.** Инструкции должны соответствовать возможностям исполнителя и входить в набор его команд. В противном случае исполнитель не сможет выполнить даже простую операцию. Если исполнитель – человек, то алгоритм должен быть составлен на понятном ему языке, учитывая его знания, жизненный опыт, профессиональную квалификацию, возраст, а также физические возможности. Если исполнитель – техническое оборудование (например, компьютер, электронные часы, машина), то алгоритм должен составляться с учетом возможностей этого оборудования.

Это означает, что любая инструкция должна быть взята из системы команд исполнителя, то есть исполнитель должен знать, как её выполнять.

**Массовость.** Каждый алгоритм должен подходить для решения подобных задач с разными исходными данными. Например, нахождение общего знаменателя для двух простых дробей подходит для нахождения общего знаменателя любых дробей; алгоритм Евклида для нахождения наибольшего общего делителя (НОД) уместен для любых натуральных чисел.

**Пример 2:** Найдите наибольший общий делитель натуральных чисел  $n$  и  $m$ .

- 1) начать;
- 2) если  $n = m$ , взять в качестве результата  $n$  и перейти к пункту 6;
- 3) определить наибольшее из чисел  $n$  и  $m$ ;
- 4) большее из чисел  $n$  и  $m$  равно разности с меньшим;
- 5) перейти к пункту 1;
- 6) конец.

**Результативность.** Каждый алгоритм должен приводить к решению задачи за конечное число шагов. Важно отметить, что после выполнения некоторого количества шагов задача может быть не решена, и это также является результатом.

Алгоритм не всегда может приводить к ожидаемому результату. Причиной может быть неправильное составление алгоритма или другие ошибки. Но отрицательный результат тоже считается результатом.

**Пример 3:** Вычислите площадь треугольника со сторонами  $a$ ,  $b$ ,  $c$ .

Используя приведенный ниже алгоритм «Проверка на существование треугольника», сначала определяем, существует треугольник или нет. Если треугольник существует, то вычислим его площадь, в противном случае не вычисляем. Это тоже считается результатом.

- 1) определить значения  $a$ ,  $b$ ,  $c$ ;
- 2) если  $a <= 0$ , или  $b <= 0$ , или  $c <= 0$ , посчитать что длина сторон треугольника не может быть отрицательным числом и перейти к шагу 5;
- 3) если  $((a+b <= c) \text{ и } (a+c <= b) \text{ и } (b+c <= a))$ , принять треугольник как не существующий и перейти к шагу 5;
- 4) вычислить полупериметр  $P = (a+b+c)/2$ ;
- 5) площадь:  $S = \sqrt{(P*(P-a)*(P-b)*(P-c))}$ ;
- 6) конец.

Только при соблюдении всех вышеперечисленных свойств последовательность инструкций или команд может считаться алгоритмом и иметь какой-то результат (положительный или отрицательный).



1. Какие инструкции исполнитель не может выполнить?
2. Перечислите основные свойства алгоритма.
3. Приведите пример свойства понятности.
4. Из какой системы должны быть взяты инструкции, чтобы быть понятными исполнителю?
5. Объясните суть свойства дискретности алгоритма.
6. Приведите примеры свойства результативности алгоритма.
7. Приведите пример последовательности инструкций, в которой не выполняется свойство результативности.
8. Объясните свойство массовости алгоритма на примерах.



1. Напишите алгоритм вычисления среднего арифметического значения чисел  $x$  и  $y$ .
2. Подготовьте небольшую проектную работу (в виде презентации), посвященную расчёту среднего геометрического значения любых 3 чисел с помощью калькулятора.

## 15–16 уроки. ВИДЫ АЛГОРИТМОВ И СПОСОБЫ ИХ ПРЕДСТАВЛЕНИЯ



1. Какими способами можно описывать алгоритмы?
2. Может ли алгоритм состоять из формул?
3. Может ли алгоритм состоять из графики?

В примерах, рассмотренных на предыдущем уроке, мы описывали алгоритмы посредством слов и математических формул. Но алгоритмы могут быть описаны и другими способами. Ниже мы познакомимся с наиболее распространенными способами описания алгоритмов.

**1. Выражение алгоритма при помощи слов.** В этом способе каждая инструкция для исполнителя дается посредством выражений, слов в виде команды. Каждая команда алгоритма выражается при помощи слов, понятных исполнителю. Приведем пример выражения алгоритма при помощи слов.

**Задача 1.** Вычисление периметра, диагонали и поверхности прямоугольника по его сторонам.

- 1) начать;
- 2) ввести значение сторон ( $a, b$ );
- 3) вычислить значение периметра ( $P$ );
- 4) вычислить значение диагонали ( $D$ );
- 5) вычислить площадь ( $S$ );
- 6) вывести значения периметра, диагонали и поверхности;
- 7) закончить.

**2. Представление алгоритма с помощью формул.** В этом способе каждая операция выражается при помощи математических формул. Для выражения операций алгоритма могут быть использованы простые математические обозначения. Этот способ используется при изучении таких точных наук, как математика, физика, химия, иногда этот способ называют аналитическим выражением. Теперь посмотрим, как Задача 1 выражается с помощью формул:

- 1) начать;
- 2) определить значения сторон прямоугольника  $a$  и  $b$ ;
- 3)  $P=2*a+2*b$ ;
- 4)  $D=\sqrt{(a^2+b^2)}$ ;
- 5)  $S=a*b$ ;
- 6) Вывести значения  $P, D$  и  $S$ ;
- 7) закончить.

**3. Представление алгоритма в виде таблицы.** Описание алгоритма в виде таблиц тоже используется часто. Например, при оформлении школьного расписания, таблицы Пифагора, таблицы химических элементов и т.д. Для построения графиков функций мы также пользуемся таблицей значений. Из-за простоты алгоритмов использования в таблицах их легко освоить.

Чтобы нарисовать график функции, мы также создаем таблицу значений аргументов функции и соответствующих значений. Это тоже может служить примером представления алгоритма в виде таблицы. Например, вы знакомы со следующей таблицей, в которой указаны некоторые точки, проходимые исполнителем, который двигается на основе алгоритма  $y = x^2 + 2$ :

|          |           |           |           |           |          |          |          |          |          |
|----------|-----------|-----------|-----------|-----------|----------|----------|----------|----------|----------|
| <b>x</b> | <b>-4</b> | <b>-3</b> | <b>-2</b> | <b>-1</b> | <b>0</b> | <b>1</b> | <b>2</b> | <b>3</b> | <b>4</b> |
| <b>y</b> | 18        | 11        | 6         | 3         | 2        | 3        | 6        | 11       | 18       |

**4. Графическое представление алгоритма.** Такой способ представления алгоритма вам знаком, так как большинство графиков, изучаемых в курсе математики, являются примерами графического представления алгоритмов. Также можно привести в качестве примеров схемы расположения домов или построек в

городских или жилых кварталах, карты и схемы для поиска и передвижения между домами или строениями, схемы маршрутов автобусов.

Другой удобной графической формой изучения основ алгоритма является блок-схема.

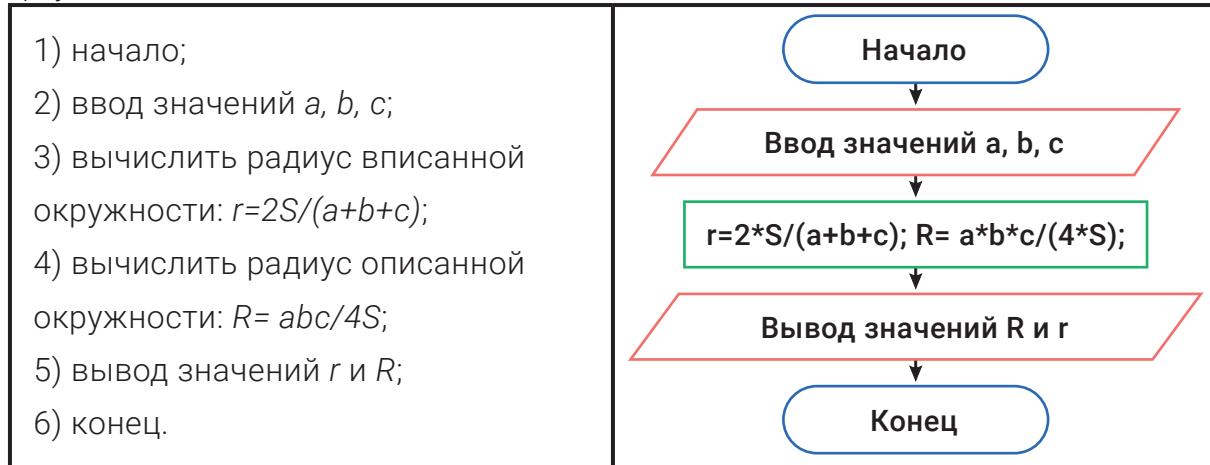
Блок-схемы состоят из специальных геометрических фигур – блоков, представляющих собой конкретную команду или инструкцию, соединенную направляющими линиями.

Основные простые геометрические фигуры, используемые при построении блок-схем, состоят из:

| Название блока                        | Внешний вид блока | Назначение блока   |
|---------------------------------------|-------------------|--|
| Начало/конец алгоритма                |                   | Используется в начале и конце алгоритма  |
| Ввод/вывод                            |                   | Служит для ввода начальных (входных) данных и вывода полученных результатов  |
| Функциональный блок (блок операторов) |                   | Служит для присвоения значений или выполнения инструкций.<br>Внутри четырехугольника записываются команды, которые должны выполняться  |
| Блок условия                          |                   | Через проверку условия определяется направление алгоритма. Если условие внутри ромба верно, управление передается направлению «да», а в противном случае в направлении «нет» |
| Блок цикла                            |                   | Используется при цикле с параметром. Необходимо знать число повторений и шаг цикла.<br>Внутри блока указываются начальный и конечные значения и шаг изменения                |
| Блок подпрограммы                     |                   | Используется для обращения к созданному заранее вспомогательному алгоритму   |
| Блок печати сообщений                 |                   | Используется для печати результатов  |
| Линия перехода в блок-схеме           |                   | Указывает направление движения в блок-схеме  |
| Оператор присваивания                 |                   | Считается инструкцией к присваиванию   |

**Задача 2.** Треугольник определен его сторонами. Вычислить радиусы и длины вписанных и описанных окружностей.

Радиус вписанной окружности равен  $r=2S/(a+b+c)$ , радиус же описанной окружности равен  $R=abc/4S$ . Где  $S$  – площадь треугольника,  $a$ ,  $b$ ,  $c$  – длины сторон треугольника.



Учитывая, что блок-схемы являются одним из наиболее удобных средств выражения алгоритмов и обладают большим визуальным потенциалом, мы используем их при изучении программирования. Поэтому с этого момента нам нужно научиться работать с блок-схемами.

ВНИМАНИЕ

## 5. Представление алгоритма в виде программы.

Сегодня доступно множество алгоритмических языков, которые называются языками программирования. Алгоритмический язык – это система определений и правил, используемых для однообразного и точного написания алгоритмов. Алгоритмический язык близок к языку общения человека и включает математические символы (как упоминалось выше). Алгоритмы, предназначенные для решения проблем, не могут быть напрямую введены в машину, поэтому необходимо перевести написанный алгоритм на алгоритмический язык.

У каждого алгоритмического языка есть своя область применения. Обычно алгоритм, записанный на языке, понятном компьютеру, называется **программой**. Язык, понятный компьютеру, называется **языком программирования**. В мире существуют тысячи языков программирования, и их число все время растет. На сегодняшний день широко используются языки программирования Pascal, Delphi, C, C++, Java, Python, и их легко изучать.

Основная цель перечисленных способов описания алгоритмов – определить наиболее легкий вариант последовательности действий для решения поставленной задачи и облегчить для человека процесс написания программы. Фактически программа – это еще одна версия алгоритма, которая предназначена облегчить человеку общение с компьютером.



1. Укажите способы описания алгоритма.
2. Приведите примеры из предмета физики, где алгоритм выражается с помощью формул.
3. Приведите примеры графического представления алгоритма.
4. Приведите примеры из реальной жизни, где алгоритм выражается словами.
5. Приведите жизненные примеры, где алгоритм представлен в виде таблицы.
6. Приведите примеры использования алгоритмов в виде таблиц в математике.
7. В каких предметах удобно давать алгоритм с помощью формул?
8. Что такое блок-схема? Расскажите о функциях основных элементов блок-схемы.



1. Объясните словами алгоритм разделения отрезка АВ на две равные части.
2. Создайте алгоритм вычисления площади окружности с радиусом R с помощью графического способа и блок-схемы
3. Напишите алгоритм построения графика функции  $y = x^2 + 3$  в MS Excel, используя слова.
4. Опишите удобным для себя способом создание фотоальбома на тему «Моя семья» в MS Power Point.
5. Создайте алгоритм определения квадратного корня из произвольного числа  $n > 0$  с помощью блок-схемы.

## 17 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ



1. Какие из следующих инструкций вы не можете выполнить как исполнитель, и почему?
  - а) Разместите 45 кроликов в 10 клетках в неравном количестве;
  - б) Вычислите площадь квадрата со сторонами 12 см;
  - г) Определите количество построек на земле при полете на высоте 5000 м;
  - е) Создайте число 66, используя четыре числа 6 и операции +, -, \*.
2. Определите исполнителей данной системы инструкций:
  - а) {информация: сбор; хранение; обработка; передача; распечатка};
  - б) {пойти в библиотеку; достать книгу; дочитать книгу до конца; отдать книгу в библиотеку};
  - в) {включить компьютер; запустить MSWord; создать файл «Моя семья!»; сохранить документ; распечатать документ; закрыть программу и выключить компьютер}.
3. Составьте алгоритм решения следующих задач с помощью слов:
  - а) инструкции исполнителя состоят только из {прибавить 5; умножить на 3; вычесть 6; разделить на 2}. Составьте такой алгоритм, чтобы этот исполнитель из числа 12 сгенерировал число 20;
  - б) создать алгоритм вычисления значения функции  $y = ax + 2b$  по заданным числам a, b, x;
  - г) создать алгоритм отправки СМС с телефона близкому вам человеку;
4. С помощью блок-схемы создайте алгоритмы для следующих задач:
  - а) Найдите площадь и периметр прямоугольного треугольника по двум заданным катетам a и b;
  - б) Найдите площадь круга, ограниченного окружностью длиной L;
  - в) Вычислите сэкономленную сумму денег при покупке товаров на сумму 367 450 сумов, если скидка составляет 12% при покупке товаров более чем на 200 000 сумов.

## 18 УРОК. КОНТРОЛЬНАЯ РАБОТА

1. Определите значение логических выражений на основе следующих значений:

$A = \text{«Клавиатура – устройство ввода»}$ ,  $B = \text{«}1010_2 = 10_{10}\text{»}$ ,  $C = \text{«}128 \text{ байт} = 16 \text{ бит}\text{»}$ :

а)  $A \vee \neg C$ ;    б)  $A \& \neg (B \& \neg C)$ ;    в)  $A \Leftrightarrow (\neg B \Rightarrow \neg C)$ .

2. а)  $X=1$ ;    б)  $X=12$ ;    в)  $X=3$

Для всех этих случаев определите значение логического выражения:  
 $((X>3) \vee (X<3)) \Rightarrow (X<4)$ .

3. Заполните пустые ячейки в таблице истинности:

| A | B | C | $C \vee A$ | $(C \vee A) \Rightarrow B$ |
|---|---|---|------------|----------------------------|
| 0 | 0 |   | 0          | 1                          |
| 0 |   | 0 | 0          | 1                          |
|   | 0 | 0 | 1          | 0                          |
| 1 | 1 | 1 | 1          |                            |

4. Напишите следующее утверждение в форме логического выражения и составьте таблицу истинности: «Я пойду к бабушке домой и, если я встречу там своих друзей, весело проведу время».

5. Создайте таблицу истинности логического выражения:  $(A \vee B) \& (\neg A \& \neg B) \vee \neg C$ .

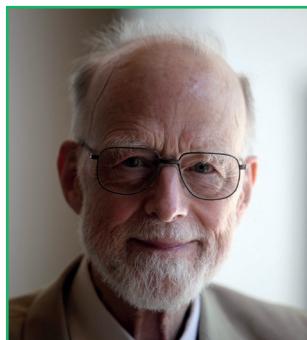
6. Для каких значений переменных X, Y, Z выражение  $X \vee Y \& \neg Z$  примет значение истина?

7. Определите порядок действий в примерах и рассчитайте для случая, когда все они верны (истинны).

а)  $A \& B \vee (\neg C)$ ;    б)  $A \& B \vee C \& E$ ;    в)  $\neg (A \& B) \vee (A \& B)$ ;    г)  $(A \& B) \vee (\neg B) \vee (\neg D)$ .

8. Самолёт «Як-40» пролетел расстояние между городами Ташкент – Бухара за 1 час 10 минут. Если скорость самолета 600 км / ч, определите расстояние между этими городами в сантиметрах на карте с масштабом 1:4 000 000. Определите реальное расстояние между этими городами.

## 19 УРОК. ЛИНЕЙНЫЕ АЛГОРИТМЫ



В 70-е годы XX века голландский ученый Эдсгер Дейкстра (1930–2002) выдвинул и полностью обосновал идею о том, что любой алгоритм, независимо от цели создания и степени сложности, может быть записан в одной из 3 алгоритмических конструкций: последовательность, ветвление и повторение.

Мы знаем, что алгоритм – одно из основных понятий информатики, и это последовательность действий, необходимых для достижения результата.

Любой алгоритм делится на три основных типа в соответствии с его логической структурой, то есть порядком выполнения: линейный, ветвящийся и повторяющийся.

**Линейным алгоритмом** называются процессы, в которых все инструкции, без рассмотрения каких-либо условий, выполняются только последовательно.

Как пример можем привести алгоритмы расчёта результатов сложения или умножения, замену значений нескольких переменных, заваривание чая, вычисление площади круга и так далее.

Рассмотрим пример описания линейного алгоритма словами. Поставлена задача заварить чай. В таком случае человек, заваривающий чай, должен выполнять следующие

действия, которые для нас с вами являются повседневными и простыми:

- 1) открыть крышку чайника;
- 2) ополоснуть чайник кипятком;
- 3) положить в чайник 1 чайную ложку заварки;
- 4) залить кипятком до краев чайника;
- 5) закрыть крышку чайника;
- 6) накрыть чайник полотенцем и оставить на пять минут.

Блок-схема алгоритма с линейной структурой в основном состоит из блока начала алгоритма, блока завершения, блока ввода/вывода и функциональных (операторных) блоков.

Линейная структура представлена в виде набора последовательных инструкций (команд), выполняемых только поочередно, в том же порядке, как они записаны в алгоритме (рисунок 1). Для представления линейного алгоритма используется следующая структура:

| С помощью слов                                      | В виде блок-схемы   | Образец  |
|---|---|--|
| 1 инструкция<br>2 инструкция<br>...<br>n инструкция | <pre>graph TD; A[1 инструкция] --&gt; B[2 инструкция]; B --&gt; C[n инструкция]</pre> | Вычислите периметр и площадь прямоугольника со сторонами $a$ и $b$ |

Рисунок 1. Линейный алгоритм.

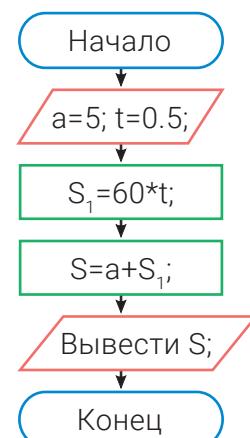
Рассмотрим несколько примеров линейных алгоритмов.

**Пример 1.** Турист отправился из деревни в город. После того, как он прошел  $a$  км пути пешком, сел в автобус и за  $t$  время приехал в город. Если автобус движется со скоростью 60 км / ч, составьте алгоритм расчёта расстояния  $S$  между деревней и городом, с учётом значений  $a = 5$  и  $t = 0,5$ .

Решение: вспомним формулу для расчета расстояния:  $S = v \cdot t$ . Турист ехал в автобусе  $S_1 = 60t$  километров. Поэтому расстояние между деревней и городом выражается формулой  $S = a + 60t$ . А при  $a = 5$  и  $t = 0,5$ ,  $S = 5 + 60 \cdot 0,5 = 35$  км.

Теперь попробуем выразить алгоритм вычисления расстояния  $S$  с помощью слов и блок-схемы:

- 1) начать;
- 2) ввести значения  $a$ ,  $t$ ;
- 3) Рассчитать расстояние, пройденное туристом на автобусе за  $t$  часов по формуле:  $S_1 = 60 * t$ ;
- 4) Рассчитать расстояние от деревни до города по формуле:  
 $S = a + S_1$ ;
- 5) Написать значение  $S$ ;
- 6) Конец.





1. На какие типы делятся алгоритмы по логическому строению?
2. Какие алгоритмы называются линейными?
3. Приведите жизненные примеры линейных алгоритмов.
4. Расскажите про структуру блок-схемы линейных алгоритмов.
5. Опишите алгоритм пути из дома в школу.

1. Определите правильный порядок действий (команд) алгоритма, приведённого в таблице ниже, и запишите его во 2 столбец таблицы. Дайте название алгоритму.

|  |  |
|--|--|
| 1) поставить обувь на место;                         |  |
| 2) пройти вверх по лестнице;                         |  |
| 3) обработать обувь щеткой до блеска;                |  |
| 4) очистить пыль с обуви тряпкой;                    |  |
| 5) принести все в комнату;                           |  |
| 6) положить на место обувную щетку и крем для обуви; |  |
| 7) взять обувную щетку и крем для обуви;             |  |
| 8) выдавить крем на обувь;                           |  |
| 9) взять обувь.                                      |  |

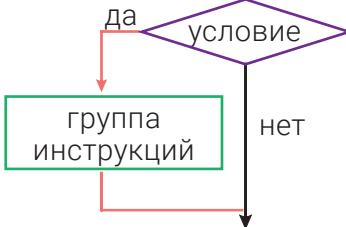
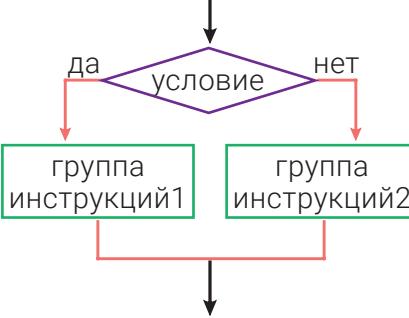
2. Сформулируйте алгоритм вычисления площади треугольника словами и блок-схемой по двум сторонам и углом между ними.

## 20 урок. РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ

Существуют также вычислительные процессы, в которых, в зависимости от выполнения определенных логических условий, процессы разбиваются на несколько сетей, и выполняется хотя бы один из них. Для реализации таких процессов созданы алгоритмы ветвления.

Такие вычислительные процессы называются алгоритмами ветвления, если вычислительный процесс продолжается в разных ветвях в зависимости от выполнения заданного условия, и каждая сеть выполняется только один раз во время вычислительного процесса. Ветвящаяся структура обычно состоит из блока логических условий. Структура ветвления обычно включает проверку какого-либо логического условия. В зависимости от результата проверки выполняется то или иное направление. Разветвленная структура позволяет выбрать один из двух вариантов в зависимости от результата проверки условия (да или нет), то есть он обеспечивает выполнение только одного из указанных ветвлений.

Эти структуры можно разделить на два основных типа – полный и сокращенный. Они могут быть представлены в виде следующих схем:

| Тип                      | С помощью слов   | В виде блок-схемы   | Образец   |
|--------------------------|--|---|---|
| <b>Если - то</b>         | <b>Если условие<br/>то группа<br/>инструкций<br/>конец</b>                                   |  | <b>Если</b> сумма цифр числа делится на 3 без остатка, <b>то</b> это число является кратным 3                                       |
| <b>Если - то - иначе</b> | <b>Если условие<br/>то группа<br/>инструкций1<br/>иначе группа<br/>инструкций2<br/>конец</b> |  | <b>Если</b> сумма цифр числа делится на 3 без остатка, <b>то</b> это число является кратным 3, а <b>иначе</b> не является кратным 3 |

**Пример 1.** Создайте алгоритм расчета квадрата заданного положительного числа **A** со значением больше чем 0 (ноль):

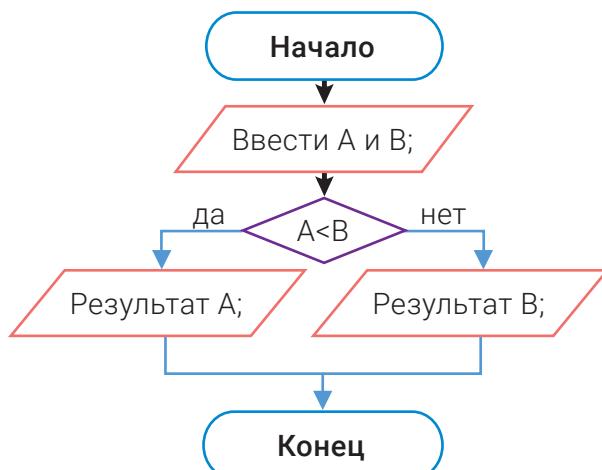
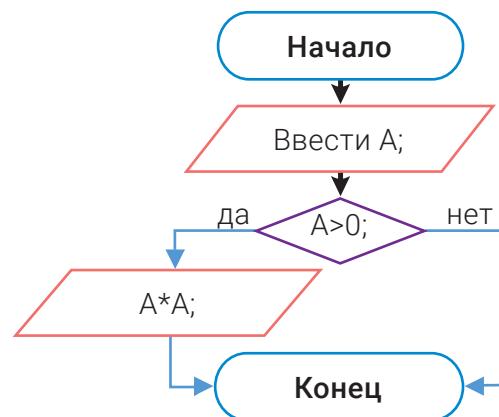
- 1) начало;
- 2) введите **A**;
- 3) если **A > 0**, то перейти к пункту 4;
- 4) пусть результат будут **A \* A**;
- 5) конец.

В этом примере, если  $A > 0$ , то инструкция в пункте 4 выполняется, а в противном случае, то есть если  $A \leq 0$ , инструкции в пункте 3 выполняться не будут.

**Пример 2.** Создайте алгоритм, определяющий наименьшее из двух заданных чисел **A** и **B**:

- 1) начало;
- 2) введите **A** и **B**;
- 3) если **A < B**, перейти к пункту 4; в противном случае переходите к пункту 5;
- 4) пусть результат будет **A**, тогда перейдите к пункту 6;
- 5) пусть результат будет **B**;
- 6) конец.

Из этого примера можно сделать следующий вывод: если выполняется условие  $A < B$ , инструкция пункта 5 не выполняется, иначе в случае  $A > B$  не выполняется инструкция в пункте 4.





- Какие алгоритмы называются алгоритмами ветвления?
- Приведите примеры из жизни, касающиеся алгоритмов ветвления.
- Какие структуры алгоритмов ветвления существуют?
- Объясните разницу между полной и сокращенными версиями алгоритмов ветвления.
- Приведите примеры полной версии представления алгоритмов ветвления.



- Если заданное число а больше 5, создайте алгоритм для вычисления его квадратного корня.
- Составьте алгоритм определения абсолютного значения заданного целого числа.
- Составьте алгоритм вычисления корней квадратного уравнения вида  $ax^2 + bx + c = 0$ .

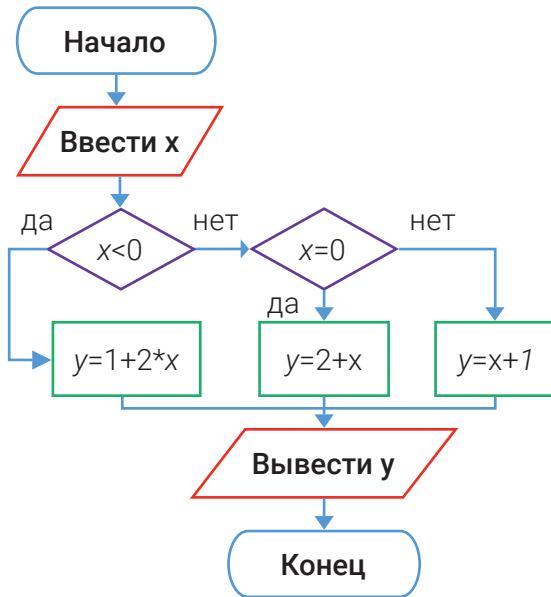
## 21 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Пример:** Алгоритм описан в виде формулы:

$$y = \begin{cases} 1+2x, & \text{если } x < 0 \\ 2+x, & \text{если } x = 0 \\ x+1, & \text{если } x > 0 \end{cases}$$

Составьте в виде блок-схемы алгоритм вычисления значения функции:

- Задачи по линейной структуре:
  - Опишите алгоритм чистки ковра посредством слов;
  - Составьте алгоритм вычисления длины окружности, площади круга и объема сферы с радиусами R (направление:  $L=2\pi R$ ;  $S=\pi R^2$ ;  $V=\frac{4}{3}\pi R^3$ );
  - Составьте алгоритм вычисления количества кирпичей размером 250 мм × 120 мм × 65 мм для стены размерами 2200 мм × 120 мм × 700 мм.



- Задачи, связанные со структурой ветвления:
  - Даны числа a, b и с. Составьте алгоритм вычисления квадратного корня, если сумма их сложения отрицательное число, а если положительное, то вычитывающий из каждого из них число 30.
  - Составьте алгоритм вычисления следующей функции для значений a = 10 и b = 1,5:

$$y = \begin{cases} x^2 + |x - 1| + 2, & \text{если } x > 1,5; \\ \sqrt{3a - 2bx + x^2}, & \text{если } x = 1,5; \\ \frac{2ax}{5} + b, & \text{если } x < 1,5. \end{cases}$$

## 22 УРОК. ПОВТОРЯЮЩИЕСЯ АЛГОРИТМЫ



Что вы видите на картинке?

Если вы пристальнее посмотрите на события, происходящие в природе и вокруг вас, вы можете увидеть несколько циклических (повторяющихся) процессов. Например, смена времён года, дня и ночи, еженедельное повторение занятий, ежедневный обед или умывание после просыпания.



**Повторяющимся алгоритмом** называется алгоритм, в котором, на основе какого-нибудь условия или различных значений параметра, происходит повторение некоторых процессов.

Существуют процессы, определенные части которых повторяются по несколько раз.

Например, ученик, проваливший тест по предмету, то есть получивший «неудовлетворительную» оценку, должен будет читать темы по этому предмету снова и снова и готовиться к тесту, пока не получит «удовлетворительную» оценку. Или же, чтобы вычислить выражение  $9! = 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9$ , необходимо 8 раз произвести умножение.

При построении алгоритмов таких процессов используются повторяющиеся алгоритмы.

Повторяющиеся алгоритмы отличаются от других тем, что в них используются инструкции типа « $i = i + 1$ », « $S = S + i$ » или « $P = P * i$ ». (\* – это операция умножения). Чтобы понять смысл таких инструкций, нужно будет просмотреть несколько шагов повторения.

Обычно начальное значение суммы (от английского SUMM, что означает сумма, заглавная буква слова)  $S = 0$  и для умножения (от англ. PRODUCT, что означает умножение, заглавная буква слова)  $P = 1$ , потому что эти значения, то есть 0 и 1, не влияют на результаты суммирования и умножения:

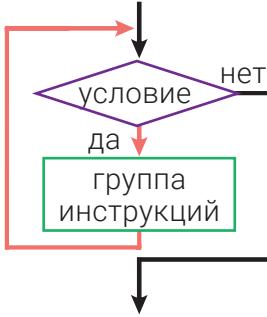
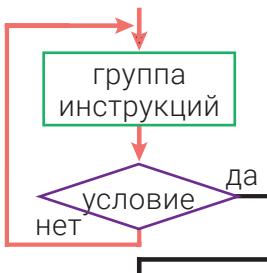
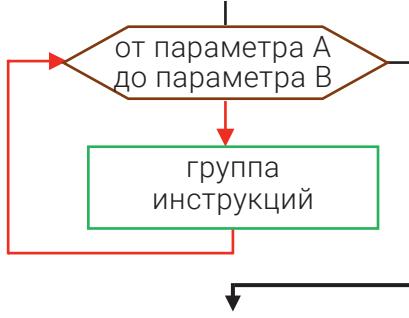
| Шаги   | $i$                         | $S$              | $P$              |
|--------|-----------------------------|------------------|------------------|
| 1 шаг: | Пусть $i=1$ , в этом случае | $S=S+i=0+1=1$ ,  | $P=P*i=1*1=1$ ;  |
| 2 шаг: | $i=i+1=1+1=2$ ,             | $S=S+i=1+2=3$ ,  | $P=P*i=1*2=2$ ;  |
| 3 шаг: | $i=i+1=2+1=3$ ,             | $S=S+i=3+3=6$ ,  | $P=P*i=2*3=6$ ;  |
| 4 шаг: | $i=i+1=3+1=4$ ,             | $S=S+i=6+4=10$ , | $P=P*i=6*4=24$ . |

Повторяющаяся часть вычислительного процесса называется телом внутреннего цикла.

Для осуществления повторяющихся действий (инструкций) существуют специальные алгоритмические структуры, называемые командами цикла. Структуры повторения обеспечивают многократное повторение групп из нескольких инструкций по несколько раз. Эти конструкции отображены в таблице на странице 41.

В предусловных повторяющихся алгоритмах пока сначала проверяется условие, затем, если условие выполнено (истинно), выполняется тело цикла. В противном случае вычисление будет приостановлено.

В постусловных повторяющихся алгоритмах сначала выполняется тело цикла, а затем проверяются выходные данные, то есть тело цикла будет выполняться до тех пор, пока условие не будет выполнено.

| Тип                          | С помощью слов  | В виде блок-схемы  | Образец  |
|------------------------------|---|--|--|
| Предусловие – пока           | Пока (условие);<br>Начало цикла;<br>Группа<br>инструкций;<br>Конец цикла.         |   | Если условие не истинно (ложно), цикл останавливается.<br>Если условие ложно с самого начала, то набор инструкций никогда не выполняется.                              |
| Постусловие – до             | ... Группа<br>инструкций;<br>До (условие).  |   | Если условие уместно (истинно), то цикл останавливается.   |
| С параметром – от ... до ... | От параметра<br>A до B;<br>Начало цикла;<br>Группа<br>инструкций;<br>Конец цикла. |  | Число повторений группы инструкций зависит от начального и конечного значения параметра цикла.<br>Число повторений этого цикла определяется выражением $(B - A + 1)$ . |

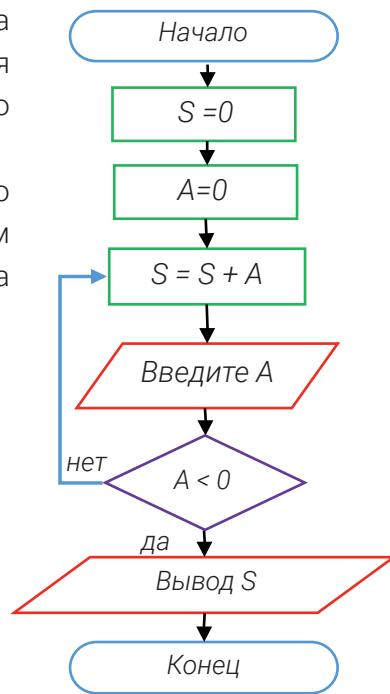
Циклы, в которых условия проверяются до, и циклы, в которых условия проверяются после, вместе являются **итерационными циклами**.

Рассмотрим несколько примеров итерационных процессов.

**Пример 1.** Представьте себе, что вводятся цифры на клавиатуре (1, 6, 8, 2, -6, 76, 1, -5). Создайте алгоритм для вычисления суммы чисел (1, 6, 8, 2), введенных до первого отрицательного числа (-6).

**Решение.** В алгоритме, выраженном с помощью слов, для обозначения совместимости с блок-схемой будем записывать комментарии в скобках. Сумму обозначим **S**, а число, вводящееся с клавиатуры, обозначим через **A**.

- 1) Начало;
- 2) Присвоить значение **S = 0** (т.е. **S = 0**);
- 3) Присвоить значение **A = 0** (т.е. **A = 0**);
- 4) Добавить **A** к **S** и получите **S** (т.е. **S = S + A**);
- 5) Ввести **A**;
- 6) Если **A < 0** неверно, то перейти к пункту 4;
- 7) Вывести результат **S**;
- 8) Конец.



## ВОПРОСЫ И ЗАДАНИЯ



1. Какие алгоритмы называются повторяющимися? Приведите примеры.
2. Опишите повторяющиеся структуры. Объясните разницу между ними.
3. Объясните алгоритм, в котором условие проверяется в начале. Приведите пример.
4. Приведите пример жизненного алгоритма с условием проверки повторений.
5. Создайте алгоритм, определяющий наименьшее из трех чисел.

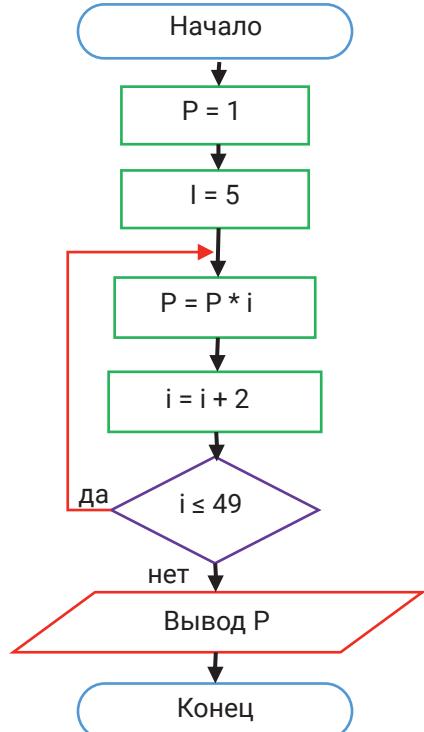
## ДОМАШНЕЕ ЗАДАНИЕ



1. Создайте алгоритм для вычисления суммы числа от 1 до 100 с использованием трех различных повторяющихся структур.
2. Определите формулировку задачи следующего алгоритма и создайте блок-схему:
  - 1) Начало;
  - 2) Присвоить значение  $P = 0$ ;
  - 3) Присвоить значение  $i$  равно 1;
  - 4) Умножить  $P$  на  $i$ , чтобы получить  $P$ ;
  - 5) Прибавить 1 к  $i$  и получаем  $i$ ;
  - 6) Если  $i \leq 50$ , перейти к пункту 4;
  - 7) Принять результат  $P$ ;
  - 8) Конец.

42

## ГЛАВА III. ОСНОВЫ АЛГОРИТМИЗАЦИИ



**Пример 2.** Составьте алгоритм вычисления произведения нечетных чисел от 5 до 49, т.е.  $P = 5 * 7 * 9 * \dots * 49$ .

- 1) Начало;
- 2) Присвоить значение  $P = 1$ ;
- 3) Присвоить значение  $i = 5$ ;
- 4) Умножить  $P$  на  $i$ , чтобы получить  $P$ ;
- 5) Прибавить 2 к  $i$  и получить  $i$ ;
- 6) При  $i \leq 49$  перейти к пункту 4;
- 7) Вывод результата  $P$ ;
- 8) Конец.

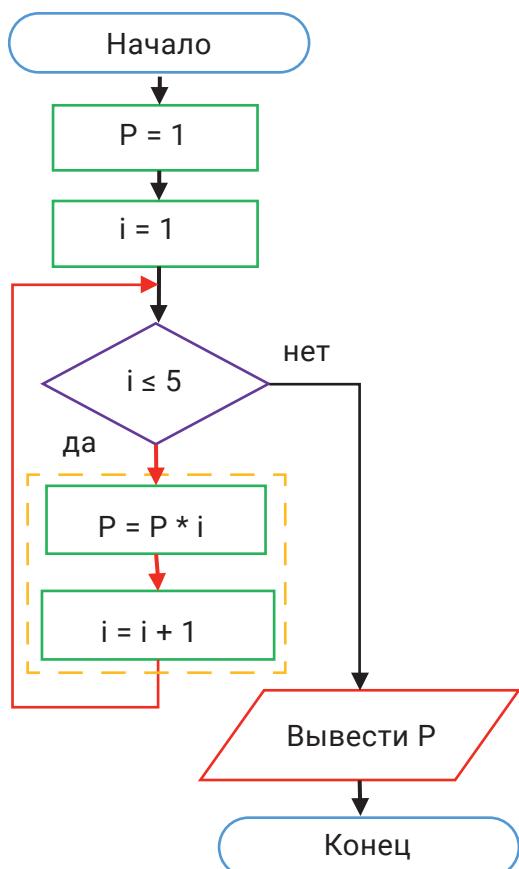
Следовательно, алгоритмы имеют линейные, разветвлённые или повторяющиеся виды и в жизни человека выступают как целостная единица.

## 23 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Пример.** Создайте алгоритм для вычисления произведения чисел от 1 до 5 с использованием трех различных повторяющихся структур.

Математическая модель:  $P = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$

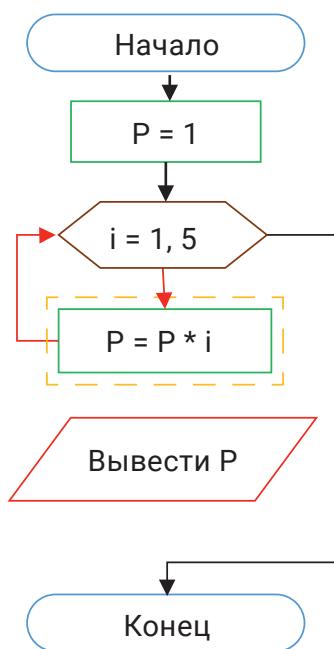
1 способ.



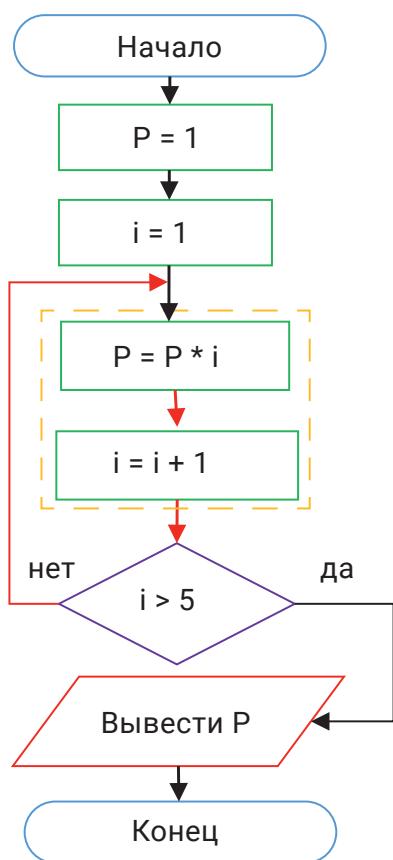
| Шаг | Операции                           | P   | i | Проверка условия         |
|-----|------------------------------------|-----|---|--------------------------|
| 1   | $P=1$                              | 1   |   |                          |
| 2   | $i=1;$                             | 1   | 1 |                          |
| 3   | $i \leq 5$<br>$P=P*i;$<br>$i=i+1;$ | 1   | 1 | $1 \leq 5$ , да (истина) |
| 4   | $i \leq 5$<br>$P=P*i;$<br>$i=i+1;$ | 2   | 2 | $2 \leq 5$ , да (истина) |
| 5   | $i \leq 5$<br>$P=P*i;$<br>$i=i+1;$ | 6   | 3 | $3 \leq 5$ , да (истина) |
| 6   | $i \leq 5$<br>$P=P*i;$<br>$i=i+1;$ | 24  | 4 | $4 \leq 5$ , да (истина) |
| 7   | $i \leq 5$<br>$P=P*i;$<br>$i=i+1;$ | 120 | 5 | $5 \leq 5$ , да (истина) |
| 8   | $i \leq 5$<br>$P=P*i;$<br>$i=i+1;$ |     |   | $6 \leq 5$ , нет (ложь)  |

2 способ.

| Шаг | Операции      | P   | i |
|-----|---------------|-----|---|
| 1   | $P=1$         | 1   |   |
| 2   | $i=1; P=P*i;$ | 1   | 1 |
| 3   | $i=2; P=P*i;$ | 2   | 1 |
| 4   | $i=3; P=P*i;$ | 6   | 3 |
| 5   | $i=4; P=P*i;$ | 24  | 4 |
| 6   | $i=5; P=P*i;$ | 120 | 5 |



3 способ.



| Шаг | Операции              | P   | i | Проверка условия    |
|-----|-----------------------|-----|---|---------------------|
| 1   | P=1                   | 1   |   |                     |
| 2   | i=1                   | 1   | 1 |                     |
| 3   | P=P*i<br>i=i+1<br>i>5 | 1   | 1 | 2>5, нет<br>(ложь)  |
| 4   | P=P*i<br>i=i+1<br>i>5 | 2   | 2 | 3>5, нет<br>(ложь)  |
| 5   | P=P*i<br>i=i+1<br>i>5 | 6   | 3 | 4>5, нет<br>(ложь)  |
| 6   | P=P*i<br>i=i+1<br>i>5 | 24  | 4 | 5>5, нет<br>(ложь)  |
| 7   | P=P*i<br>i=i+1<br>i>5 | 120 | 5 | 6>5, да<br>(истина) |



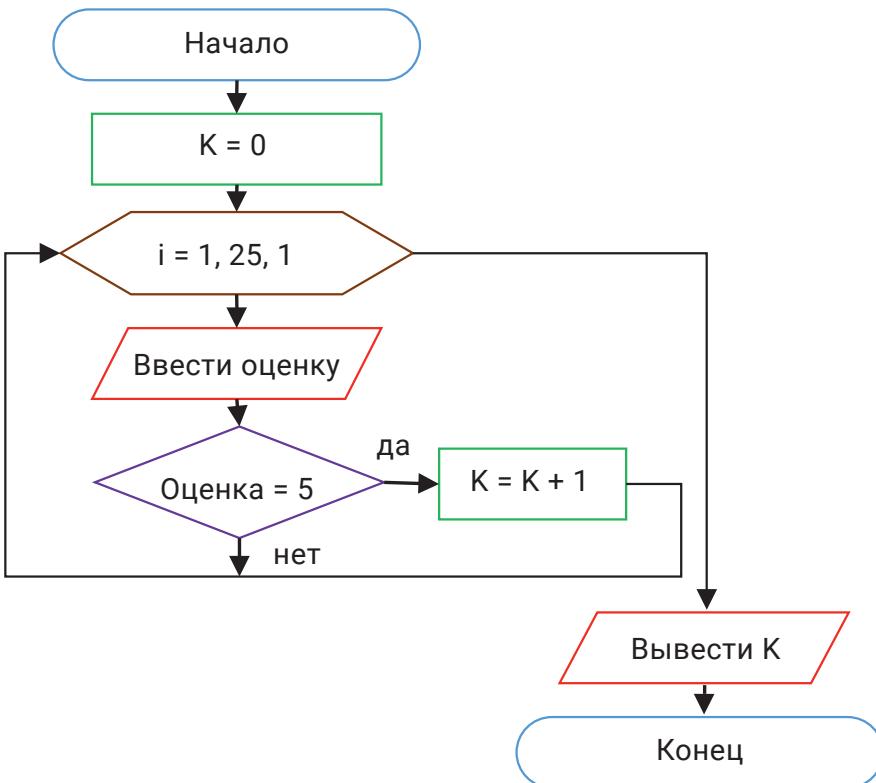
- Создайте алгоритм вычисления суммы чисел от 1 до n кратных к 5.
- Создайте алгоритм вычисления значения выражения  $y = \sin 0 + \sin 0,1 + \sin 0,2 + \dots + \sin n$ .
- Создайте алгоритм вычисления произведения  $P=(2+2)*(2+3)*(2+4)*(2+5)$ , используя 3 различные структуры повторения.
- Создайте алгоритм вычисления функции  $y = 2x^3 + 1$  для целых значений x в интервале  $[-3;5]$  в виде блок-схемы.

## 24 УРОК. СМЕШАННЫЕ (КОМБИНИРОВАННЫЕ) АЛГОРИТМЫ

Алгоритм, в составе которого участвуют несколько типов алгоритмов, называется **смешанным** (комбинированным) алгоритмом.

Примерами смешанных алгоритмов могут служить следующие:

**Задача 1.** В классе 25 учеников. Создайте алгоритм, определяющий количество учеников, получивших оценку «Отлично» по информатике.



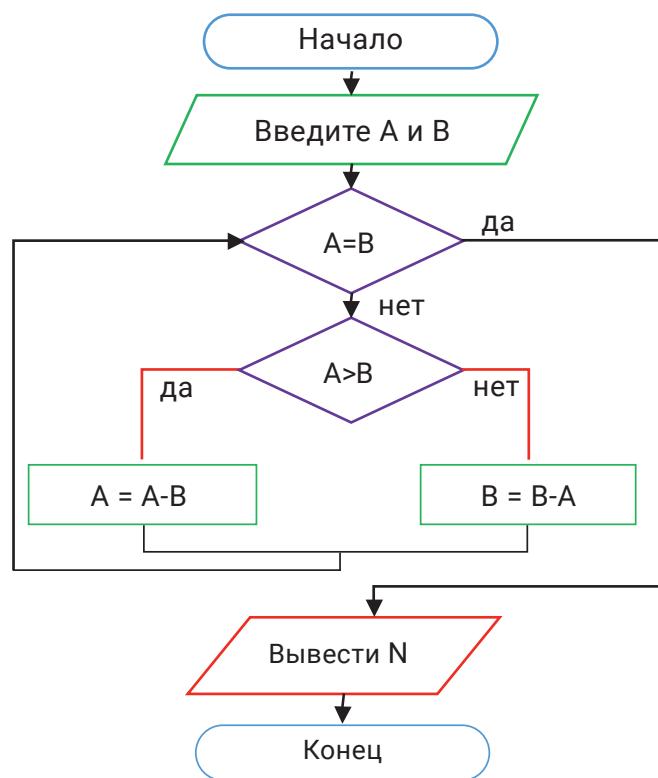
Для решения задачи использовались разветвляющиеся и повторяющиеся алгоритмы.

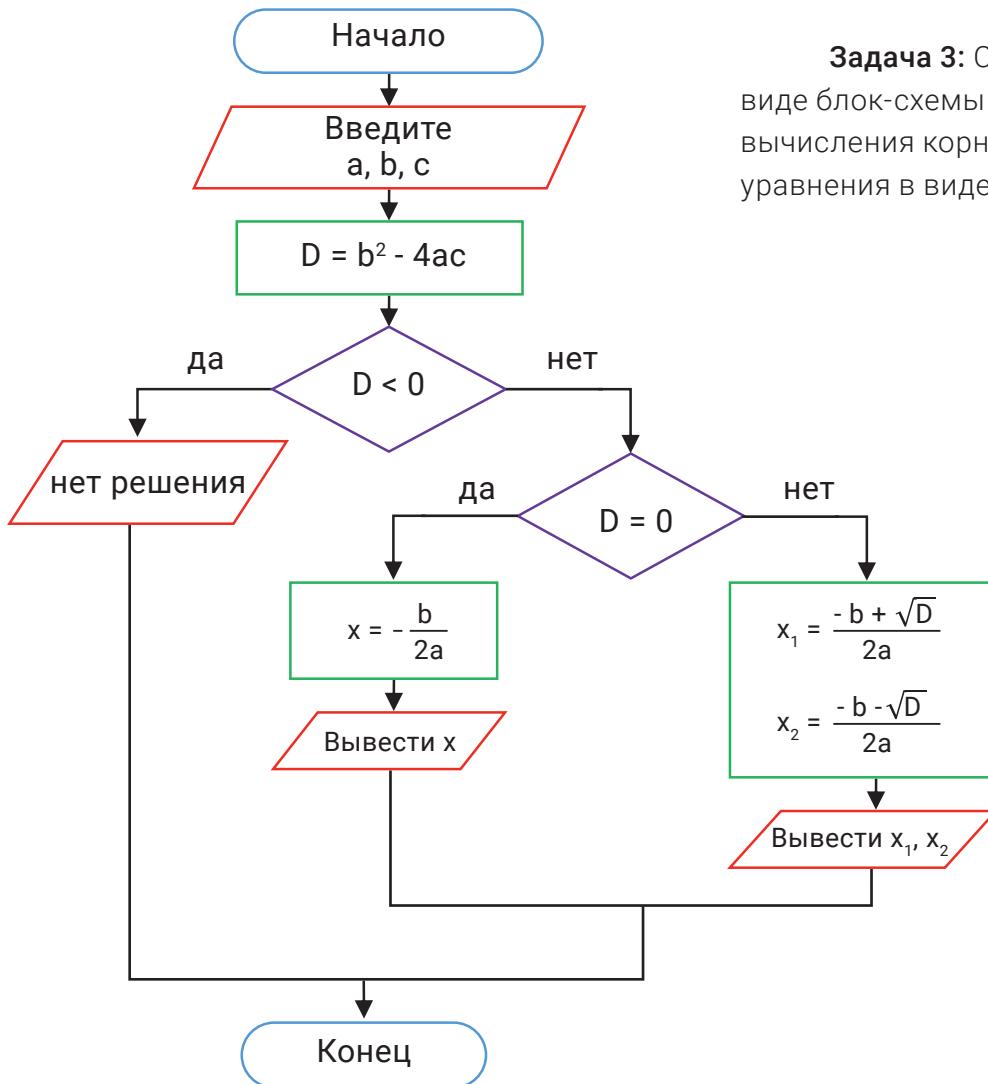
**Задача 2.** Создайте алгоритм вычисления наибольшего общего делителя (НОД) двух заданных натуральных чисел, выраженный словами и блок-схемой.

Мы уже знаем, что алгоритм Евклида – это смешанный алгоритм, и этот алгоритм для вычисления наибольшего общего делителя (НОД) двух натуральных чисел также уместен для всех натуральных чисел.

Найдите наибольший общий делитель для двух натуральных чисел А и В:

- 1) Начало;
- 2) Если  $A = B$ , то принять результат  $N$  – и перейти к пункту 6;
- 3) Определить наибольшее из чисел А и В;
- 4) Определить большее из чисел А и В, равное разнице наибольшего числа с меньшим;
- 5) перейти к пункту 2;
- 6) Конец.





**Задача 3:** Составьте в виде блок-схемы алгоритм вычисления корней квадратного уравнения в виде  $ax^2+bx+c=0$ .

### ВОПРОСЫ И ЗАДАНИЯ



1. Какие алгоритмы называются смешанными?
2. Приведите примеры из жизни со смешанными алгоритмами.
3. Где используются смешанные алгоритмы?
4. Какие алгоритмы могут входить в состав смешанных алгоритмов?

### ДОМАШНЕЕ ЗАДАНИЕ



1. Дано 8 различных чисел. Составьте алгоритм, умножающий только положительные из них.
2. В классе 18 учеников. Составьте алгоритм определения количества учеников с оценкой «хорошо» по математике.
3. Дано 12 различных чисел. Найдите сумму отрицательных и нечётных чисел.

## 25 УРОК. О ПРОГРАММЕ И ПРОГРАММИРОВАНИИ

Сегодня повсеместно идёт процесс компьютеризации ускоренными темпами. Теперь телефон – это не просто устройство для общения и разговора, но и возможность отправки текстовых, аудио, видеосообщений, а также общения через социальные сети. Наши ученики не только должны уметь пользоваться всеми этими технологиями, но и посредством программирования разрабатывать, оцифровывать и развивать их.

1. Что такое компьютерная программа?
2. Что такое программирование?
3. Считаются ли программами компьютерные приложения?



ЗНАЕТЕ ЛИ ВЫ?

Как известно, компьютер – это самый близкий помощник пользователя в решении различных задач. В частности, для удобства использования разработаны текстовые и графические редакторы, программное обеспечение для создания презентаций, электронные таблицы и многие другие приложения. Также существуют специальные компьютерные программы для образования, банковского дела, налогообложения, юриспруденции, медицины.

Так что же такое программа?

Для решения задачи на компьютере прежде всего создается ее модель и алгоритм, затем алгоритм на основе определенных правил переводится в инструкции и команды, понятные компьютеру, и записывается с использованием определенного алфавита. Созданный текст называется программой, написанной на компьютерном языке.

**Компьютерная программа** –

последовательность инструкций, которые должен выполнять компьютер для решения задачи. Компьютерная программа похожа на иностранный язык, который любой может быстро выучить.

Предположим, что почти все устройства вокруг вас управляются через компьютерные программы, то есть эти устройства подчиняются инструкциям, последовательно написанным программистом. Например, компьютерные приложения для создания документов, прослушивания песен, просмотра видео, подключения к интернету и т. д. – на самом деле компьютерные программы, написанные программистами и выполняемые компьютером.

### Основные понятия:

**Компьютерная программа** –

последовательность инструкций, предназначенных для выполнения компьютером решения какой-либо задачи.

**Программирование** – процесс создания программы для компьютера.

**Программист** – личность, создающая программу.



**Мобильные телефоны.**

Программа позволяет устройству совершать звонки и отправлять сообщения. Приложение находит номер телефона человека, сохраненный в контактах.

**Стиральная машина.**

Устройство запрограммировано для стирки в разных режимах, код контролирует температуру и продолжительность стирки.

**Автомобили.**

Без современных компьютеров и их программного обеспечения трудно сегодня представить передвижение машин. Компьютерная программа отслеживает скорость, подачу топлива и температуру.



**Программист** – это человек, создающий программы и контролирующий процессы, которые можно просматривать и выполнять на компьютере.



- 1 Что такое программа?
2. Какие программы вы знаете?
3. Что вы понимаете под определением «программирование»?
4. Работает ли стиральная машина на базе программы?
5. Перечислите устройства, работающие на базе программ.



Найдите подходящие понятия для новых терминов, приведенных в таблице, и объедините их линией.

Программа

процесс создания программы для компьютера

Программист

создает программы и контролирует процессы, которые можно просматривать и выполнять на компьютере

Программирование

«язык», понятный компьютеру

Язык  
программирования

последовательность инструкций, которые должен выполнить компьютер для решения какой-либо задачи

## 26 УРОК. ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Как и люди, компьютеры общаются на своеобразном собственном языке. Это компьютерный язык с ограниченным словарным запасом и строгими правилами правописания. «Язык», который понимает компьютер и с которым взаимодействует, называется языком программирования. Любой, кто знает какой-нибудь язык программирования, может легко создать свою собственную программу.

1. Что такое язык программирования?
2. Что такое транслятор?
3. Что такое компилятор?
4. Что такое интерпретатор?



Процессор напрямую не понимает программу, написанную на языке программирования. Для этого требуется переводчик (цифровой преобразователь), переводящий программу на язык процессора.

Есть два типа преобразователя: компилятор и интерпретатор.

**Компилятор** вначале полностью переводит код, написанный на языке программирования, в машинный код, а затем создает исполняемый файл.

**Интерпретатор** пошагово переводит, анализирует и последовательно исполняет код, написанный на языке программирования. В случае возникновения ошибки немедленно выводит сообщение.

Существует много языков программирования, и каждый используется для решения различных и своеобразных задач. Ниже мы ознакомимся с самыми популярными.

### Основные понятия:

**Язык программирования** –

официальный язык общения между человеком и компьютером. Он обрабатывает данные, основываясь на определенных правилах.

**IDE** (*Integrated Development*

*Environment* – интегрированная среда программирования) – текстовый редактор для программистов, имеющий специальные функции.

|            |  |        |   |
|------------|--|--------|---|
| C          | Язык, предназначенный для разработки операционных систем компьютера. | JAVA   | Язык программирования компьютеров, мобильных телефонов и планшетов.                           |
| JAVASCRIPT | Язык, предназначенный для разработки интерактивных web-сайтов.       | PHP    | Язык для создания динамических web-сайтов.  |
| SCRATCH    | Визуализированный язык для изучения программирования.                | PYTHON | Универсальный язык для решения различных задач и разработки систем искусственного интеллекта. |

Большинство языков программирования, как, например C++, Pascal, Java, Python и другие, имеют интегрированную среду программирования (IDE).

**IDE** (*Integrated Development Environment* – интегрированная среда разработки) – набор программных средств для создания программного обеспечения.

Языки программирования используются для создания системных и прикладных программ. Процесс создания программного обеспечения очень трудоемкий. Разработка программ на языках программирования считается частью этого процесса. В предыдущих темах мы рассматривали этапы решения задач на компьютере. Процесс создания компьютерных программ аналогичен процессу решения задач на компьютере и включает несколько шагов.

**На первом этапе** определяется потребность в программе. На этом этапе выясняются цели программы, входящие и исходящие величины. Оцениваются расходы на реализацию проекта, разработку программ и ресурсов.

**На втором этапе** разрабатывается проект программы. Формируются техническое задание и задачи. Оформляются рабочие документы и график работы.

**На третьем этапе** пишется программный код. Это процесс кодирования (программирования), созданный алгоритм переводится на язык программирования.

**На четвертом этапе** заканчивается процесс кодирования. Начинается тестирование и процесс устранения ошибок. На данном этапе оптимизируются конфигурация, производительность кодов и другие параметры программы.

**На пятом этапе** программа внедряется на практике. Если программа адаптирована к конкретным потребностям клиента, то это самый важный шаг. Архитектура и данные, использованные в предыдущей версии программы, будут адаптированы к новой программе. Необходимо обучить специалистов для работы с данной программой.

**Шестой и последний этап** – сопровождение. На данном этапе могут быть внесены изменения с учетом недостатков и спроса, возникшего в процессе работы программы.

## ВОПРОСЫ И ЗАДАНИЯ



1. Что подразумевается под языком программирования?
2. Какие виды языков программирования существуют?
3. Этапы процесса программирования.
4. Роль интерпретатора.

## ДОМАШНЕЕ ЗАДАНИЕ



1. В таблице даны этапы создания программы на компьютере. Проставьте очередность этапов.

| Процессы                          | Этапы |
|-----------------------------------|-------|
| Пишется код программы             |       |
| Поддерживается программа          |       |
| Выясняются требования к программе |       |

## Состав IDE

### Текстовые редакторы

Текстовый редактор, имеющий дополнительные функции

### Транслятор

Компилятор и/или интерпретатор, переводящий программный код в машинный

### Средство автоматизации агрегации

Агрегатор всех исполняемых и созданных компилятором файлов

### Окно исправления ошибок

Приложения для пошагового поиска, нахождения и исправления ошибок

|   |  |
|---|--|
| Программа тестируется и выявляются ошибки |  |
| Разрабатывается проект программы          |  |
| Внедряется в практику                     |  |

2. Найдите соответствия в таблице с терминами и понятиями, объедините их линией.

|            |   |
|------------|---|
| JAVASCRIPT | язык для создания операционных систем компьютера                                  |
| JAVA       | визуализированный язык программирования, удобный для изучения программирования    |
| PYTHON     | язык создания программ для компьютеров, мобильных телефонов и планшетов           |
| SCRATCH    | язык для создания динамических web-сайтов   |
| PHP        | язык для создания интерактивных web-сайтов  |
| C          | универсальный язык для решения различных задач и создания интеллектуальных систем |

## 27 УРОК. УСТАНОВКА СРЕДЫ ПРОГРАММИРОВАНИЯ PYTHON

Теперь начнем учиться программировать на языке Python. Python является одним из самых популярных языков программирования в мире для создания современных программных продуктов, которые можно использовать для создания веб-сайтов, приложений и игр. Ниже приведены дополнительные аргументы для изучения языка Python.

### Легкость обучения и применения.

Python – простой и легкий язык программирования по сравнению с другими языками. На нем очень легко начать программировать.

**Доступность совершенной библиотеки.** В процессе программирования на Python можно воспользоваться многочисленными готовыми функциями из библиотеки, которые позволяют разработать сложные приложения в короткие сроки.

**Использование данного языка программирования известными фирмами.** Из-за того, что Python – отличный язык программирования, на сегодняшний день с его помощью разрабатывают свои программы такие компании, как Google, NASA и Pixar.

### Основные понятия:

**IDLE (Integrated Development and Learning Environment** – интегрированная среда разработки) – IDE, рекомендуемый для изучения языка Python.

**Интерактивная среда** – среда ввода кода программы и получения результата без сохранения его в файле.

**Среда программирования** – среда ввода кода программы и его запуска с сохранением в виде файла.

**Оператор** – команда языка программирования.

**Интерпретатор Python можно установить бесплатно.** Можно бесплатно загрузить этот программный продукт и пользоваться им.



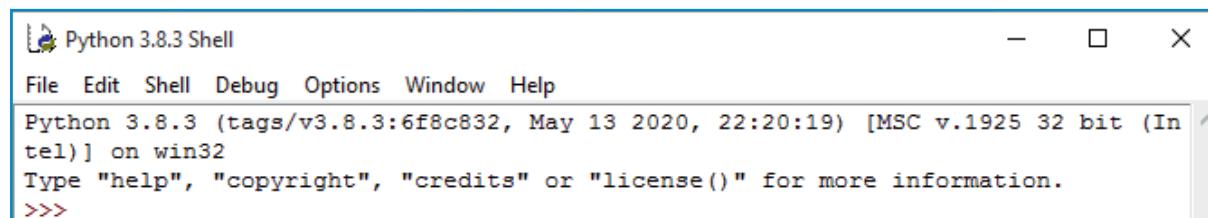
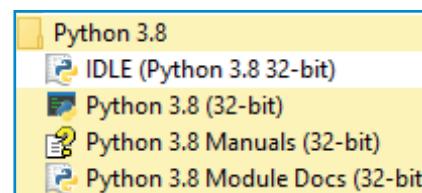
1. Как установить среду программирования Python?
2. Что такое интерактивная среда?
3. Что такое среда программирования?
4. Что такое среда IDLE?
5. Как определяются ошибки?

Для установки среды программирования Python её сначала надо загрузить с сайта разработчика. Python устанавливается на компьютер вместе с IDLE-средой. IDLE-среда предназначена для начинающих программистов и состоит из простого текстового редактора, который рассчитан для написания кода программы и окна, в котором отображаются все процессы и ошибки.

#### Установка среды программирования Python.

1. Заходим на официальный сайт Python – <http://www.python.org>. Перейдем на вкладку **Downloads**.
2. Загрузка Python. Нужно выбрать установщик для вашей операционной системы (например Windows) и загрузить последнюю версию (Python 3.8).
3. Установка. После загрузки запускаете установочный файл и в диалоговом окне выбираете **«Установить для всех пользователей»**, отвечаете на все вопросы в процессе установки и выбираете кнопку **Next**. (Не забудьте вставить галочку в пункте Add to Path – от перев.)
4. Запуск IDLE. Для проверки корректности установки в меню **«Пуск»** во вкладке **«Все программы»** открыть папку Python и запустить IDLE. В результате должно запуститься окно Python для ввода кода программы.

<https://www.python.org>



У окна **IDLE** две среды, они называются интерактивной и средой программирования.

**Интерактивная среда IDLE** (иногда называется консолью) – окно, в котором можно получить результат сразу же после ввода кода программы (не сохраняя). Эта среда удобна для изучения команд языка, составления маленьких программ, для быстрого просмотра результата кода. Интерактивную среду также можно использовать как калькулятор.

**Программная среда IDLE** – окно, в котором можно вводить код программы, редактировать и запускать его. Результат программы выводится в интерактивном окне. Эта среда используется для создания программ больших размеров, с последующим

редактированием кода. Преимущество IDLE-среды в том, что нет необходимости каждый раз повторно вводить код. Единственным недостатком данной среды является то, что программу сначала требуется сохранить и лишь затем запускать.

Давайте попробуем создать свою первую программу в среде программирования Python. Для этого мы используем оператор **print()**. С помощью **оператора print()** можно вывести значение переменной на экран.

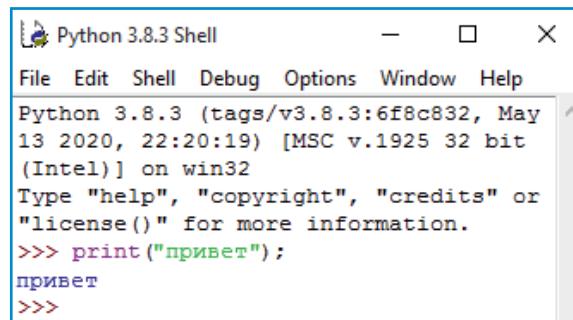
Синтаксис:  
**print** (данные для вывода)

**print** – оператор (или функция);  
**исходящая информация** – это вывод на экран переменных, констант и выражений, разделенных запятой.

### Составление программ в интерактивной среде IDLE:

1. Запускаем IDLE. С помощью этого окна можно вводить код программы, просматривать результаты и ошибки.
2. Код программы вводится после знака **>>>**. Например: `print ("Привет!")`;
3. Нажав кнопку **Enter**, можно просмотреть результат программы.

Запомните: в дальнейшем код программы в интерактивной среде (окна консоли) будем оформлять голубым цветом.

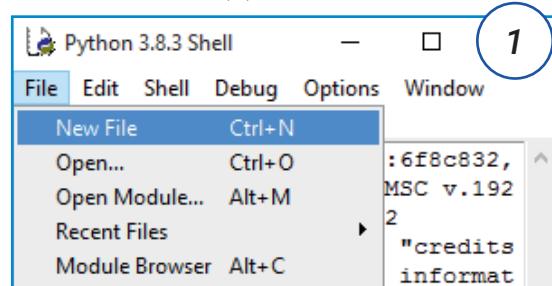


```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("привет")
привет
>>>
```

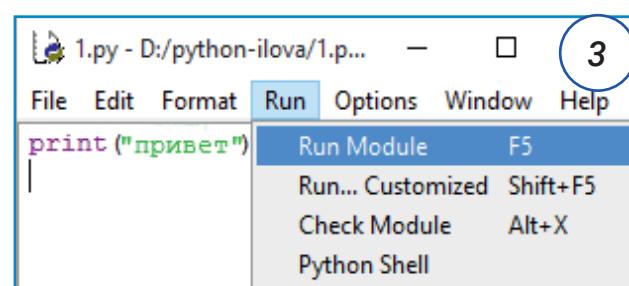
### Программирование в среде IDLE

1. Запускаем IDLE. В этом окне мы можем видеть результаты и ошибки. Программный код вводится в среду программирования.
2. Выберите «New File» в меню «File» (1).
3. В окне среды программирования введите программный код (2).
4. Выберите «Save» или «Save as...» в меню «File». Введите имя файла и сохраните, нажав на кнопку **Save** (или кнопка **Enter** на клавиатуре).
5. В окне среды программирования выберите команду «Run Module» в меню «Run» (3).
6. Мы можем увидеть результат в интерактивном окне IDLE.

Запомните: на следующих уроках программирование в среде будем выражать в розовом цвете (2):



Python 3.8.3 Shell  
File Edit Shell Debug Options Window  
New File Ctrl+N  
Open... Ctrl+O  
Open Module... Alt+M  
Recent Files  
Module Browser Alt+C



1.py - D:/python-ilova/1.py... File Edit Format Run Options Window Help  
print("привет") Run Module F5  
Run... Customized Shift+F5  
Check Module Alt+X  
Python Shell



### Запомните!

Для запуска написанной программы в среде IDLE вы должны обязательно сохранить код, а затем запустить его. Не запускайте программу без сохранения!



Ведите  
код

Сохраните  
программу

Запустите

Получите  
результат

### Запомните!

- Как и во всех других языках программирования, в Python тоже пользуются синтаксисом языка при вводе кода. Кроме этого, в Python в качестве документирования общепринятого метода ввода кода разработан PEP8 (Python Enhanced Proposal – предложения по поводу усовершенствования языка Python). На странице <http://dr.rtm.uz/pep8> вы можете ознакомиться с полным перечнем PEP8.

### ВОПРОСЫ И ЗАДАНИЯ



- С какой целью был разработан язык программирования Python?
- Как запустить язык программирования Python?
- Что такое интерактивная среда?
- Что такое среда программирования?
- В чем разница между интерактивной средой IDLE и средой программирования?
- В какой среде вводится код программы на языке программирования Python?

### ДОМАШНЕЕ ЗАДАНИЕ



- Выведите на экран строку «Моя первая программа» в интерактивной среде IDLE.
- В среде программирования IDLE составьте код «Моя первая программа», сохраните его и выведите результат на экран.
- Подготовьте небольшую проектную работу по теме «Язык программирования Python» (используйте текстовый редактор или редактор презентаций) по нижеследующему плану:  
План:
  - История языка программирования Python.
  - Краткое введение в язык программирования Python.
  - Программы и игры, написанные на языке программирования Python.
  - Функции меню среды программирования Python.

## 28 УРОК. ПЕРЕМЕННЫЕ В PYTHON

Как и во всех языках, в языках программирования тоже есть свой алфавит. Алфавит языка Python состоит из прописных и строчных букв латинского алфавита, арабских цифр, специальных символов и ключевых (служебных) слов.

### ЗНАЕТЕ ЛИ ВЫ?



- Какие буквы, цифры и символы содержит алфавит языка программирования Python?
- Что такое идентификатор?
- В чем разница между переменными и константами?
- Какие бывают типы переменных?

## Алфавит языка программирования Python

### Прописные и строчные буквы латинского алфавита

A, B, C, ..., X, Y, Z, a, b, c, ..., x, y, z

### Арабские цифры

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

### Специальные символы

Арифметические операции +, -, \*, /, кавычки, знаки препинания и др.

### Зарезервированные слова

if, for, print, input, class и др.

## Основные понятия:

**Переменные** – величины, имеющие свое значение и тип, имя ячейки в памяти компьютера, которая хранит в себе значения.

Значение переменной может меняться в течение программы.

**Константа** (постоянная) – название ячейки в памяти компьютера, хранящей в себе значения только для чтения.

Константы, как и переменные, имеют свое значение и тип.

**Идентификаторы** – общее название переменных, констант, функций, процедур, модулей и программ.

Обычно приложения предназначены для принятия введенных данных, их обрабатывания и отображения результата. При написании программ используются переменные или константы, в которых хранятся данные. Переменные идентифицируют информацию, которая может изменяться в ходе программы, а константы используются для неизменяемых данных. Для обозначения переменных и констант используются разные названия, то есть идентификаторы.

Идентификаторы состоят из различных комбинаций букв и цифр. Например, a25, b5c88 и другие. Прописные и строчные буквы не одинаковы.

## Запомните!

### Правила объявления переменных:

В имени переменной можно использовать любые буквы или цифры;

Прописные и строчные буквы различаются. label5, Label5 и LABEL5 считаются названиями различных переменных, поэтому рекомендуется всегда использовать только строчные буквы;

В названиях переменных нельзя использовать пробелы, вместо них для связывания слов между собой можно использовать нижнее подчеркивание ('\_');

Название переменной не может начинаться с цифр;

Нельзя использовать символы -, /, # или @;

Нельзя использовать названия специальных команд;

Нельзя использовать в качестве названия переменных служебные слова, типа: and, as, assert, break, class, continue, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, print, raise, return, try, while, with, yield.

## Объявление переменной.

Переменной можно присвоить строку или число с помощью знака «=». И это присвоенное значение называется значением переменной. Для присвоения переменной числового значения после знака «=» нужно писать число. А для присвоения

строкового значения после знака «=» надо ставить одинарные апострофы (' ') или двойные кавычки (" ") и внутри них писать строки для присвоения.

### Изменение значения переменной

Для изменения значения переменной достаточно присвоить новое значение.

### Использование переменных между собой

Для присвоения равнозначного значения двум переменным используется знак «=».

```
>>> age = 15  
>>> print(age)  
15  
>>> name = 'Anvar'  
>>> print(name)  
Anvar  
>>> age = 18  
>>> print(age)  
18
```

```
>>> age = 18  
>>> grad_age = age  
>>> print(age, grad_age)  
18 18
```

```
>>> age = 18  
>>> grad_age = age  
>>> age = 22  
>>> print(age, grad_age)  
22 18
```

| Название ячейки | age | grad_age |
|-----------------|-----|----------|
| Значение ячейки | 18  | 18       |

Переменной grade\_age присвоено значение переменной age. В результате обе переменных имеют одинаковое значение.

**Пример:**

```
>>> a = 6  
>>> b = a  
>>> a = 10  
>>> c = a + b  
>>> print(c)
```

16

| Название ячейки | age | grad_age |
|-----------------|-----|----------|
| Значение ячейки | 22  | 18       |

Переменной grade\_age присвоено значение переменной age. А переменной age присвоено новое значение. В результате переменная age выводит новое значение, а переменная grade\_age ранее присвоенное.

### Константы

Для неизменяемых (констант) нужно использовать только прописные буквы.

Например:

PI = 3.1415



1. Как устроен алфавит языка программирования Python?
2. Что такое переменные?
3. Разница между переменной и константой?
4. Какие символы нельзя использовать для наименования переменных?

**1. Ширина ворот 4 метра, высота – 3 метра:**

- 1) создайте программу расчета площади ворот (S);
- 2) создайте программу расчета периметра (P) кромки ворот.

**2. Две лодки в стоячей воде плывут навстречу друг другу со скоростью****4 км / ч и 2 км / ч соответственно. Если расстояние между ними равно 24 км:**

- 1) через какое время они встретятся?
- 2) Через какое время расстояние между ними составит 12 км?

**3. Радиус круга составляет 4 метра ( $\pi = 3,14$ ):**

- 1) вычислить площадь круга;
- 2) вычислить длину окружности.

## 29 УРОК. ОБРАБОТКА ОШИБОК С ПОМОЩЬЮ PYTHON

В процессе написания любой программы могут возникнуть различные ошибки. Если в написанной программе есть ошибка, программа не запустится, и на экране появится сообщение об ошибке.

1. Какие ошибки могут возникнуть в процессе программирования?
2. Можно ли исправить ошибку в программе?
3. Как исправляется ошибка в языке программирования Python?
4. Какую ошибку означает сообщение *NameError*?

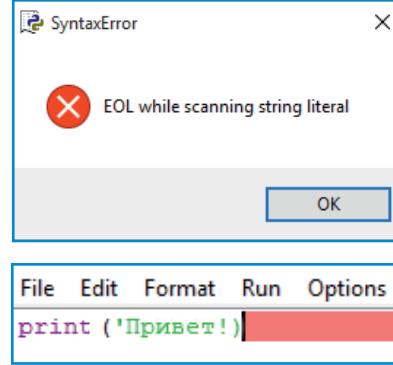
**Ошибка в среде программирования и ее исправление**

После написания программного кода в среде программирования при наличии ошибки в процессе запуска на экране появится окно сообщения об ошибке (*SyntaxError*).

Это означает, что в программном коде есть какая-то ошибка, которая мешает запуску.

1. В окне *SyntaxError* указывается обнаруженная ошибка (например, *invalid syntax* – Ошибка при вводе текста – *unexpected indent* в программе неправильно выделено место и т. д.). Нажатием кнопки **OK** можно вернуться в окно программы.

2. Тщательно проверить строку с ошибкой, которая выделена красным, и исправить.

**Ошибка в интерактивной среде и ее исправление**

Иногда вследствие допущенной ошибки появляется сообщение в красном цвете. Такие ошибки препятствуют запуску программы.

1. Ошибка *NameError* означает, что Python не распознает какое-то слово. (например, *pront* вместо *print*) (1).
2. Если такая ошибка возникает в интерактивной среде, то в рабочей области (2) щелкните правой кнопкой мыши и выберите **Go to file/line** (3).

3. В среде программирования необходимо перейти на строку с ошибкой и исправить команду (4).

### Строка с выявленной ошибкой 2

```
Traceback (most recent call last):
  File "C:/Users/User/AppData/Local/Programs/Python/Python38-32/111.py", line 1,
in <module>
    pront ('Привет!');
NameError: name 'pront' is not defined
>>> Слово, не распознанное языком Python 1
```

Слово, не распознанное языком Python 1

Cut  
Copy  
Paste  
—  
Go to file/line  
Squeeze

Перейти в среду программирования 3

```
print ('Привет!'); 4
```

## ВОПРОСЫ И ЗАДАНИЯ



1. Какие ошибки могут возникнуть в Python?
2. Как исправляются ошибки?
3. Значение `SyntaxError`.
4. Значение `NameError`.



### Запомните!



#### Частые ошибки в программировании.



**Прописные или строчные буквы.** Если в программе одновременно используются слова `print` и `Print`, Python не понимает, что это за команда.



**Одиночные и двойные кавычки.** Нельзя смешивать эти два типа кавычек. Для закрытия надо пользоваться теми же кавычками, которые использовались в качестве открывающих.



**Минус и нижнее подчеркивание.** Нельзя ни в коем случае путать символы минуса `(-)` и нижнего подчеркивания `(_)`.



**Различные скобки.** Есть несколько типов скобок: `()`, `{}` и `[]`. Открывающая и закрывающая скобки должны совпадать.

## ДОМАШНЕЕ ЗАДАНИЕ



1. Найдите ошибки во фрагментах программных кодов:

**a**

```
>>> a = 12
>>> b = 18
>>> k = a + b
>>> print(c)
```

**b**

```
>>> A = 12
>>> B = 18
>>> C = a + b
>>> print(c)
```

**d**

```
>>> age=15
>>> print("Возраст Ахмеда:")
>>> print(ag)
```

**e**

```
>>> age = 15
>>> print("Возраст Ахмеда:")
>>> print(age)
```

**f**

```
>>> age = 15
>>> grad_year = 2019
>>> print("Возраст Ахмеда:")
>>> print(age)
>>> print("Год окончания учебы:")
>>> print(grad_year)
```

## 30 УРОК. ТИПЫ ДАННЫХ В РУТНОН

Известно, что информация может передаваться в текстовой, цифровой, звуковой, графической и других формах. Для обработки такой информации в языках программирования применяется разделение на типы. Типы данных, используемые в программе, зависят от цели программы: простой калькулятор использует числа, программа для проверки адресов электронной почты работает с текстом. Числа бывают натуральными, целыми и вещественными. Текстовая же информация может состоять из символов или строк.

1. Что такое тип данных?
2. Какие типы данных вы знаете?
3. Можно ли изменить тип данных?



**ЗНАЕТЕ ЛИ ВЫ?**

Тип данных – это форма данных переменных или констант.

Определение типа данных нужно для резервирования необходимого места в памяти компьютера. Обычно в языках программирования тип данных объявляется вместе с переменной или константой. Python – это язык программирования с динамической типизацией. Следовательно, в Python тип переменной определяется используемым им значением, но для изменения типа на другой необходимо указать его.

### Основные понятия:

**Тип данных** – это форма данных, хранящихся в ячейке памяти компьютера.



| Тип данных           | Определение типа данных  | Пример                                   |
|----------------------|--|--|
| <code>int()</code>   | <b>Целые числа</b> , например, для выражения количества учеников | <code>&gt;&gt;&gt; age = 15</code>       |
| <code>float()</code> | <b>Вещественные числа</b> , например, для выражения суммы денег  | <code>&gt;&gt;&gt; prise = 20.45</code>  |
| <code>str()</code>   | <b>Строковые</b> , например, для выражения слов или предложений  | <code>&gt;&gt;&gt; name = 'Ахмед'</code> |

`bool()`

**Логические**, для выражения истинности или лжи

`>>> a= True`  
`>>> b= False`

В Python существуют и другие типы данных, но с ними мы познакомимся чуть позже.

### Изменение типа данных

Переменная может содержать информацию любого типа. Для изменения типа данных используется соответствующая команда. Данные, введенные с помощью оператора `input()`, будут отображаться в виде строки.

### Запомните!

Тип значения, принимаемый переменной, можно определить с помощью функции `type()`.

```
>>> name = 'Ахмед'  
>>> type(name)  
<class 'str'>
```

Синтаксис:  
**input** (**входящая информация**)

**input** – оператор (или функция);  
**входящая информация** – это последовательность переменных, разделенных запятой.

Для введения данных целого типа используется `int(input())`. Или вы можете изменить тип вводимых данных с помощью функции `int()`.

Например:

```
>>> age=input('Enter your age:')  
Enter your age: 15  
>>> print(age + 1)  
Traceback (most recent call last):  
  File <<pyshell#27>>,  
line 1, in <module>  
  print(age+1)  
TypeError: can only concatenate  
str (not <<int>>) to str
```

В этом примере возникла ошибка при добавлении к переменной age числа 1, потому что введенное пользователем число 15 было принято как строка.

```
>>> age=input('Enter your age:')  
Enter your age: 15  
>>> print(int(age) + 1)  
16
```

А в этом случае введенное пользователем число 15 и принятное программой как строка было преобразовано с помощью функции `int()` в число. Как следствие получаем результат 16.



1. Какой тип данных принимает значение переменной?
2. Что такое входящая информация?
3. В чем разница между переменной и константой?
4. Какие символы нельзя использовать для обозначения переменных?



1. Ширина и высота ворот вводятся пользователем:
  - 1) создайте программу вычисления площади ворот;
  - 2) создайте программу вычисления периметра кромки ворот.
2. Определите типы следующих переменных:  
 $\text{alfa} = 8764; \text{beta} = \text{'Лола'}$ .
3. Создайте программу, определяющую тип переменной во время ее ввода.

## 31 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

В зависимости от присвоенного значения переменная будет иметь соответствующие типы: целое число – int, действительное число – float, строка (одинарные или двойные кавычки) – str.

**Пример.**

```
>>> a = 12
>>> print(type(a))
<class 'int'>
>>> b = 12.3e+2
>>> print(type(b))
<class 'float'>
>>> name='clever'
>>> print(type(name))
<class 'str'>
>>> parol='@Asdf123'
>>> print(type(parol))
<class 'str'>
```

1. Определите типы следующих переменных:

$\text{alfa} = 579413; \text{beta} = \text{'clever'}; \text{d} = \text{True}; \text{s} = 0; \text{resp} = \text{'d'}; \text{b} = 100; \text{maks} = \text{False}; \text{fc} = \text{'True34'}; \text{t} = 102,5; \text{res} = \text{'2500'}; \text{a} = \text{'-50'}; \text{b} = 45,67$ .



2. Две лодки плывут в стоячей воде навстречу друг другу со скоростью  $a$  км/ч и  $b$  км/ч. Если расстояние между ними  $S$  км, через сколько времени они встретятся?  $a$  и  $b$  вводятся пользователем.

3. Высота ( $x$ ) и ширина ( $y$ ) комнаты вводятся пользователем. Напишите программу вычисления площади ( $S$ ) и периметра ( $P$ ) комнаты.

4. Длина основания треугольника ( $c$ ) и высоты ( $h$ ) вводится пользователем. Напишите программу вычисления площади треугольника ( $S$ ).

5. Средняя скорость автомобиля  $v$  (км/ч), а пройденное расстояние -  $s$  (км), вводятся пользователем. Напишите программу вычисления времени нахождения автомобиля в пути  $t$  (часов).

6. Радиус окружности равен  $r$ . Напишите программу вычисления площади круга ( $S$ ) и длины окружности ( $l$ ). Радиус круга вводится пользователем ( $\pi = 3,14$ ).

## 32 УРОК. КОНТРОЛЬНАЯ РАБОТА

- Составьте алгоритм вычисления площади круга с радиусом R произвольным способом.
- Составьте словесный алгоритм для создания графика функции  $y=x^2+3$  в MS Excel.
- Даны основание и высота треугольника. Составьте алгоритм вычисления площади треугольника.
- Составьте словесный алгоритм и блок-схему для определения совпадений с условием «Я читал эту книгу» и «К сожалению, я не читал эту книгу».
- Составьте блок-схему алгоритма, вычисляющего значение следующих функций:  
$$y = \begin{cases} x + 0.3, & \text{если } x > -6 \\ \frac{2+x}{5}, & \text{если } x \leq -6 \end{cases}$$
- Составьте алгоритм вычисления произведения четных чисел в диапазоне от -30 до 20.
- Даны 15 целых чисел. Создайте алгоритм для определения:
  - суммы отрицательных чисел;
  - количество положительных чисел;
  - количество чисел, кратных 3.
- Составьте алгоритм определения знака числа.
- С помощью слов составьте алгоритм и блок-схему для вычисления суммы чисел, находящихся в интервале между 30 и 90 и кратных 6.
- Напишите программу вычисления пройденного пути реальной точки в течение времени  $t$ , начальная скорость которого  $v_0$ , с равномерным ускорением  $a$  ( $s = v_0t + at^2/2$ ).

## 33–34 УРОКИ. ВЫПОЛНЕНИЕ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ В PYTHON

В языке программирования Python над данными числового типа можно выполнять различные арифметические операции. Если выражения составлены правильно, то можно создавать программы, выполняющие различные вычисления.

ЗНАЕТЕ ЛИ ВЫ?



- Перечислите арифметические операции.
- Существуют ли другие операции, кроме арифметических?
- Как рассчитывается остаток от деления?

## Арифметические операции

| Название операции                 | Символ операции |        | Пример           |
|-----------------------------------|-----------------|--------|------------------|
| Сложение                          | +               | x + y  | print(7+5) # 12  |
| Вычитание                         | -               | x - y  | print(7-5) # 2   |
| Умножение                         | *               | x * y  | print(7*5) # 35  |
| Деление                           | /               | x / y  | print(7/5) # 1.4 |
| Вычисление целой части от деления | //              | x // y | print(7//5) # 1  |
| Вычисление остатка от деления     | %               | x%y    | print(7%5) # 2   |
| Возведение в степень $x^y$        | **              | x**y   | print(5**2) # 25 |

Один из наиболее используемых шагов в программировании – увеличение или уменьшение переменной на указанную величину. Для выполнения таких операций используются операции (+ =) **increment** (увеличение) и (- =) **decrement** (уменьшение).

Когда в арифметических операциях рядом используется операция присваивания (=), результат равняется переменной слева.

### Краткое использование операций в программе

| Обозначение символа | Краткое оформление выражения | Полное оформление выражения | Пример     |
|---------------------|------------------------------|-----------------------------|------------|
| x = 4               | x+=y                         | x=x+y                       | x+=1 # 5   |
| -=                  | x-=y                         | x=x-y                       | x-=2 # 2   |
| *=                  | x*=y                         | x=x*y                       | x*=2 # 8   |
| /=                  | x/=y                         | x=x/y                       | x/=2 # 2   |
| //=                 | x//=y                        | x=x//y                      | x//=2 # 2  |
| %=                  | x%=y                         | x=x%y                       | x%=2 # 0   |
| **=                 | x**=y                        | x=x**y                      | x**=2 # 16 |

Выражения указывают последовательность выполнения операций. Выражения состоят из переменных, констант, скобок и операций.

| Математическое выражение               | Оформление выражения на языке программирования |
|--|--|
| $y=\frac{x^2+x-3}{x^2+5x}+\frac{1}{x}$ | y=(x**2+x-3)/(x**2+5*x)+1/x                    |

**Пример:** Дано четырехзначное число. Напишите программу, чтобы найти произведение первой и последней цифры этого числа.

```
>>> print('Enter a 4-digit number')
Enter a 4-digit number
>>> x=int(input())
4568
>>> a= x// 1000
>>> b= x % 10
>>> c= a * b
>>> print(c, '-multiply the first and last digit of the
number', x)
32-multiply the first and last digit of the number 4568
```

|                |
|----------------|
| x=4568         |
| a=4568//1000=4 |
| b=4568%10=8    |
| c=4*8=32       |



- Какие арифметические операции вы знаете?
- Как вычисляется целая часть от деления?
- Как вычисляется остаток от деления?
- Как выполняется возведение числа в степень?



1. Вычислите сумму и произведение чисел  $a$  и  $b$ . Напишите программу вычисления умножения последних цифр суммы и произведения.

| Дано    | Выражение                | Результат        |
|---------|--------------------------|------------------|
| $a = 8$ | $d = a + b = 8 + 9 = 17$ |                  |
| $b = 9$ | $c = a * b = 8 * 9 = 72$ | $S = 7 * 2 = 14$ |

- Напишите программу, вычисляющую сумму цифр двузначного числа.
- Напишите следующее выражение на Python:
  - $x + 2y + 5^2 * 4 - 58$
  - $256 + (2589 - 1549) * 458 + 456^{14} - 4565 / 5$

## 35 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Задача:** Длина каната дана в миллиметрах. Напишите программу для выражения этой длины в километрах, метрах, сантиметрах и миллиметрах.

```
>>> a=int(input())
12325458
>>> b=a//1000000 # сколько км
>>> a=a%1000000 # остаток
>>> c=a//1000 # сколько метров
>>> a=a%1000 # остаток
>>> d=a//10 # сколько см
>>> a=a%10 # остаток в мм
>>> print(b, 'км' , c, 'м', d, 'см', a, 'мм')
12 км 325 м 45 см 8 мм
```

```
a=12325458
b=12325458//1000000=12
a=12325458%1000000=325458
c=325458//1000=325
a=325458%1000=458
d=458//10=45
a=458%10=8
```



- Длина бассейна 6 метров, ширина 4 метра и глубина 3 метра. На основе данных напишите программу для решения следующих задач:
  - Сколько квадратных метров плитки понадобится для покрытия бассейна изнутри?
  - Сколько литров воды нужно для заполнения бассейна ( $1 \text{ л} = 1000 \text{ см}^3$ )?
- В приведенной выше задаче напишите программу, в которой длина, ширина и глубина бассейна вводятся пользователем.
- Создайте программу, выражающую 10 288 секунд в часах, минутах и секундах.

4. Создайте программу для вычисления произведения цифр заданного двузначного числа.

5. Создайте программу для вычисления суммы и произведения цифр заданного трехзначного числа.

| Дано | Выражение          | Результат |
|------|--------------------|-----------|
| 897  | $8+9+7$<br>$8*9*7$ | 24<br>504 |

6. Используя укороченную запись операций, выведите результат следующей программы:

| Дано                    | Выражение   | Результат |
|-------------------------|---|-----------|
| $a=8$<br>$b=5$<br>$c=9$ | $a=int(input())$<br>$b=int(input())$<br>$c=int(input())$<br>$a=b$<br>$a*=c$<br>$a+=(b*c+b)$<br>$print(a)$ |           |

7. Запишите следующее выражение на Python:  $y = \frac{7}{x^2+x+1} + x^2$

ЗНАЕТЕЛИ ВЫ?

- 1. Что такое строка?
- 2. Можно ли вырезать фрагмент строки?
- 3. Как можно определить длину строки?

### Создание строковых переменных и работа со строками

**Строка** – это последовательность букв, цифр, символов и пробела. Строки можно вводить с помощью переменных. В Python для строк используются одинарные и двойные кавычки. Одна из самых широко используемых операций над строками – это операция слияния. Для этого применяется операция +.

### Основные понятия:

**Строка** – любая последовательность в Unicode кодировке, заключенная в кавычки.

### Запомните!

Длину строки можно определить с помощью функции **len()**. Python сам посчитает количество всех символов и пробелов

```
>>> a='Весна!'
>>> len(a)
6
```

```

>>> a = 'Good morning!'
>>> b = 'Welcome.'
>>> c=a + b
>>> print(c)

Good morning! Welcome.

>>> c = a+'Dear pupil.' + b
>>> print(c)

Good morning! Dear pupil. Welcome.

```

В Python есть возможность несколько раз вывести на экран одно и то же слово, для этого достаточно ввести его всего лишь один раз.

```

>>> a = 'Hello!'
>>> print(a * 10)

Hello! Hello! Hello! Hello! Hello!
Hello! Hello! Hello! Hello! Hello!

```

### Обозначение фрагментов строк

У каждого символа в строке есть свой номер, который указывает положение символа в строке. Такая маркировка может понадобиться для определения позиции символа или для вырезания его из строки.

В Python нумерация символов в строке начинается с 0, и это число называется индексом.

|          |          |          |          |          |          |          |          |          |          |           |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|
| <b>0</b> | <b>1</b> | <b>2</b> | <b>3</b> | <b>4</b> | <b>5</b> | <b>6</b> | <b>7</b> | <b>8</b> | <b>9</b> | <b>10</b> |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|

От строк можно отрезать не только символ, но и часть строки. Для этого мы используем следующий синтаксис:

a [index] – вырезает символ, находящийся в положении **index** в строке **a**.  
 a [: end] – вырезает символы с нулевого индекса по индекс **end** от строки **a**;  
 a [start: end] – вырезает символы с индекса **start** по индекс **end** от строки **a**;  
 a [start:] – вырезает символы с индекса **start** и до конца строки **a**;  
 a [start: end: step] – вырезает символы с индекса **start** по индекс **end** от строки **a** шагом **step**.

```

>>> a = "О'ЗБЕКИСТОН"
>>> a[4]

```

выводит символ с индексом 4

```
'Е'
```

выводит последовательность с 3-го по 6 индексы

```
>>> a[3:6]
```

```
'ВЕК'
```

выводит последовательность с 0 по 6 индексы

```
>>> a[:6]
```

```
'О'ЗБЕК'
```

выводит символы с 6-го

```
>>> a[6:]
```

```
'ИСТОН'
```

и до конца строки

```
>>> a[3:10:3]
```

```
'ВІО'
```

выводит символы с 3-го по 10 с шагом 3



1. Как выполняется объединение строки?
2. Какая функция используется для определения длины строки?
3. Как вырезать подстрочку из строки?
4. Функция операции «\*» при работе со строками.



1. Составьте слово из фрагментов «Гул», «ис» и «тан»
2. Напишите программу для вывода вашего имени 5 раз на экран.
3. Класс (например, 9) и имя (например, Миршодбек) вводятся пользователем. Напишите программу для вывода строки «Я ученик 9 класса – Миршодбек».

## 37 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Специальные символы:** В Python, как и в остальных языках, существуют следующие специальные символы:

\t – знак табуляции;

\n – знак перехода на новую строку;

\' – знак одинарных кавычек;

\\" – знак двойных кавычек.

```
>>> print("Good morning.\nWelcome!")
Good Morning.
Welcome!
```

**Пример.** Можно построить простые фигуры посредством последовательности команд:

```
>>> print('*' * 12)
>>> print('*' * 4 + ' ' * 4 + '*' * 4)
>>> print('*' * 1 + ' ' * 10 + '*' * 1)
>>> print('*' * 4 + ' ' * 4 + '*' * 4)
>>> print('*' * 12)
```



1. Напишите программу, составляющую слова-подстроки из строки «ЦВЕТИК-СЕМИЦВЕТИК»
2. Напишите программу, составляющую предложение из следующих слов: «ПИШУ», «РУТНОН», «НА», «ЯЗЫКЕ», «Я», «ПРОГРАММУ».
3. Напишите программу для определения длины строки, введенной пользователем.
4. Постройте треугольник с помощью символов «\*».

```
*  
**  
***  
****
```

5. Постройте квадрат с помощью символов «+».

```
+++++  
+++++  
+++++  
+++++
```

~  
--

```
(O O)  
/ V \\\n/( - )\\  
^v ^v
```

6. Изобразите пингвиненка с помощью символов:



# 38 УРОК. ОПЕРАТОРЫ И ВЫРАЖЕНИЯ В ПУТНОН

На прошлых занятиях мы рассмотрели самые простые способы ввода данных в программу с помощью операторов `input()` и `print()` и вывода результатов на экран. Теперь познакомимся с другими возможностями и методами операторов `input()` и `print()`.

ЗНАЕТЕ ЛИ ВЫ?

1. Функция оператора `input()`.
2. Какую функцию выполняет оператор `print()`?
3. Способы ввода данных.
4. Способы вывода данных.

## Метод ввода данных

Метод `split()` может разделить данные строки, введенные оператором `input()`, на части с помощью знака-разделителя. По умолчанию, в роли разделителя выступает пробел. Если другой знак используется для разделения частей строки, то необходимо заключить его в круглые скобки `split()`.

Синтаксис:  
`split(sep)`

`split` – оператор (или функция);  
`sep` – знак, разделяющий значения.

Пример:

```
>>> a=input().split()  
book pen pencil notebook  
>>> print(a)  
['book', 'pen', 'pencil', 'notebook']
```

```
>>> a=input().split(';')  
5;8;7;1;2  
>>> print(a)  
['5', '8', '7', '1', '2']
```

## Методы вывода данных

`print()` выполняет функции вывода данных на экран или их записи в файл. С помощью методов вывода данных их можно отразить в любом виде. Для этого может использоваться полный синтаксис оператора `print()`.

`print('результат', sep=' ', end='')`

Аргумент `sep=''` используется для разделения данных результата. Существует два способа разделения данных: дефис « - » (его можно заменить и другим знаком, например «+» или «\*») и знак новой строки(`\n`).

```
>>> a='Отабек'  
>>> b='14'  
>>> c='years old'  
>>> print(a,b,c, sep='+')  
Отабек+14+years old  
>>> print(a,b,c, sep='\n')  
Отабек  
14  
years old
```

### Запомните!

`end` и `sep` являются аргументами функции `print()` и используются для изменения и управления параметрами вывода.

`end=''` определяет, каким знаком будет заканчиваться результирующая строка.

```

>>> a = 'Отабек'
>>> b = '14'
>>> c = 'years old'
>>> print(a,b,c, sep='-', end='.')
Yulduz-14-years old.

```

**Задача:** Анвару дали задание ввести в программу операцию  $478 + 874$  и вычислить её. Какую программу он должен написать?

```

>>> a=input().split('+')
478+874

```

a=['478', '874'] значение, состоящее из строк

```

>>> a1 = int(a[0])
>>> a2 = int(a[1])
>>> b = a1+a2
>>> print(b, end='; ')
1352;

```

Первому элементу присваивается число 478, а второму элементу – 874. После загружаются их переменные a1 и a2, соответственно.  
 $478 + 874$  = загружается как переменная b.  
 Вывод на экран комбинацией ";" после b/

1. Методы оператора `input()`.
2. Как используется аргумент `end = '\n'` оператора `print()`?
3. Как используется аргумент `sep` оператора `print()`?
4. Для какой цели используется метод `split()` оператора `input()`?



ВОПРОСЫ И  
ЗАДАНИЯ

1. В библиотеку привезли книги из двух книжных магазинов. Из первого магазина привезли  $n$  книг. Из второго магазина книг было на  $k$  больше, чем из первого. Сколько всего книг привезли в библиотеку? Создайте программу решения задачи. Значения  $n$  и  $k$  вводятся пользователем.

2. Камила получила задание ввести в программу и рассчитать  $854 * 89657 * 4587 * 425$ . Какую программу она должна написать?

ДОМАШНЕЕ  
ЗАДАНИЕ

## 39 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Пример:** Дано число  $k$ . Напишите программу для вычисления суммы цифр этого числа. ( $0 < k < 9999$ ).

```

n = input("Enter number: k(0<k<9999)")
n = int(n)
d1 = n % 10
d2 = n % 100 // 10
d3 = n % 1000 // 100
d4 = n // 1000
print("Result:", d1 + d2 + d3 + d4)

```

29

|   |
|---|
| '8795'  |
| 8795  |
| d1=8795%10=5<br>d2=8795%100//10=9<br>d3=8795%1000//100=7<br>d4=8795//1000=8<br>5+9+7+8=29 |

-  1. В саду дети сорвали яблоки: Анвар а, Дильшод б и Махмуд с яблок. По сколько яблок будет у каждого, если разделить собранные яблоки поровну? Сколько яблок останется? Количество яблок, которые собрал каждый ребенок, вводится пользователем. Создайте программу для решения задачи.
2. Садовник собрал со своего сада  $n$  (23856) кг урожая. Исходя из полученных данных, напишите программы для следующих заданий:
- Выразите урожай садовода в тоннах, центнерах и килограммах;
  - Если в каждый ящик помещается по 25 кг винограда, сколько потребуется ящиков?
  - В приведенной выше задаче напишите программу, где количество урожая вводится пользователем.
  - Создайте программу, где с помощью метода `split()` выполняется вычисление данной строки «5746 + 4186 + 8426 + 8266».
  - Тело получило ускорение  $a$  под действием силы  $F$ . Если значения  $F$  и  $a$  равны следующим значениям, напишите программу для расчета массы тела ( $m = F / a$ ):
    - $F = 25, a = 45;$
    - $F = 12, a = 30;$
    - $F = 72, a = 90;$
    - $F = 150, a = 15.$  - Выразите следующие выражения на Python, напишите программы для вычисления их со следующими значениями:  $a = 14, b = 8, c = 452, r = 41$
    - $S = a + b + ac;$
    - $P = \pi r^2 + ac.$

## 40 УРОК. ПРОГРАММИРОВАНИЕ ПРОСТЫХ ЗАДАЧ НА PYTHON

В процессе программирования используются три основных вида алгоритмов: линейный, разветвляющийся и повторяющийся.

Среди них линейный алгоритм широко используется для решения простых задач.



- Приведите примеры линейных алгоритмов.
- Что вы понимаете под выражением линейная программа?
- Какие операторы используются для ввода данных и значений в программу?

### Последовательность программирования простых задач:

- Определите основные данные и их типы.

Выберите названия для переменных.

- Определите, каков результат и какого он типа. Выберите названия для переменных, обозначающих результат.

- Составьте алгоритм, состоящий из ввода

### Основные понятия:

**Линейный алгоритм –**

выполнение команд в строгой последовательности.

данных, вычисления и вывода результата на экран.

4. Протестируйте алгоритм посредством введения в него различных значений.

Выполнение действий в строгой последовательности называется **линейным исполнением**. Линейный алгоритм – это последовательное выполнение действий без условий и повторений.

Выражение линейных алгоритмов в виде программы называется линейной программой.

**Задача:** Ширина класса 10 м, а его длина 12 м. Напишите программу вычисления площади класса. Ширина и длина вводятся пользователем.

| Входящие данные | Вычисление        | Выводимые данные |
|-----------------|-------------------|------------------|
| 10<br>12        | $S=a*b=10*12=120$ | 120              |

| № | Блок-схема | Название блока   | Код программы                                  |
|---|------------|------------------|--|
| 1 | Начало     | Начало алгоритма |  |
| 2 | a, b       | Блок ввода       | <code>a=int(input())<br/>b=int(input())</code> |
| 3 | $S=a*b$    | Блок вычисления  | <code>s=a*b</code>                             |
| 4 | S          | Блок вывода      | <code>print(s)</code>                          |
| 5 | Конец      | Конец алгоритма  |  |

1. Что такое линейная программа?
2. Этапы выполнения решения простых задач.
3. Основной этап в решении простых задач.



ВОПРОСЫ И  
ЗАДАНИЯ

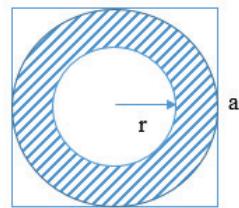
1. Даны два основания трапеции ( $a$  и  $b$ ) и высота  $h$ , проведенная к основанию. Напишите программу расчета площади  $S$  трапеции.  $a$ ,  $b$  и  $h$  вводятся пользователем.
2. Сторона равностороннего треугольника равна  $a$ . Напишите программу для вычисления его площади.
3. Сторона квадрата равна  $n$ . Создайте программу для расчета его площади.
4. Используя метод `split()`, введите в программу строку «5489 \* 245 \* 58 \* 69 \* 142 \* 4587 \* 54» и вычислите данное выражение.



ДОМАШНЕЕ  
ЗАДАНИЕ

# 41 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Задача:** Внутри квадрата нарисованы два круга. Радиус наименьшего из них равен  $r$ . Напишите программу вычисления площади заштрихованной области.



| № | Блок-схема   | Название блока   | Код программы  |
|---|--|------------------|--|
| 1 | Начало   | Начало алгоритма |  |
| 2 | a, r   | Блок ввода       | <code>a=int(input())</code><br><code>r=int(input())</code>                       |
| 3 | $r_1=a/2$<br>$S_1=\pi \cdot r_1^{**2}$<br>$S_2=\pi \cdot r^{**2}$<br>$s=S_1-S_2$ | Блок вычисления  | $r_1=a/2$<br>$S_1=\pi \cdot r_1^{**2}$<br>$S_2=\pi \cdot r^{**2}$<br>$s=S_1-S_2$ |
| 4 | s  | Блок вывода      | <code>print(s)</code>  |
| 5 | Конец  | Конец алгоритма  |  |

## ПРАКТИЧЕСКИЕ ЗАДАНИЯ:



1. Внутри квадрата со сторонами  $a$  нарисован круг. Напишите программу, вычисляющую площадь заштрихованной зоны.
2. Напишите программу для вывода на экран чисел, находящихся по обе стороны (до и после) введенного пользователем числа.
3.  $n$ -ное количество учеников собрали  $k$  яблок и поделили их поровну между собой. Остальные яблоки оставили в корзинке.
  - а) Какое количество яблок досталось каждому ученику?
  - б) Сколько яблок в корзине?
  - в) Напишите программу для варианта, когда  $n$  и  $k$  вводятся пользователем.
4. Напишите программу для определения двух последних цифр введенного пользователем числа, большего, чем двузначное.
5. Автобус за день проходит  $n$  километров. За сколько дней автобус проедет расстояние  $m$  километров?  $n$  и  $m$  вводятся пользователем. Напишите программу для решения задачи.
6. Путь, пройденный муравьем, обозначен в миллиметрах. Обозначьте его в метрах, сантиметрах и миллиметрах. (например,  $45\ 786 = 45$  м  $78$  см  $6$  мм).
7. Грань куба равна  $a$ . Напишите программу, вычисляющую объем куба.
8. Даны числа  $a$ ,  $b$ ,  $c$  и  $d$ . Напишите программу для вычисления их среднеарифметического значения.
9. Школьная администрация решила организовать кабинет математики для 3 классов. Поскольку занятия проходят одновременно, необходимо закупить парты для каждого кабинета отдельно. За одной партой не смогут сидеть больше двух учеников. Сколько парт надо купить, если известно количество учащихся в каждом классе? Пользователь вводит три значения – количество учеников в каждом классе.



# 42–43 УРОКИ. ПРОГРАММИРОВАНИЕ ЛОГИЧЕСКИХ ЗАДАЧ В РУТНОН

Для принятия решения в процессе программирования необходимо сравнение переменных, чисел и строк, и только после этого можно переходить к следующему шагу.

На этапе сравнения переменных используются операнды с логическим значением True или False.

Эти операнды в качестве результата возвращают логическое значение типа boolean: True(истина) или False(ложь).

## Основные понятия:

Операции сравнения – это операции управления логикой программы и принятие решений на основании сравнивания двух или более переменных/значений.

1. Какие операции используются для решения логических задач?
2. Знаете ли вы операции сравнения?
3. Возврат результатов операций сравнения.

### Операции сравнения:

Операции сравнения используются для сравнения двух значений. С помощью операций сравнения можно создавать простые условия.

| Операция           | Название         | Предназначение  |
|--------------------|------------------|---|
| <code>==</code>    | равно            | Возвращает True, если оба операнда равны, в противном случае False                          |
| <code>!=</code>    | не равно         | Возвращает True, если оба операнда не равны, в противном случае False                       |
| <code>&lt;</code>  | меньше           | Возвращает True, если первый operand меньше второго, в противном случае False               |
| <code>&gt;</code>  | больше           | Возвращает True, если первый operand больше второго, в противном случае False               |
| <code>&lt;=</code> | меньше или равно | Возвращает True, если первый operand меньше второго или равен ему, в противном случае False |
| <code>&gt;=</code> | больше или равно | Возвращает True, если первый operand больше второго или равен ему, в противном случае False |

ЗНАЕТЕ ЛИ ВЫ?

**Пример:** В библиотеку привезли 20 книг.

```
>>> num_book = 20
>>> num_book == 1
False
>>> num_book != 1
True
>>> num_book < 1
False
>>> num_book > 1
True
>>> num_book <= 10
False
>>> num_book >= 1
True
```

### Запомните!

Для выяснения нахождения внутри строки другой подстроки или символа используется оператор `in`.

```
>>> 'book' in 'Book shop'
True
>>> 'book' in 'Book shop'
False
```

### Логические операции

В процессе программирования для формулировки сложных выражений используются логические операции. Логические операции дают возможность управления порядком выполнения команд в программе и применяются вместе с условными и повторяющимися операторами.

| Операция         | Название             | Предназначение   |
|------------------|----------------------|--|
| <code>and</code> | Логическое умножение | Возвращает True, только если все входящие в состав сложного выражения простые выражения принимают значение True, в противном случае False            |
| <code>or</code>  | Логическое сложение  | Возвращает True, только если хотя бы одно простое выражение, входящее в состав сложного выражения, принимает значение True, в противном случае False |
| <code>not</code> | Логическое отрицание | Возвращает True, если значение выражения False, и наоборот   |

**Пример:** 15 марта – день рождения Шабнам. Эта программа с помощью логических операций проверяет дату на соответствие с днем рождения.

```
>>> day = 15
>>> month = 3
>>> day == 15 and month == 3
True
>>> day = 15
>>> month = 3
>>> not (day == 15 and month == 3)
False
```

Операция `and` проверяет оба условия на истинность.

Операция `not` заменяет результат на противоположный. Для всех дней, кроме дня рождения, результат будет `True`.

```
>>> day=15  
>>> month=3  
>>>(day==15 and month==3) or (day==1 and month==1)  
True
```

Операция **or** возвращает **True**, если хотя бы одно условие внутри скобок будет **True**.

С помощью операций сравнения также можно сравнивать и строки.

```
>>> name='Book shop'  
>>> name == 'Book shop'  
True  
  
>>> name == 'book shop'  
False  
  
>>> name == 'Book shop '  
False
```

Из-за полного совпадения строк возвращает **True**.

Из-за первой строчной буквы возвращает ложь (**False**).

Из-за знака пробела в конце строки возвращает ложь (**False**)

- Основание принятия решений в процессе программирования.
- Какие операции входят в состав операций сравнения?
- Цель использования логических операций.
- Чем отличаются простые и сложные условия?



ВОПРОСЫ И  
ЗАДАНИЯ

1. Напишите результаты следующих логических выражений:

- (3 > 5) and (2 > 4)
- (2 < 5) and (3 > 0)
- (4 > 2) or (4 < 1)
- (3 > 1) or (5 > 0)



ДОМАШНЕЕ  
ЗАДАНИЕ

2. Найдите результаты следующих логических выражений:

a)

```
>>> a = 20  
>>> b = 28  
>>> a > 17 and b = 28
```

b)

```
>>> a = 20  
>>> b = 28  
>>> c = True  
>>> a > 17 and b > 22 and c
```

## 44 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Задача:** Рассчитайте значение логического выражения:  $a=10$ ;  $b=a+4$ ;  $a+3>=b-5$

```
>>> a=10  
>>> b=a+4  
>>> a+3>=b-5  
True
```



1. Напишите программы для определения результатов логических выражений:

- 1)  $a=8; b=a^*3; a < b/3$
- 2)  $a=10; b=a^*3; a \leq b/3$
- 3)  $a=8; b=a; a+b=2^*b$
- 4)  $a=8; b=a-4; a+3>=b-2$

2. Расчитайте значения логических выражений:

- 1)  $(1 > 3) \text{ or } (4 < 0)$
- 2)  $\text{not } (5 > 6)$
- 3)  $\text{not } (6 > 5)$
- 4)  $(2 = 0) \text{ or } (2 \neq 2)$
- 5)  $(2 = 0) \text{ and } (2 > 0)$
- 6)  $(3 > 0) \text{ or } (2 > 0)$

3. Рассчитайте результаты логических выражений:

1) 

```
>>> a = 20
>>> c = False
>>> a > 17 or c
```

2) 

```
>>> a = 20
>>> c = False
>>> not a > 17 or not c
```

3) 

```
>>> a = True
>>> b = True
>>> c = False
>>> not(a and c) and (a or b) or c
```

4) 

```
>>> a = 66
>>> b = 22
>>> c = 7
>>> not((a > b) or (b < c))
```

4. Напишите программы для определения результатов логических выражений и получите результаты:

- 1)  $a \leq 5 \text{ or } a >= 0 \text{ and } a < 3$
- 2)  $x^{**}2 + y > 0 \text{ and } a = 0.1 \text{ or } (b > 3.7 \text{ and } c != 4)$
- 3)  $a < 1 \text{ or } a > 0 \text{ or not } x*x + x*x \leq 1$
- 4)  $\text{not(not(not(a > b) or True) and False)}$

## 45 УРОК. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ. ОПЕРАТОР IF...ELSE

На основании результата логического выражения принимается решение о том, какую часть программы следует выполнить. Такие задачи решаются с помощью алгоритмов ветвления.



1. Что такое алгоритм ветвления?
2. Как проверяется условие?
3. Какое значение возвращается в результате операции сравнения?

**Алгоритм ветвления** — это алгоритм, который определяет, выполняется ли последовательность команд в соответствии с условием. В алгоритме ветвления проверяется одно или несколько условий и выполняется последовательность команд на основе возврата истинного или ложного значения.

Как и во всех языках программирования, в Python тоже существуют условные операторы.

## Оператор условного перехода if

Синтаксис:  
**if** условие:  
    блок команд

### Основные понятия:

**Оператор if** – условный оператор, который выполняет определенный набор команд, только если данное условие будет истинно.

Если условие в составе оператора **if** возвращает значение **True** (истина), выполняется блок команд. Если возвращается значение **False** (ложь), блок команд выполниться не будет. По умолчанию блок команд пишется со сдвигом относительно оператора **if** на 4 пробела. Команды в блоке команд можно писать в отдельных строках или в одной, **разделяя их точкой с запятой (;)**.

| № | Блок-схема                 | Название блока        | Код программы                      |
|---|----------------------------|-----------------------|------------------------------------|
| 1 | Начало                     | Начало алгоритма      |                                    |
| 2 | age                        | Блок ввода            | age=int(input ('Enter your age?')) |
| 3 | age>18                     | Блок проверки условия | if age>18:                         |
| 4 | да<br>msg='You can enter!' | Блок выполнения       | msg='You can enter!'               |
| 5 | msg                        | Блок вывода           | print(msg)                         |
| 6 | Конец                      | Конец алгоритма       |                                    |

## Оператор условного перехода if-else

Синтаксис:  
**if** условие:  
    блок\_команд1  
**else:**  
    блок\_команд2

Вместе с оператором **if** можно использовать и команду **else**. Если условие вернет значение **True** (истина), то выполнится блок\_команд1, в противном случае выполнится блок\_команд2

**Пример:**  $y = \begin{cases} x - 3, & \text{если } x > 6 \\ x, & \text{если } x \leq 6 \end{cases}$  напишите программу решения системы уравнений.

```
x = int(input())
if x > 6:
    y = x-3
else:
    y=x
print(y)
```

8  
5



1. Оператор проверки условия.
2. Как в целом выглядит оператор условного перехода?
3. Для чего нужна проверка условия в процессе программирования?
4. Краткий и полный вид оператора ветвления.



1. Напишите программу для определения, является ли введенное число а положительным или отрицательным.
2. Напишите программу для определения, является ли введенное число а четным или нечетным.
3. Напишите программу для определения, является ли прямоугольник со сторонами а и b квадратом.
4. Напишите программу для определения, является ли введенное число а четырехзначным.

## 46 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Задача:** Напишите программу, которая сложит число 2 к введенному числу, если оно положительное, и вычтет 2, если оно отрицательное.

| № | Блок-схема | Название блока        | Код программы                    |
|---|------------|-----------------------|----------------------------------|
| 1 |            | Начало алгоритма      |                                  |
| 2 |            | Блок ввода            | <code>a=int(input ())</code>     |
| 3 |            | Блок проверки условия | <code>if a&gt;0:</code>          |
| 4 |            | Условие выполнено     | <code>    s=a+2</code>           |
| 5 |            | Условие не выполнено  | <code>else:<br/>    s=a-2</code> |
| 6 |            | Блок вывода           | <code>print(s)</code>            |
| 7 |            | Конец алгоритма       |                                  |



1. Даны два числа а и b. Напишите программу вычисления с условием, если число b меньше а, заменяет b нулем, в противном случае оставляет b без изменений.
2. Напишите программу для определения результата деления без остатка числа а на целое число b, не равное нулю.
3. Напишите программу, в которой вычисляется сумма цифр а и b, если данное целое число а делится без остатка на целое число b, не равное нулю, в противном случае вычисляется их произведение.
4. Даны три числа а, b и c. Составьте программу, которая вычисляет произведение этих чисел, если выполняется условие  $a^2 - b^2 = c^2$ , в противном случае – их сумму.
5. Дано целое число. Напишите программу вычисления, если это число имеет положительное значение, то добавить к нему цифру 1, иначе выведет его без изменений.
6. Напишите программу, которая умножит на 10 введенное число, если оно положительное; если число имеет отрицательное значение, то выведет его без изменений.

# 47 УРОК. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ. ОПЕРАТОР ELIF

В процессе решения поставленной задачи иногда приходится проверять не одно условие. Существуют такие задачи, для решения которых приходится проверять несколько условий.

1. Виды операторов условного перехода.
2. Условия работы операторов условного перехода.



## Оператор условного перехода elif

Другие языки программирования в случае, когда надо проверить несколько условий, используют оператор выбора CASE. В Python оператор CASE не используется, поэтому эта задача решается с помощью оператора **elif**.

**Elif** – это комбинированные **else** и **if** и означает «если не, то».

Синтаксис:

```
if условие:  
    блок_команд  
elif условие1:  
    блок_команд1  
...  
else:  
    блок_команд2
```

Если условие верно True (истинно), выполняется блок\_команд, иначе проверяется следующее условие – условие1. Если условие1 верно True (истинно), выполняется блок\_команд1, и так далее. А в конце, если все условия не выполняются, False (ложь), то выполняется блок\_команд2.

### Основные понятия:

**Оператор CASE** – продвинутый вариант оператора **if**, который для каждого условия выполняет свой набор команд.



**Пример:** Напишите программу простого калькулятора.

```
a=int(input('a='))  
b=int(input('b='))  
deystvie=input('add/sub/mul/div:')  
if deystvie=='add':  
    c=a+b  
elif deystvie=='sub':  
    c=a-b  
elif deystvie=='mul':  
    c=a*b  
elif deystvie=='div':  
    c=a/b  
else:  
    c='Error'  
print('Result = ', c)
```

```
a=8  
b=4  
add/sub/mul/div:add  
Result = 12  
a=72  
b=8  
add/sub/mul/div:div  
Result = 9  
a=2  
b=4  
add/sub/mul/div:deg  
Result = Error
```

## Вложенные операторы условного перехода if.

Оператор условного перехода if может содержать в себе другого оператора условного перехода if. Это называется вложенным оператором условного перехода. Внутренний if должен быть записан с отступом в один абзац (4 пробела) относительно внешнего, в противном случае будет считаться, что выражение создано отдельным оператором условия, который не является вложенным.

Синтаксис:

```
if условие:  
    блок_команд  
    if условие1  
        блок_команд1  
    .....  
else:  
    блок_команд2
```

Если условие верно True (истинно), выполняется

блок\_команд и проверяется следующее условие – условие1.

Если условие1 верно True (истинно), выполняется блок\_команд1.

Если условие не выполняется, False(ложь), то выполняется блок\_команд2.

**Пример:** Напишите программу, выводящую результат экзамена.

```
result= int(input('Введите результат (оценки: 0-5) :'))  
if result>=3:  
    print('Вы сдали экзамен!')  
if result>=5:  
    print('Высшая оценка!')
```

```
Введите результат (оценки: 0-5)  
Вы сдали экзамен!  
Высшая оценка!
```

### ВОПРОСЫ И ЗАДАНИЯ

- ?
- 1. Какой оператор используется, если в задаче проверяется только одно условие?
- 2. Как проверить несколько условий?
- 3. Будет ли программа работать корректно, если оба вложенных оператора if находятся на одном уровне?
- 4. Какие служебные команды используются в условии для последовательного исполнения команд в операторе ветвления?

### ДОМАШНЕЕ ЗАДАНИЕ

- !
- 1. Напишите программу, определяющую, являются ли цифры в двузначном числе нечетными.
- 2. Напишите программу, определяющую наличие одинаковых цифр в трехзначном числе.
- 3. Напишите программу, которая определяет, какое из чисел a и b четное.
- 4. Вводятся 3 целых числа. Напишите программу, которая определяет, какое из них четное.

## 48 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Задача:** Напишите программу для определения количества положительных и отрицательных чисел из 3 введенных.

| № | Блок-схема | Название блока        | Код программы                |
|---|------------|-----------------------|------------------------------|
| 1 |            | Начало алгоритма      | <code>a=int(input ())</code> |
| 2 |            | Блок ввода            | <code>b=int(input ())</code> |
| 3 |            | Блок проверки условия | <code>c=int(input ())</code> |
| 4 |            | Условие выполнено     | <code>k=0</code>             |
| 5 |            | Условие не выполнено  | <code>d=0</code>             |
| 6 |            | Блок вывода           | <code>if a&gt;0:</code>      |
| 7 |            | Конец алгоритма       | <code>    k+=1</code>        |
|   |            |                       | <code>else:</code>           |
|   |            |                       | <code>    d+=1</code>        |
|   |            |                       | <code>if b&gt;0:</code>      |
|   |            |                       | <code>    k+=1</code>        |
|   |            |                       | <code>else:</code>           |
|   |            |                       | <code>    d+=1</code>        |
|   |            |                       | <code>if c&gt;0:</code>      |
|   |            |                       | <code>    k+=1</code>        |
|   |            |                       | <code>else:</code>           |
|   |            |                       | <code>    d+=1</code>        |
|   |            |                       | <code>print(k,d)</code>      |
|   |            |                       |                              |

- Даны три целых числа  $a$ ,  $b$  и  $c$ . Напишите программу поиска среди них количества положительных чисел.
- Даны три целых числа  $a$ ,  $b$  и  $c$ . Напишите программу для определения количества положительных и отрицательных чисел.
- Даны два числа  $a$  и  $b$ . Напишите программу, которая печатает сначала большее, а затем меньшее из них.
- Даны три целых числа  $a$ ,  $b$  и  $c$ . Напишите программу для расчёта квадратов только положительных чисел.
- Напишите программу для решения квадратного уравнения.
- Напишите программу для вывода дня недели по введенному числу, в диапазоне от 1 до 7.
- Даны числа  $a$  и  $b$ . Напишите программу с условием, если числа положительные и сумма больше 100, вычислить деление числа  $a$  к числу  $b$ , а если они положительные и сумма не превышает 100, то вычислить произведение  $a$  и  $b$ .
- Даны два числа. Напишите программу вычисления с условием, если первое число больше второго, то выводить цифру 1, если второе число больше первого, то число 2, если они равны, то число 0.

# 49–50 УРОКИ. ПРОГРАММИРОВАНИЕ ПОВТОРЯЮЩИХСЯ АЛГОРИТМОВ. ОПЕРАТОР FOR

В решении некоторых задач приходится по несколько раз повторять одни и те же операции. Для решения таких задач используются повторяющиеся алгоритмы.

ЗНАЕТЕ ЛИ ВЫ?

-  1. Как написать код, который повторяется несколько раз?
- 2. Как работают повторяющиеся алгоритмы?
- 3. Для чего нужны циклы?

**Повторяющиеся алгоритмы** – это повторение определенной группы команд в определенное время или до тех пор, пока выполняется указанное условие. Для программирования задач с повторяющимися алгоритмами используются операторы цикла.

Например, для того чтобы определить, какие из  $n$  чисел положительные, нужно проделать одни и те же действия  $n$  раз. В таких случаях предпочтительнее использовать оператор цикла для проверки  $n$  чисел в одном блоке кода, чем писать  $n$  раз один и тот же код. Операторы цикла служат для повторяющихся команд кода.

Последовательность этих команд называется телом цикла. А каждое повторение называется **итерацией**.

## Основные понятия:

**Цикл for** – цикл со счетчиком (count-controlled). Он используется только в тех случаях, когда число повторений известно заранее.

| Название операторов цикла | Функция                                      | Пояснения  |
|---------------------------|--|--|
| <b>for</b>                | Код повторяется определенное количество раз  | Используется в случаях, когда число повторений известно заранее.   |
| <b>while</b>              | Код повторяется, пока основное условие верно | Когда число повторений неизвестно, код может не выполняться ни разу. Перед запуском кода проверяется условие. Если условие не верно, то код в цикле не запустится. |

Оба типа циклов могут использоваться для решения проблемы, но выбор правильного типа для данного условия повысит эффективность программы.

## Оператор цикла for

Синтаксис:

**for *i* in range(*start, stop, step*):**  
*sikl tanasi*

i – число повторений (итераций);  
start – начальное значение для i (по умолчанию равно 0);  
stop – конечное значение i (обязательно указать);  
step – шаг изменения (по умолчанию равен 1).

**Пример:** Напишите программу для вывода четных чисел в диапазоне от 0 до 11.

```
for i in range(0,11,2):
    print(i, end=' ')
0;2;4;6;8;10
```

Выводит числа с 0 до 11 с шагом 2.

**Пример:** Напишите программу для вывода чисел до 10.

```
for i in range(10):
    print(i, end=' ')
0;1;2;3;4;5;6;7;8;9
```

Выводит числа с 0 до 10 с шагом 1.

```
for i in range(10,0,-1):
    print(i, end=' ')
10;9;8;7;6;5;4;3;2;1
```

Выводит числа с 10 до 0 с шагом -1.

## Вложенные циклы

Использование цикла внутри цикла называется вложенными циклами.

Синтаксис:

```
for i in range(start1, stop1, step1):
    for j in range(start2, stop2, step2):
        sikił tanası
```

i – число повторений 1 цикла;  
j – число повторений 2 цикла;  
start1 – начальное значение i (по умолчанию равно 0);  
stop1 – конечное значение i (обязательно указать);  
step1 – шаг i (по умолчанию равен 1);  
start2 – начальное значение j (по умолчанию равно 0);  
stop2 – конечное значение j (обязательно указать);  
step2 – шаг j (по умолчанию равен 1).

Для каждой итерации внешнего цикла выполнится j итераций. Для i итерации внешнего цикла выполнится i\*j итераций внутреннего цикла.

**Пример:**

```
n = 3
for i in range(1,n+1):
    for j in range(1,n+1):
        print(i,'x', j,'=', i*j)

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
```

Вводится количество шагов для *i* и *j*. Цикл начинается с 1, чтобы избежать умножения на 0, поэтому количество шагов равно *n* + 1.

Результат выводится на экран. Для 1-го повторения внешнего цикла внутренний цикл выполнится 3 раза и выведет результат.

2 повторение внешнего цикла.

3 повторение внешнего цикла.



1. Какой оператор работает в качестве счетчика на языке программирования Python?
2. Каков синтаксис оператора цикла, который работает в форме счетчика?
3. Объясните функции *start*, *stop*, *step*.
4. В каких случаях нельзя использовать операторы цикла со счетчиком?
5. Будет ли работать корректно программа, если оба оператора *for* находятся на одном уровне?



1. Даны числа *a* и *b*. Напишите программу, выводящую все числа от *a* до *b*, где *a* <= *b*.
2. Даны числа *a* и *b*. Напишите программу, выводящую все числа от *a* до *b*, если *a* < *b*, в возрастающем порядке, и в обратном порядке, если *a* >= *b*.
3. Напишите программу, которая выводит квадраты натуральных чисел от 1 до 10.
4. Напишите программу, которая возвращает сумму натуральных чисел от 1 до 10.

# 51 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Пример:** Напишите программу, выводящую  $n$  чисел, кратных  $x$ .

| № | Блок-схема  | Название блока                | Код программы  |
|---|---|-------------------------------|--|
| 1 | Начало  | Начало алгоритма              |  |
| 2 | <code>x, n</code>   | Блок ввода                    | <code>x = int(input())</code><br><code>n = int(input())</code><br><code>i = 0</code> |
| 3 | <code>i=0</code>  | Количество циклов             | <code>for i in range(n):</code>  |
| 4 | <code>n</code>  | Тело цикла<br>Блок выполнения | <code>s=i*x</code>   |
| 5 | <code>s</code>  | Блок вывода                   | <code>print(s)</code>  |
| 6 | Конец   | Конец алгоритма               |  |
|   | Для выводения 6 цифр, кратных 8, вводим для $n = 6$ , а для $x = 8$ . В результате получим 0 8 16 24 32 40. |                               | <code>x=8</code><br><code>n=6</code><br><code>0 8 16 24 32 40</code>                 |

1. Определите количество итераций в следующем программном коде:

1. `for i in range(487):`  
 `cms='Привет'`

2. `for i in range(2, 89):`  
 `cms=12`

3. `for i in range(-17, 15, 4):`  
 `cms='программа'`

4. `for i in range(-58, 0):`  
 `cms=4789`

5. `for i in range(73, 161, 2):`  
 `cms=2020`

6. `for i in range(100, 1982):`  
 `cms='UZB'`

7. `for i in range(1874, 21):`  
 `cms='4782'`

8. `for i in range(-21, 21, 5):`  
 `cms='4782'`

2. Напишите программу, которая вычисляет сумму квадратов натуральных чисел от 1 до  $n$ .

3. Дано число  $n$ , отвечающее условию  $n \geq 2$ . Напишите программу для вычисления выражения:  $d = 1 * 2 + 2 * 3 + \dots + (n - 1) * n$

4. Дано несколько чисел. Напишите программу для поиска числа 2.

5. Напишите программу для вычисления суммы  $S = 11 + 13 + 15 + \dots + 49$ .

6. Напишите программу, отображающую  $n$  треугольников.  $n$  принимает число от 1 до 9

\*  
\*\*  
\*\*\*  
\*\*\*\*

ПРАКТИЧЕСКИЕ  
ЗАДАНИЯ:

## 52 УРОК. КОНТРОЛЬНАЯ РАБОТА.

1. Напишите программу, при выполнении которой выводятся два числа, находящихся до и после введенного числа. Результат программы должен выглядеть так:

| Входные данные | Выходные данные  |
|----------------|--|
| 254            | Число перед 254 – это 253<br>Число после 254 – это 255 |

2. Дано натуральное число. Напишите программу, определяющую последнюю цифру этого числа.

3. Дан отрезок времени  $n$  секунд дня. Напишите программу, которая определяет, сколько времени – час ( $h$ ), мин ( $min$ ) и секунд ( $s$ ) – прошло с начала суток.

Например,  $n = 13257 = 3 * 3600 + 40 * 60 + 57$ ;  $h = 3$  и  $min = 40$ .

4. Цена товара –  $s$  сум. Какова будет стоимость покупки  $n$ -штук этого товара?

5. Дано положительное целое число. Напишите программу для определения цифры в десятичном разряде этого числа.

6. Напишите программу, которая отображает следующее сообщение при вводе вашего имени. При создании программы используйте метод `end()` оператор `print()`.

| Входные данные | Выходные данные           |
|----------------|---------------------------|
| Бекзод         | Добро пожаловать, Бекзод! |

## 53 УРОК. ПРОГРАММИРОВАНИЕ ПОВТОРЯЮЩИХСЯ АЛГОРИТМОВ. ОПЕРАТОР WHILE

Оператор цикла `for` эффективен для решения задач, в которых число повторений заранее известно. Но количество повторений не всегда известно заранее. Лучше всего использовать оператор цикла `while`, чтобы решить, следует ли продолжать цикл или остановить, только проверив условие.

-  1. Какой цикл лучше использовать, если число повторений не известно заранее?  
2. Как работает цикл `while`?

### Оператор цикла `while`

В операторе цикла `while` выполнение цикла продолжается, пока условие в операторе принимает значение истина (`True`). Если условие прекращает принимать значение истина и примет значение ложь (`False`), цикл прекращает свою работу.

Синтаксис:

**while выражение условия:**

**тело цикла**

### Основные понятия:

- Цикл `while`** – разновидность цикла, в котором тело цикла будет выполняться, если условие истинно.  
Если в начале цикла условие не выполнится, цикл не сработает.

### Пример:

```
res='да'
while answer == 'да':
    print('Можете воспользоваться')
    res=input('Вы хотите воспользоваться
программой? (да/нет)')
    print('Пожалуйста.')
```

Можете воспользоваться  
Вы хотите воспользоваться программой? (да/нет) да  
Можете воспользоваться  
Вы хотите воспользоваться программой? (да/нет) да  
Можете воспользоваться  
Вы хотите воспользоваться программой? (да/нет) нет  
Пожалуйста.

### Запомните!

В интерактивной среде **IDLE** для остановки бесконечного цикла надо держать нажатой кнопку **Ctrl** и букву **C**. Или несколько раз нажать комбинацию клавиш **Ctrl+C**. **IDLE** отправит запрос на остановку программы.

### Пример: Найдите произведение чисел от 1 до n. $P=1*2*...*n=n!$

| № | Блок-схема   | Название блока        | Код программы  |
|---|--|-----------------------|--|
| 1 |  | Начало алгоритма      |  |
| 2 |  | Блок ввода            | <code>n=int(input ())</code><br><code>i=1</code><br><code>p=1</code> |
| 3 |  | Блок проверки условия | <code>while i&lt;=n:</code>  |
| 4 |  | Блок выполнения       | <code>P=P*i</code><br><code>i+=1</code>                              |
| 5 |  | Блок вывода           | <code>print('1*...*',n,'=',p)</code>                                 |
| 6 |  | Конец алгоритма       |  |
|   | Даем n значение 5, начальное значение i равно 1. Вычисляется произведение, поэтому начальное значение для P будет равен 1. Выведется результат $1*1*2*3*4*5=120$ . |                       | <code>n=5</code><br><code>1*...*5= 120</code>                        |

### Бесконечный цикл

Если условие в цикле **while** будет всегда выполняться (возвращать значение **True**), цикл никогда не остановится. То есть, может продолжаться вечно. Создать бесконечный цикл – очень просто, для этого достаточно вместо условия написать слово **True**.

### Пример.

```
while True:
    res=input('Введите слово:')
    print('продолжайте')
```

Введите слово книга  
продолжайте  
Введите слово:



1. Какой оператор работает в языке программирования Python как условный цикл?
2. В чем разница между оператором условного цикла и оператором цикла со счетчиком?
3. Какие еще типы операторов условного цикла вы знаете?



1. Напишите программу, выводящую четные числа в диапазоне от 0 до 20.
2. Даны  $n$  и  $k$  – целые положительные числа, где  $n > k$ . Напишите программу вычисления следующего выражения:  $\frac{n!}{k!(n-k)!}$
3. Даны натуральные числа  $a$  и  $b$ . Напишите программу, выводящую четные числа в диапазоне от  $a$  до  $b$ . Здесь  $a \leq b$ .
4. Дано натуральное число  $n$ . Напишите программу, которая выводит все числа, квадрат которых меньше  $n$ .

## 54 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Пример.** Напишите программу, вычисляющую выражение  $S=1*2+3*4+5*6+\dots+99*100$   
 $S=1*2+3*4+5*6+\dots+99*100 \Rightarrow \sum_{i=1}^{99} i*(i+1)$ ,  $i=i+2$

| №   | Блок-схема | Название блока        | Код программы           |
|---|------------|-----------------------|-------------------------|
| 1   |            | Начало алгоритма      |                         |
| 2   |            | Блок ввода            | $i=1$<br>$s=0$          |
| 3   |            | Блок проверки условия | $while i \leq 99:$      |
| 4   |            | Блок выполнения       | $s=s+i*(i+1)$<br>$i+=2$ |
| 5   |            | Блок вывода           | <code>print(s)</code>   |
| 6   |            | Конец алгоритма       |                         |
| Начальное значение $i$ равно 1. Из-за того, что вычисляется сумма, начальное значение $s$ равно 0. $0+1*2+3*4+\dots+99*100$ |            |                       | 169150                  |

1. Напишите программу для вычисления выражения  $S = 0,5 + 1,5 + 2,5 + \dots + 98,5 + 99,5$ .
2. Напишите программу, которая выводит все делители натурального числа  $n$ .
3. Напишите программу, которая определяет количество цифр введенного числа  $n$  (инструкции:  $n = n \% 10$ , выражение выполняется до  $n = 0$ ).
4. Напишите программу, которая вычисляет сумму цифр введенного числа  $n$ .
5. Напишите программу, которая считает количество четных цифр введенного числа  $n$ .
6. Дано натуральное число  $n$ . Напишите программу, выводящую числа из диапазона от 1 до  $n$ , последняя цифра которых кратна 3.

## 55 УРОК. УПРАВЛЕНИЕ ЦИКЛАМИ: ОПЕРАТОРЫ CONTINUE, BREAK

Для управления циклами используются специальные операторы **break** и **continue**. Оператор **break** используется для выхода из цикла. А оператор **continue** – для перехода в следующую итерацию.

1. Есть ли другой способ остановить бесконечный цикл, кроме  $Ctrl + C$ ?
2. Можно ли продолжить цикл после его остановки?

### Выход из цикла

Если даже условие возвращает `True`, но внутри цикла будет обращение к оператору **break**, то в таком случае цикл остановится. Любая команда внутри цикла будет отменена после обращения к оператору **break**.

### Основные понятия:

**break** – оператор, останавливающий работу цикла.

**continue** – оператор, переходящий к следующей итерации, пропуская текущую.

**Пример.** Проверим знания учеников по таблице умножения.

```
table=8
for i in range(1,11):
    print(table, 'x', i, '= ?')
    pup=input()
    res=table*i
    if int(pup)==res:
        print('Молодец!')
    else:
        print('Неверно ответ:', res)
print('Конец')
```

```
8 x 1 = ?
5
Неверно, ответ: 8
```

и вычисляет 10 циклов с 1 до 11(не включительно). Выводит вопрос из таблицы умножения.

Принимает ответ, введенный учеником. Вычисляет произведение.

Сравнивает результат с ответом, введенным учеником.

Если верно, выводит текст **Молодец**, а если нет, проинформирует о неверности ответа и выведет правильный ответ.

Для выхода из цикла добавим в код программы оператор **break**. Когда ученик вводит слово «Не знаю», цикл останавливает свою работу.

```

table=8
for i in range(1,11):
    print(table, 'x', i, '= ?')
    pup=input()
    if pup=='незнаю':
        break
    res=table*i
    if int(pup)==res:
        print('Молодец!')
    else:
        print('Неверно, ответ:', res)
print('Конец')

```

```

8 x 1 = ?
8
Молодец!
8 x 2 = ?
16
Молодец!
8 x 3 = ?
незнаю
Конец
>>>

```

Если ученик ответит «**Не знаю**», то программа выйдет из цикла и выведет текст «Конец».

### Продолжение цикла.

С помощью оператора **continue** можно, не выходя из цикла, пропустить вопрос и продолжать процесс. Если этот оператор находится внутри тела цикла, то все остальные операторы внутри тела цикла будут пропущены, и цикл продолжит свою работу со следующей итерации.

```

table=8
for i in range(1,11):
    print(table, 'x', i, '= ?')
    pup=input()
    if pup=='незнаю':
        break
    if pup=='следующий':
        print('Следующий вопрос')
        continue
    res=table*i
    if int(pup)==res:
        print('Молодец!')
    else:
        print('Неверно, ответ:', res)
print('Конец')

```

Если ученик введет ответ **«Следующий»**, программа перейдет на следующую итерацию.

```

8 x 1 = ?
8
Молодец!
8 x 2 = ?
16
Молодец!
8 x 3 = ?
следующий
Следующий вопрос
8 x 4 = ?
32
Молодец!
8 x 5 = ?

```

### Обмен переменных значениями между собой.

В языке программирования Python можно минимальными шагами обменять значения переменных между собой.

**Например,**

```
>>> a, b = 0, 1
```

```
a = 0
b = 1
```

Обычно эта операция может потребоваться для одновременного изменения значений двух переменных.

| На других языках                            | На языке программирования Python         |
|---|--|
| <pre>a=1 b=2 tt=a a=b b=tt print(a,b)</pre> | <pre>a=1 b=2 a, b=b, a print(a, b)</pre> |
| 21  | 21                                       |



ВОПРОСЫ И  
ЗАДАНИЯ

- Какие операторы управляют работой цикла?
- Какова функция оператора *break*?
- Какова функция оператора *continue*?
- Как происходит обмен значениями переменных между собой?



ДОМАШНЕЕ  
ЗАДАНИЕ

- Напишите программу простого калькулятора, состоящего из операций сложение, вычитание, умножение и деление.
- Напишите программу для вычисления суммы чисел, введенных пользователем.  
Если введено отрицательное число, цикл должен прекратить свою работу.

## 56–57 УРОКИ. ПОДПРОГРАММЫ: ФУНКЦИИ И ПРОЦЕДУРЫ

В процессе программирования может потребоваться использовать повторно какой-то фрагмент определенных операций в разных частях программы. Часть программы, содержащая этот набор действий, называется **подпрограммой**. Подпрограммы выполняют определенную функцию, но не образуют отдельную систему.

Во время обращения к подпрограмме основная программа, обратившаяся к ней, останавливает свою работу и передаёт управление подпрограмме. После завершения выполнения подпрограммы управление опять возвращается к основной программе.



ЗНАЕТЕ ЛИ ВЫ?

- Какие типы подпрограмм существуют?
- Чем функция отличается от процедуры?
- Какие преимущества дает использование подпрограмм?
- Какой оператор можно использовать для возврата результата?

Вызов подпрограмм в основной программе дает следующие возможности:

- Подпрограмма вызывается только тогда, когда в ней есть необходимость. Она устраняет необходимость писать один и тот же код по несколько раз, и им можно воспользоваться повторно. Это повышает разделение кода на блоки, облегчает понимание кода и помогает в обнаружении ошибок.

- Наличие ошибки в коде можно проверять в одном лишь блоке. Если ошибка находится в блоке, его можно устраниТЬ, всего лишь исправив одну ошибку в подпрограмме. А если не пользоваться подпрограммами, а несколько раз скопировать один и тот же код в несколько мест программы, то для устранения ошибки придется искать их по всей программе.
- Можно будет в одном месте обновлять код: все внесенные изменения начнут действовать, как только произойдет обращение к подпрограмме.

### Виды подпрограмм

- **Функция** – подпрограмма, выполняющая определенную задачу, имеющая название, принимающая одну или несколько значений, возвращающая после завершения работы в основную программу одно или несколько значений.
- **Процедура** – как и функция, подпрограмма многократного пользования, с одной лишь разницей: она не возвращает никаких значений в основную программу.

В составе языка программирования Python существуют несколько полезных стандартных функций, предназначенных для решения различных задач.

### Стандартные функции

**print()** – выводит данные для пользователя. Например, разные выводы и результаты вычислений

**input()** – обратная функция print(), передаёт данные, введенные пользователем программы.

**randint()** – выводит случайное число. Например, можно пользоваться для генерации в программе случайного числа.

Позже мы более подробно ознакомимся со стандартными функциями.

### Объявление и вызов функции

В Python каждой созданной подпрограмме, как функции, так и процедуре, обязательно требуется название, и оно вводится после ключевого слова **def** – от define (анг.define – определение).

Синтаксис:

```
def название_функции([список параметров]):  
    блок_команд
```

**def** – ключевое слово, объявляющее функцию;

**название\_функции** – название для функции;

**список параметров** – этот список может состоять из нескольких параметров, они разделяются запятой;

**блок\_команд** – тело функции должно располагаться с отступом, как и некоторые ранее рассмотренные операторы.

Когда функция вызывается называнием, команды в ее теле будут выполняться последовательно. После окончания выполнения этих команд управление возвращается в ту самую точку, откуда была вызвана эта подпрограмма, и продолжится.

## Пример. Вывод сообщения

```
def welcome():
    msg='Добрый день! '
    return msg

print(welcome())
```

Добрый день!

Объявлена функция под названием welcome. Переменной msg присвоено значение.

Задача функции – возвращение значения переменной msg .

Вызывать функцию и вывести на экран.

## Объявление процедуры

```
def msg():
    print('Сегодня в 14.00 начнется
          родительское собрание!')

>>> msg()
Сегодня в 14.00 начнется
родительское собрание!
```

## Передача значений функции

В функции для обработки можно передавать данные.

**Пример:** Напишите программу, вычисляющую длину окружности при введенном радиусе.

```
def circle(r):
    PI=3.14
    len=2*PI*r
    return len
radius = int(input('Радиус окружности: '))
uz= circle(radius)
print('Длина окружности равна: ', uz)

Радиус окружности: 4
Длина окружности равна:  25.136
```

Объявлена функция под названием circle, она принимает значение r. Вычисляется длина окружности. Функция возвращает длину окружности. Значение, введенное пользователем, переводится в целое число. Вызывается функция под названием circle. Выводится длина окружности.

## Пример. Напишите программу, вычисляющую факториал n.

S=1\*2\*3\*...\*n=n!

```
def factor(n):
    res=1
    for i in range(2,n+1):
        res*=i
    return res
n=int(input('Введите число n: '))
print(factor(n))

Введите число n: 5
```

120

Объявлена функция под названием factor. Ввели первое значение производной. Цикл выполнится с 2 до n+1, то есть 1 раз. res=1\*2\*...\*n Возвращается результат res. Вызывается функция, вычисляющая факториал n, и выводится результат на экран.

## Рекурсия

Самовызов функции называется рекурсией, и такая функция, соответственно, называется рекурсивной.

Рекурсивные функции считаются мощным инструментом программирования, но они не всегда эффективны, потому что во многих случаях допускают ошибки. Самая распространенная ошибка – это бесконечная рекурсия. В ней самовызов функции обретает бесконечный характер и продолжается, пока не заполнится все свободное пространство в памяти компьютера.

Причины возникновения бесконечной рекурсии:

- неправильно сформулированное условие. Например, при вычислении факториала, если не применить условие `if n==0`, то факториал(0) вызовет факториал(-1), а факториал(-1) вызовет факториал(-2) и т.д.
- вызов рекурсивной функции с неправильным параметром. Например, если факториал(n) будет вызывать факториал(n), опять возникнет бесконечная рекурсия.

Поэтому, прежде чем использовать рекурсивную функцию, надо подумать об условии окончания рекурсии.

```
def factor(n):  
    if n==0:  
        return 1  
    else:  
        res= n*factor(n-1)  
        return res  
n=int(input('Введите число n: '))  
print(factor(n))  
  
Введите число n: 5  
120
```

Объявлена рекурсивная функция под названием `factor`. Функция возвращает 1, если `n==0`. В противном случае продолжит работать. Функция сама вызовет себя, вычислит выражение `res=n*(n-1)*...*3*2*1` и продолжит работу, пока не достигнет условия `factor(0)`. Возвращает результат `res`. `n` присваивается значение. Вызывается функция, вычисляющая факториал `n`, и результат выводится на экран.



1. Что такое подпрограмма?
2. Для какой цели в программе используются процедуры и функции?
3. Какие виды подпрограмм вы знаете?
4. Какие преимущества даёт использование подпрограмм?
5. В чём разница функции и процедуры?
6. Когда вместо функции можно воспользоваться процедурой?



1. Даны целые положительные числа  $n$  и  $k$ ,  $n > k$ . Используя функцию, напишите программу вычисления следующего выражения:  $\frac{n!}{k!(n-k)!}$
2. Дано натуральное число  $n$ . Напишите программу, выводящую все числа, квадраты которых меньше  $n$ .
3. Одна единица длины равна '`-`'. Напишите программу, рисующую линию, состоящую из  $n$  символов '`-`'.

| Входные данные | Выходные данные               |
|----------------|-------------------------------|
| <code>n</code> | $n$ линий (' <code>-</code> ) |
| 5              | -----                         |

## 58 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Пример.** Напишите программу вычисления самого большого из трех чисел a, b и c. Используйте функцию.

```
def max(a, b):  
  
    if a > b:  
        return a  
  
    else:  
        return b  
  
def max3(a, b, c):  
  
    return max(max(a, b), c)  
  
a = int(input())  
b = int(input())  
c = int(input())  
  
print(max3(a,b,c))
```

89  
78  
58  
89

Объявлена функция **max**,  
принимающая параметры a и b

Если a > b, функция возвращает a

В противном случае функция возвращает b

Объявлена функция **max3**,  
принимающая параметры a, b и c

Функция **max3** возвращает функцию **max** с  
параметрами **max(a, b)**, c. Вначале сравниваются  
a и b, затем с наибольшим из них сравнивается  
c, и функция возвращает наибольшее из них

a, b и c присваиваются значения

Вызывается функция определения  
наибольшего из трех чисел, и результат  
функции выводится на экран.

1. Напишите программу, рисующую квадрат со сторонами n из символов \*.

ПРАКТИЧЕСКИЕ  
ЗАДАНИЯ:

| Входящие данные | Выходящие данные  |
|-----------------|---|
| n               | квадрат, состоящий из сторон,<br>равных n-му количеству (*) |
| 3               | ***<br>***<br>***   |

2. Напишите программу, которая разделяет делители данного числа n пробелом в строке. Воспользуйтесь процедурой.

| Входящие данные | Выходящие данные |
|-----------------|------------------|
| 6               | 1 2 3 6          |

3. Напишите программу, выражающую в римских цифрах введенное число n. Воспользуйтесь процедурой.

| Входящие данные | Выходящие данные |
|-----------------|------------------|
| 125             | CXXV             |

4. Напишите программу, вычисляющую сумму цифр заданного числа  $n$ . Воспользуйтесь процедурой.

| Входящие данные | Выходящие данные |
|-----------------|------------------|
| 125             | 8                |

5. В спортивном состязании выход спортсменов оценивался жюри по балльной системе. Для вычисления окончательного балла выбирались самая высшая и самая низкая оценка, из остальных баллов вычислялся средний арифметический балл. Напишите программу исключения самых высоких и самых низких показателей оценок, выставленных 5 членами жюри. Воспользуйтесь функцией.

| Входящие данные | Выходящие данные |
|-----------------|------------------|
| 1 2 3 4 5       | 15<br>3.00       |

6. Напишите программу, выводящую количество цифр заданного числа  $n$ .

| Входящие данные | Выходящие данные |
|-----------------|------------------|
| 647521          | 6                |

## 59–60 УРОКИ. ФУНКЦИИ И ПЕРЕМЕННЫЕ

Каждая переменная после объявления выполняет какое-то важное действие в программе и связывается с другими элементами программы. Переменные, объявленные внутри функции (локальные переменные), и переменные, объявленные вне функции (глобальные переменные), тоже имеют разные степени доступа и важности.

ЗНАЕТЕ ЛИ ВЫ?



1. Как используются подпрограммы в программах?
2. Как работают подпрограммы?
3. Что такое функция и как она работает?
4. Что такое процедура и как она работает?

### Локальные переменные

Локальные переменные можно использовать только внутри функции, в которой она была объявлена. В основной программе и других функциях она не работает. Поэтому во время обращения к локальной переменной вне функции появляется сообщение об ошибке.

### Основные понятия:

**Локальные переменные** видны только в локальной области видимости, в роли которой может выступать отдельно взятая функция.

**Глобальные переменные** видны во всей программе.

|   |   |
|---|---|
| <pre>&gt;&gt;&gt; def val():     a=10     print(a) &gt;&gt;&gt; val()</pre>   | <p>Объявлена функция под названием <b>val</b>. Объявлена переменная, ей присвоено значение и дана команда вывести значение <b>a</b> на экран.</p>   |
| <b>10</b>   | <p>В основной программе во время вызова функции <b>val</b> на экран выводится значение <b>a</b>.</p>  |
| <pre>&gt;&gt;&gt; print(a)  Traceback (most recent call last): File "&lt;pyshell#15&gt;", line 1, in &lt;module&gt;     print(a) NameError: name 'a' is not defined</pre> | <p>Если дана команда в основной программе вывести на экран значение <b>a</b>, выводится сообщение об ошибке. Причина этому – переменная, <b>a</b> – это локальная переменная, которая была объявлена внутри функции, и она не действительна в основной программе.</p> |

### Глобальные переменные

Глобальные переменные – это переменные, активные во всей программе. Они объявляются вне подпрограммы, то есть, в основной программе. Их обычно объявляют после импортирования модулей, в начале кода. К ним можно обращаться как к обычным переменным.

|   |   |
|---|---|
| <pre>&gt;&gt;&gt; b=5 &gt;&gt;&gt; def val2():     print(b) &gt;&gt;&gt; val2()</pre> | <p>Объявлена глобальная переменная <b>b</b> и ей дано значение 5. Объявлена функция под названием <b>val2</b>. Внутри функции дана команда вывести значение на экран.</p> |
| <b>5</b>  | <p>В основной программе, когда вызвана функция <b>val2</b>, на экран выводится значение <b>b</b>.</p>   |
| <pre>&gt;&gt;&gt; print(b)</pre>  | <p>К глобальной переменной <b>b</b> можно обращаться из любой точки программы: из основной или подпрограммы.</p>  |
| <b>5</b>  | <p>В любом из этих случаев выводится значение <b>b</b>.</p>   |

### Изменение значения глобальной переменной

Если вам нужно изменить значение глобальной переменной в функции, вы должны повторно объявить переменную в функции, используя ключевое слово `global`.

|  |   |
|--|---|
| <pre>&gt;&gt;&gt; b=5 &gt;&gt;&gt; def val3():     global b     b=b+1     print(b) &gt;&gt;&gt; val3()</pre> | <p>Объявлена глобальная переменная <b>b</b>, и ей дано значение 5. Объявлена функция <b>val3</b>. Для того чтобы изменить переменную <b>b</b> и добиться ее актуальности во всей программе, её нужно повторно объявить внутри функции в качестве глобальной переменной. Внутри функции дана команда вывести на экран значение <b>b</b>.</p> |
| <b>6</b>   | <p>После обращения внутри основной программы к функции <b>val3</b> на экран выводится значение <b>b</b>.</p>  |

&gt;&gt;&gt; print(b)

6

Когда идет обращение к глобальной переменной **b** из основной программы, в качестве результата выводится не 5, а 6, потому что она объявлена внутри функции как глобальная.

### Переменная в качестве параметра функции

Если переменная используется в качестве параметра функции, её значение будет использовано как новое значение переменной внутри функции.

```
>>> def val4(d):  
  
    print(d)  
    d=100  
  
    print(d)  
>>> c=200  
>>> val4(c)
```

200  
100

Значение **d** равно тому значению, с которым вызывается функция **val4**, то есть равно **c**.

Внутри функции дана команда вывести на экран значение **d**, которая выступает в качестве параметра.

Дано новое значение локальной переменной **d**.

Значение локальной переменной **d** выведено на экран.

Объявлена глобальная переменная.

Когда в основной программе функция **val4** вызвана с параметром **c**, сперва выводит число 200, принятое как параметр, а затем выводится новое значение переменной - 100.



1. Что такое параметры функции?
2. Какие коды находятся в теле функции?
3. Что такое локальная переменная?
4. Что такое глобальная переменная?
5. В чем разница между локальными и глобальными переменными?



1. Дано натуральное число  $n$ . С помощью процедуры напишите программу для вычисления выражения  $S = 1 * 5 + 2 * 6 + 3 * 7 + \dots + n * (n + 4)$ .
2. Даны натуральные числа  $a$  и  $b$ . Создайте функцию, находящую большее из чисел  $a$  и  $b$ .
3. Используя функцию, создайте программу для вычисления наибольшего из чисел  $a$ ,  $b$  и  $c$ .

## 61 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Пример.** Создайте программу для поиска наибольшего общего делителя (НОД) двух заданных чисел. Используйте функцию.

Самый простой способ найти наибольший общий делитель (НОД) двух заданных чисел показан в школьных учебниках, где два числа делятся на простые делители, а общий знаменатель умножается.

60=2\*2\*3\*5

Этот метод, может, и удобен в математике, но в программировании

21=3\*7

это неудобная и медленная процедура. Поэтому для решения

НОД(60,21)=3

данной задачи мы используем алгоритм Евклида.

В алгоритме Евклида используются два метода:

**1 метод:** Меньшее из двух чисел вычитается, и если разница меньше меньшего числа, они заменяются.

1)  $60 - 21 = 39$ ; 2)  $39 - 21 = 18$ ; 3)  $21 - 18 = 3$ .

$\text{НОД}(60, 21) = 3$

**2 метод:** Для уменьшения количества шагов вместо вычитания можно воспользоваться делением с остатком.

1. Большее из чисел делится на меньшее и находится остаток.

2. Они меняются местами.

3. 1 и 2 шаги повторяются до тех пор, пока одно из чисел не станет равным нулю.

4. Число, оставшееся в конце, и есть НОД двух этих чисел.

$60 / 21 = 2 * 21 + 18$  остаток

$21 / 18 = 1 * 18 + 3$  остаток

$18 / 3 = 6 * 3 + 0$  остаток

$\text{НОД}(60, 21) = 3$

```
def ekub(a, b):  
  
    while a != 0 and b != 0:  
  
        if a > b:  
            a %= b  
  
        else:  
            b %= a  
  
        ekub_q=a+b  
  
    return ekub_q  
  
a = int(input())  
b = int(input())  
print(ekub(a,b))
```

60  
21  
3

Объявлена функция, принимающая параметры a и b, под названием ekub

Команды тела цикла будут повторяться, пока a и b не будут равны 0

1 цикл. a = 60; b = 21

2 цикл. a = 18; b = 21

3 цикл. a = 18; b = 3

4 цикл. a = 0; b = 3 выход из цикла

Если число a больше b, остаток от a/b присваивается a.

1 цикл.  $60 > 21 \rightarrow a = 60 \% 21 = 18$

3 цикл.  $18 > 3 \rightarrow a = 18 \% 3 = 0$

В противном случае остаток от b/a присваивается a.

2 цикл.  $18 < 21 \rightarrow b = 21 \% 18 = 3$

ekub\_q=0+3=3 вычисляется НОД.

ekub\_q возвращает значение.

a и b даются значения.

Вызывается функция НОД двух чисел, и результат выводится на экран.

1. Напишите программу для вычисления наименьшего из четырех чисел.

Для этого создайте функцию min4.

2. Напишите программу для вычисления степени k (целое) числа a (действительное). Для этого создайте функцию stepen (a, k).

3. Данна строка из латинских букв и цифр. Напишите программу для подсчета количества символов в этой строке.

| Входящие данные              | Выходящие данные |
|------------------------------|------------------|
| jdf423h4545b5213b8u58hkj2k32 | 17               |

4. Напишите программу, зеркально отражающую заданное число. Например, 123 отражается как 321. Используйте функцию.

5. Данна последовательность чисел, заканчивающихся нулем. Создайте программу, которая вычисляет сумму чисел без использования цикла.

Например,  $1\ 7\ 9\ 0 = 17$

6. Напишите функцию для преобразования заданной единицы в тоннах, килограммах и граммах в граммы.

| Входящие данные            | Выходящие данные |
|----------------------------|------------------|
| тонна=14<br>кг=32<br>г=125 | 14032125 г       |

## 62–63 УРОКИ. БИБЛИОТЕКА ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Написание кода новой программы занимает довольно много времени. Поэтому использование готовых подпрограмм очень удобно для любого программиста. В современных языках программирования для облегчения этого процесса существует библиотека, хранящая в себе готовые коды программ.

### Основные понятия:

**Модули** – собрание кодов, записанное в отдельном файле, которым можно воспользоваться в разных программах.

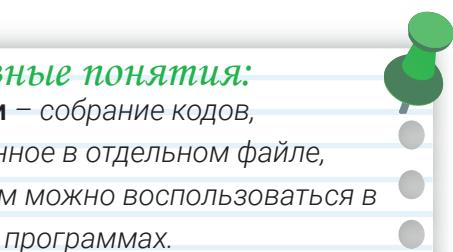
### ЗНАЕТЕ ЛИ ВЫ?



1. Как решить задачу в программировании без написания кода?
2. Функции стандартной библиотеки.
3. Что такое модули?

Как и другие языки программирования, стандартная библиотека языка программирования Python состоит из множества готовых фрагментов кода (модулей, стандартных функций и т. д.). Для дальнейшего улучшения языка программирования Python также можно выгружать пользовательские модули в отдельную часть библиотеки.

В языке программирования Python много модулей, и мы рассмотрим основные.



### Запомните!

Реплика «Batteries included» (батарейки в комплекте) означает, что в наборе языка программирования Python есть много готовых встроенных библиотек кодов.



## ГЛАВА IV. ОСНОВЫ ПРОГРАММИРОВАНИЯ

| Название модуля | Описание модуля   |
|-----------------|---|
| <b>math</b>     | используется для вычисления сложных математических выражений  |
| <b>random</b>   | генерирует случайные числа или выстраивает в случайном порядке элементы списка  |
| <b>tkinter</b>  | даёт возможность пользоваться разными графическими элементами, вроде окон или кнопок, для установки интерфейса между пользователем и программой |
| <b>datetime</b> | дает возможность вывода текущей даты и времени, вычисления в них и выполнения над ними различных операций                                       |
| <b>socket</b>   | используется для установки связи компьютера через интернет  |
| <b>turtle</b>   | используется для рисования линий и фигур на экране  |
| <b>locale</b>   | используется для решения проблем определения установленного порядка при форматировании чисел  |
| <b>decimal</b>  | используется для работы с десятичными дробями и округлениями  |
| <b>os</b>       | дает возможность работы с папками и файлами   |
| <b>copy</b>     | предназначена для решения задач, связанных с копированием   |
| <b>sys</b>      | считается средой выполнения программы в интерпретаторе Python   |

Чтобы использовать модули в программе, необходима установка. Это требует обращения к фрагменту кода, хранящемуся в нем. Есть три разных способа загрузки модулей в программу.

**Способ 1.** В этом методе загрузки функций модуля имя модуля должно быть указано перед фрагментом кода, на который вы ссылаетесь. Такие программы очень легко читать, потому что вы можете быстро определить, к какому модулю принадлежит код.

Синтаксис:

**import** название модуля

**import** – ключевое слово для загрузки модуля.

```
>>> import random
>>> random.randint(1, 5)
```

Загрузили функции модуля `random` из стандартной библиотеки.

3

Перед каждой функцией указывается название модуля

**Способ 2.** Этот метод загрузки функций модуля полезен для небольших приложений. В больших программах программу трудно понять, а это значит, что требуется определенное усилие, чтобы определить, к какому модулю принадлежит функция.

Синтаксис:

**from** название модуля **import**\*

**from ... import**\* – ключевое слово для загрузки модуля.

```
>>> from random import *
>>> randint(1,5)
```

4

Загрузили функции модуля **random** из стандартной библиотеки.

Не указывается принадлежность функции к модулю

**3 способ.** Из модуля можно загрузить только сами функции. Если из всего модуля нам требуется лишь одна функция, то удобнее вместо всех загрузить только необходимые из них.

Синтаксис:

**from** название модуля **import** название функции

```
>>> from random import randint
>>> randint(1,5)
```

3

Загрузили функцию **randint** модуля **random** из стандартной библиотеки.

Не указывается принадлежность функции к модулю

Если у вас возникнут вопросы о функциях, выполняемых модулями, обратитесь в справочную систему библиотеки Python. Чтобы не терять время в процессе программирования, важно изучить и запомнить стандартную библиотеку, её модули и функции, уметь пользоваться готовыми решениями. Для запуска справочной системы надо выбрать пункт **Python Docs** в меню **Help**.

Модуль **random** управляет генерацией случайных чисел.

## функции модуль random

| Функции                             | Описание   |
|-------------------------------------|--|
| <b>random(x)</b>                    | Генерирует случайные числа от 0 до 1                               |
| <b>randint(start, stop)</b>         | Генерирует случайные числа в интервале от start до stop            |
| <b>randrange(start, stop, step)</b> | Генерирует случайные числа в интервале от start до stop шагом step |

### ВОПРОСЫ И ЗАДАНИЯ



1. Что такое библиотека языков программирования?
2. Что такое модуль и для чего он используется?
3. Какие модули вы знаете?
4. Как называется модуль, выполняющий математические вычисления?

### ДОМАШНЕЕ ЗАДАНИЕ



1. Напишите программу для извлечения 10 случайных чисел в диапазоне от 0 до 1.
2. Напишите программу для вывода 5-ти случайных чисел в диапазоне от 10 до 10 000.
3. Напишите программу для вывода 7 случайных чисел с шагом 2 в диапазоне от 20 до 50.

## 64 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

В составе модуля math библиотеки Python находятся функции, выполняющие математические, тригонометрические и логарифмические операции.

| Функции                   | Описание   |
|---------------------------|--|
| <code>ceil(x)</code>      | Округляет до самого близкого большего целого числа. <code>Ceil(1.5)==2, ceil(-1.5)==-1</code>    |
| <code>round(x, n)</code>  | Округляет число x до n символов после точки  |
| <code>floor(x)</code>     | Округляет до самого близкого меньшего целого числа. <code>floor(1.5)==1, floor(-1.5)== -2</code> |
| <code>round(x)</code>     | Округляет число x  |
| <code>log(a, b)</code>    | Вычисляет логарифм на основе b   |
| <code>log10(x)</code>     | Вычисляет десятичный логарифм числа x  |
| <code>sqrt(x)</code>      | Вычисляет квадратный корень x  |
| <code>pow(x, n)</code>    | Вычисляет n-ную степень x  |
| <code>factorial(x)</code> | Вычисляет факториал x  |
| <code>abs(x)</code>       | Вычисляет модуль x   |
| <code>cos(x)</code>       | Вычисляет косинус x  |
| <code>sin(x)</code>       | Вычисляет синус x  |
| <code>tan(x)</code>       | Вычисляет тангенс x  |
| <code>acos(x)</code>      | Вычисляет арккосинус x   |
| <code>asin(x)</code>      | Вычисляет арксинус x   |
| <code>atan(x)</code>      | Вычисляет арктангенс x   |
| <code>degrees(x)</code>   | Переводит радианты в градусы   |
| <code>radians(x)</code>   | Переводит градусы в радианты   |

**Задача.** Напишите программу вычисления площади равностороннего треугольника со стороной a. a вводится пользователем.

$$s = \sqrt{\frac{3}{4}a^2}$$

```
from math import *
a = int(input())
s = sqrt(3/4)*pow(a, 2)
print(s)
```

4  
13.856406460551018

1. Напишите программу вычисления длины заданной угловой дуги. Угол наклона дуги (в градусах) и радиус вводятся пользователем.
2. Напишите программу вычисления значения функции  $y = x * \cos x$ . x вводится пользователем.
3. Напишите программу вычисления корней квадратного уравнения. a, b, c вводятся пользователем.
4. Округлите дробную часть данного действительного числа с точностью от 1 до 4.  
Данное число: 0.26598  
Результат:  
Точность 1: 0,3      Точность 2: 0,27  
Точность 3: 0,266      Точность 4: 0,2660
5. Напишите программу расчета площади сектора круга. Радиус окружности и угол сектора (в градусах) вводятся пользователем.

ПРАКТИЧЕСКИЕ  
ЗАДАНИЯ:

# 65–66 УРОКИ. РАБОТА С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ ПОЛЬЗОВАТЕЛЯ В PYTHON

Большинство языков программирования используют элементы управления: окна, текстовые поля и кнопки для взаимодействия с пользователем. Все вместе они называются графическим пользовательским интерфейсом (GUI).

## Основные понятия:

**Widget** (виджеты) – элементы интерфейса вроде кнопок или текстовых полей, используемые для создания приложения с GUI.

### ЗНАЕТЕ ЛИ ВЫ?

1. Что такое графический интерфейс пользователя?
2. Как создавать приложения с графическим интерфейсом?

Окно, в котором расположены все элементы, является основой графического интерфейса. Для создания окон и его элементов (виджетов) используется модуль Tkinter из стандартной библиотеки Python.

**Tkinter** – это стандартная графическая библиотека в Python. При установке Python библиотека поставляется вместе с программой. После установки Python у вас появится возможность создания отличных приложений с графическим интерфейсом, используя необходимые объекты и методы для создания. GUI для создания приложений требуется:

- импортировать модуль Tkinter;
- создать главное окно Tkinter;
- добавить в приложение один или несколько виджетов;
- войти в основной цикл с доступом к базовому коду.

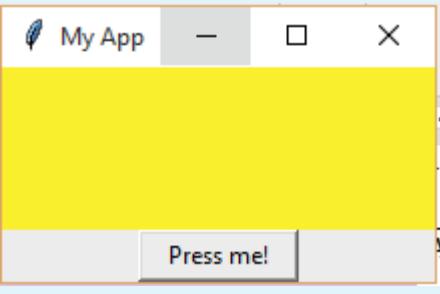
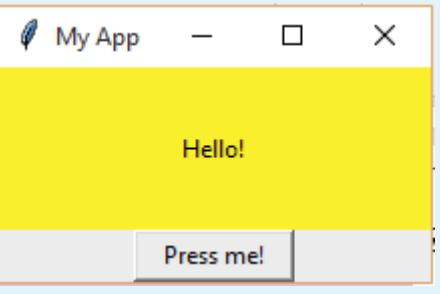
```
from tkinter import *
window = Tk()
window.title('My App')
window.geometry('250x50')
window.configure(background='yellow')

my_label=Label(window, width=40,
height=5, bg='yellow', text='')

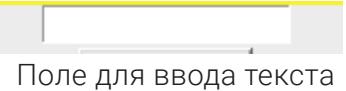
my_label.grid(row=0, column=0)

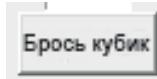
def change_text():
    my_label.config(text='Hello!')
```

|  |
|--|
| Загрузить модуль Tkinter из библиотеки   |
| Создание окна Tkinter  |
| Дать название заголовку окна Tkinter   |
| Размер окна Tkinter  |
| Цвет фона окна Tkinter   |
| С помощью функции Label устанавливается пустой текстовый виджет с шириной 40 и высотой 5 |
| Размещение текстового поля в ячейку 0-строки 0-столбца                                   |
| Объявление функции, которая стартует при нажатии на кнопку                               |

|  |   |
|--|---|
| <pre>my_button=Button(window, text="Press me!", width=10, command=change_text) my_button.grid(row=1, column=0)</pre>   | <p>При помощи функции Button устанавливается кнопка под названием <b>Press me!</b> с шириной 10. При помощи атрибута command= указывается функция, исполняемая при нажатии кнопки</p>   |
| <pre>window.mainloop()</pre>  <p>При запуске программы выводится следующее окно</p> |  <p>Установление кнопки в ячейку, 1-строка 0-столбец</p> <p>Вход в основной цикл</p> <p>В пустом текстовом поле появляется надпись Hello!</p> |

Все виджеты, используемые в приложениях программы, вводятся между командами `window = Tk()` и `window.mainloop()`. Метод `grid()` в модуле Tkinter позволяет, используя ячеичную систему координат, расположить виджеты в нужные координаты.

| Виджеты             | Вид в программе  | Результат   |
|---------------------|--|---|
| <i>Label()</i>      | <pre>my_label=Label(window, width=40, height=5, bg='yellow', text='Hello!') my_label.grid(row=0, column=0)</pre>   |  <p>Текстовое поле</p>                       |
| <i>Text()</i>       | <pre>my_text_box=Text(window, width=4, height=2) my_text_box.grid(row=0, column=1)</pre>   |  <p>Текстовое поле для вывода результата</p> |
| <i>Entry()</i>      | <pre>my_text_box=Entry(window, width=20) my_text_box.grid(row=0, column=0)</pre>   |  <p>Поле для ввода текста</p>                 |
| <i>OptionMenu()</i> | <pre>options=(1,2,3) my_variable_object=IntVar() my_variable_object.set('Выберите:') my_dropdown=OptionMenu(window, my_variable_object, *options) my_dropdown.grid()</pre> |  <p>Список для выбора</p>                    |

|                      |   |   |
|----------------------|---|---|
| <b>Radiobutton()</b> | jinsi=StringVar()<br>radio1=Radiobutton(window, text='Ayol', variable=jinsi, value='ayol')<br>radio1.grid(row=3, column=0, sticky=W)<br>radio1.select()<br>radio2=Radiobutton(window, text='Erkak', variable=jinsi, value='erkak')<br>radio2.grid(row=3, column=1, sticky=W)<br>radio2.select() |    |
| <b>Checkbutton()</b> | var1=IntVar()<br>checkbox1=Checkbutton(window, text='Python', variable=var1)<br>checkbox1.grid(row=0, column=0)<br><br>var2=IntVar()<br>checkbox2=Checkbutton(window, text='Java', variable=var2)<br>checkbox2.grid(row=1, column=0)  |    |
| <b>Button()</b>      | my_button=Button(window, text='Кубик от!', command=kubik)<br>my_button.grid(row=4, column=0)  |   |
| <b>PhotoImage()</b>  | foto=PhotoImage(file='image/foto.png')<br>my_label=Label(window, image=foto)<br>my_label.grid(row=0, column=0)  |  |

## ВОПРОСЫ И ЗАДАНИЯ



- Что такое графический интерфейс пользователя (GUI)?
- Виджет и его назначение.
- Как добавить виджет в приложение с графическим интерфейсом.
- Методы размещения виджета в окне приложения.
- Вызов функции с помощью кнопки.

## ДОМАШНЕЕ ЗАДАНИЕ

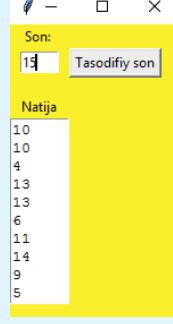


- Напишите код программы, создающий окно GUI, размером 100x100, под названием «Моё первое приложение». Установите в окно кнопку, выводящую сообщение «Привет, Узбекистан!».
- Напишите код программы, создающий окно GUI, размером 250x150, под названием «Фрукты». Расположите виджет, позволяющий выбрать один из 4 фруктов.

## 67 УРОК. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

**Пример.** Используйте GUI для создания программы вывода 10 случайных чисел от 1 до n. N вводится пользователем.

|  |  |
|--|--|
| from tkinter import *  | Загружается модуль Tkinter из библиотеки Python                                  |
| from random import randint                                   | Загружает из библиотеки функцию randint модуля random                            |
| def tasodifiy():   | Объявляется функция  |
| number = int(textbox_input.get())                            | Значение переменной textbox_input, введенной через текстовое поле                |
| textbox_output.delete(0.0, END)                              | Удаляет значение переменной textbox_output, то есть поле вывода результата       |
| for i in range (1,11):                                       | Работают 10 циклов с 1 по 11. Нахождение случайного числа [1:number]             |
| t_son = str (randint(1, number)) + '\n'                      | Сопоставляет случайное число и поле вывода результата, то есть с textbox_output. |
| textbox_output.insert(END, t_son)                            | Создание окна Tkinter  |
| window = Tk()  | Название окна Tkinter  |
| window.title('Tasodifiy son')                                | Установление размера окна Tkinter  |
| window.geometry('250x250')                                   | Устанавливает фоновый цвет окна  |
| window.configure(background='yellow')                        | Создает ярлык для указания верхней границы случайного числа                      |
| input_label = Label (window, text='Son: ', bg='yellow')      | Устанавливает текстовое поле в ячейку: 0-строки и 0-столбца                      |
| input_label.grid (row=0, column=0)                           | Создает ярлык для указания названия поля вывода случайных чисел.                 |
| output_label = Label(window, text = '\nNatija', bg='yellow') | Устанавливает в ячейку текстовое поле: 2-строка и 0-столбец.                     |
| output_label.grid(row=2, column=0)                           | Создает текстовое поле для ввода верхней границы случайного числа                |
| textbox_input = Entry (window, width=5)                      | Устанавливает в ячейку текстовое поле: 1-строка и 0-столбец.                     |
| textbox_input.grid (row=1, column=0)                         | Создает текстовое поле для вывода 10 случайных чисел.                            |
| textbox_output = Text(window, height=10, width=6)            |  |

|   |  |
|---|--|
| <pre>textbox_output.<br/>grid(row=3, column=0)<br/><br/>kubik_button = Button(window, text='Tasodify son', command=tasodify)<br/><br/>kubik_button.grid(row=1,<br/>                  column=1)<br/><br/>window.mainloop()</pre> | <p>Устанавливает в ячейку текстовое поле 3-строку и 0-столбец</p> <p>Создает кнопку «Случайное число»,зывающую функцию tasodify</p> <p>Устанавливает кнопку в ячейку.<br/>1 строка и 1 столбец</p> <p>Вход в основной цикл</p> |
|  <p>При запуске программы выводится следующее окно</p>   |  <p>При нажатии на кнопку выводятся случайные числа в диапазоне от 1 до 15</p>   |



- Используя графический интерфейс, создайте программу, которая возвращает 10 чисел, кратных заданному числу. Заданное число вводится пользователем.
- Используя графический интерфейс, создайте программу, которая выводит числа, кратные заданному числу. Заданное число и количество чисел вводится пользователем.
- Используя графический интерфейс, создайте программу, которая вычисляет  $a^b$ , принимая два числа  $a$  и  $b$ .
- Используя графический интерфейс, создайте программу, которая производит п простых чисел (). Простым числом называется число, делящееся на 1 и самого себя. п вводится пользователем.
- Используя графический интерфейс пользователя, создайте программу, которая принимает два числа  $a$  и  $b$  и вычисляет их НОД.
- Используя графический интерфейс пользователя, создайте программу, которая принимает два числа  $a$  и  $b$  и вычисляет их НОК.

## 68 УРОК. КОНТРОЛЬНАЯ РАБОТА

- Напишите программу вычисления суммы кубов натуральных чисел от 1 до n.
- Напишите программу расчета выражения  $P = 2 * 4 * 6 * \dots * 40$ .
- Напишите программу отражения данных последовательных чисел, которые заканчиваются на 0.

Например, поменяйте 1230 на 0321.

4. Создайте программу с помощью процедуры замены в приведенном тексте символа «k» символом «q», символа «t» символом «d», символа «n» на символ «m».

5. Создайте программу, которая вычисляет НОД двух или более чисел.

| Входящие данные | Выходящие данные |
|-----------------|------------------|
| 18 24           | 72               |

6. Напишите функцию перевода величины, данной в часах, минутах и секундах, в секунды.

| Входящие данные                       | Выходящие данные |
|---------------------------------------|------------------|
| часы = 4<br>минут = 15<br>секунд = 60 | 15360 с          |

7. Создайте открытку с текстом и изображением в поле Canvas.

8. Создайте калькулятор с помощью графического интерфейса.

9. Улитка ползёт к вершине дерева высотой  $h$  метров. Днем она поднимается на  $a$  метров. А ночью спускается на  $b$  метров. Сколько дней нужно, чтобы достичь вершины дерева?

Значения  $h$ ,  $a$  и  $b$  ( $a > b$ ) вводятся пользователем.

10. Даны два числа  $a$  и  $b$ . Создайте программу для выявления большего из них.

11. Напишите программу, выводящую пингвина, созданного с использованием символов  $n$  раз.  $n$  принимает натуральные числа от 1 до 4.



12. Напишите программу, выводящую на экран  $n$  квадратов.  $n$  принимает числа от 1 до 5.

+++++  
+++++  
+++++  
+++++

## **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ВЕБ-САЙТОВ**

1. Boltayev B., Azamatov A., Asqarov A., Sodiqov M., Azamatova G. Informatika va hisoblash texnikasi asoslari. Umumiy o'rta ta'lim mакtablarining 9-sinfи uchun darslik. Toshkent: "Cho'lpon" nomidagi NMIU, 2015. – 160 b.
2. Chris Roffey. Computer science. Programming book for Python. – USA: Cambridge university press. 2017, – p. 204
3. Chris Roffey. Python basics. Coding club. Level 1,2. – USA: Cambridge university press. 2012, – p. 85
4. Eric Matthes. Python crash course: a hands-on, project-based introduction to programming. – San-Francisco: No Starch Press, 2015. – p. 562
5. Matt Harrison. Illustrated guide to Python 3. 2017, – p. 267
6. Dan Bader. Python tricks the book. Anja Pircher Design, 2017, – p. 299
7. Jamie Chan. Learn python in one day and learn it well. – p. 132
8. Jake VanderPlas. A whirlwind tour of python. – USA: O'Reilly Media. 2016. – p. 98
9. Carol Vorderman. Computer Coding for Kids: A unique step-by-step visual guide, from binary code to building games. London: Dorling Kindersley Ltd, 2014, – p. 224.
10. Robert Sedgewick and Kevin Wayne. Algorithms. Fourth edition. Princeton University. First printing, March 2011.
11. Садыгов И. Дж., Махмудзаде Р. А., Исаева Н. Р. Информатика-11. Учебник для общеобразовательных школ. «Bakıñəşr» – Баку, 2011, 128 стр.
12. Алексеев Е. Г., Богатырев С. Д. Информатика. Мультимедийный электронный учебник. <http://inf.e-alekseev.ru/text/index.html>
13. Васильев А. Н. Python на примерах. – Санкт-Петербург: Наука и техника, 2018, – 430 с.
14. Джейсон Бриггс. Python для детей. Самоучитель по программированию / пер. с англ. Станислава Ломакина; [науч. ред. Д. Абрамова]. – М.: Манн, Иванов и Фербер, 2017. – 320 с.
15. Кадиркулов Р. А., Рыскулбекова А. Информатика, 7 класс.  
<https://www.opiq.kz/Kit/Details/61>
16. <https://www.w3resource.com/python/>
17. <https://younglinux.info/python/task/>
18. <https://pythonru.com/>
19. <https://python-scripts.com/>
20. <https://www.rupython.com/>
21. <https://informatics.msk.ru/>
22. <https://pythonworld.ru/>
23. <http://pythoshka.ru/>

Fayziyeva Maxbuba Raximjonovna,  
Sayfurov Dadajon Muxammedovich,  
Xaytullayeva Nafisa Saxobiddinovna

*O'quv nashri*

## **INFORMATIKA VA AXBOROT TEHNOLOGIYALARI**

Umumiy o'rta ta'lif maktablarining 9-sinfi uchun darslik

(Rus tilida)

**Главный редактор**

З.Сардарян

**Дизайнеры**

С.Дониёров, К.Шадрин

**Технический редактор**

С.Серенков

**Верстка**

К.Мельникова

**Перевод с узбекского**

У.Б.Маматкулов, М.М.Алимова,  
Д.М.Сайфуров

«Nashriyot uyi Tasvir»

Ташкент – 2020

Лицензия издательства AI №292, 23.02.2017

Подписано в печать 14 января 2021 года. Формат 60x84 1/8.

Бумага книжная легкомелованная.

Кегль 11. Гарнитура «Roboto, Agency FB, a\_EmpiricalNr» Офсетная печать.

Усл. печ.л.13,02. Уч.-изд..л.12,63. Тираж 68 188. Заказ № 3265.

Опечатано в типографии «Kolorpak» .

г.Ташкент, ул. Элбек, 8



**COLORPACK**

Таблица состояния арендованного учебника

| № | Фамилия, имя ученика | Учебный год | Состояние учебника при получении | Подпись классного руководителя | Состояние учебника при сдаче | Подпись классного руководителя |
|---|----------------------|-------------|----------------------------------|--------------------------------|------------------------------|--------------------------------|
| 1 |                      |             |                                  |                                |                              |                                |
| 2 |                      |             |                                  |                                |                              |                                |
| 3 |                      |             |                                  |                                |                              |                                |
| 4 |                      |             |                                  |                                |                              |                                |
| 5 |                      |             |                                  |                                |                              |                                |
| 6 |                      |             |                                  |                                |                              |                                |

При сдаче учебника классный руководитель оценивает его состояние по показателям, заполняя таблицу:

|                    |  |
|--------------------|--|
| Новое              | Состояние учебника, полученного в первый раз.  |
| Хорошее            | Обложка в хорошем состоянии, переплет целый.   |
| Удовлетворительное | Все страницы в наличии, не порваны и не исписаны.  |
| Плохое             | Обложка немного повреждена, переплет книги нарушен, уголки страниц загнуты. Некоторые страницы исписаны. Оторванные страницы заново приклеены. |