Котов В.М., Макарова Н.П., Лапо А.И., Войтехович Е.Н.

Информатика

Учебное пособие для 6 класса учреждений образования, реализующих образовательные программы общего среднего образования с русским (белорусским) языком обучения

Общее количество страниц _____
Общее количество иллюстраций _____
Дата «____» _____20____г.
Подпись(и) автора(ов)_____
_____
_____

Минск
2023

ГЛАВА 6. АЛГОРИТМЫ И ИСПОЛНИТЕЛИ

§16. *Понятие алгоритма и исполнителя*

16.1. Понятие алгоритма

В повседневной жизни нам приходится решать много задач, простых и сложных: сбор в школу, покупка мороженого, звонок по мобильному телефону. Более сложно получить отметку 10 по информатике, победить на соревнованиях, вырастить дома цветок шафрана и др.

Для решения любой задачи необходимо выполнить несколько действий (пример 16.1).

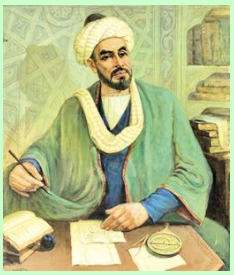
Алгоритм – понятная и конечная последовательность точных действий (команд), формальное выполнение которых позволяет получить решение поставленной задачи.

Команда в алгоритме – указание на выполнение конкретного действия.

Для решения одной и той же задачи могут использоваться разные алгоритмы. Например, для написания поздравительной открытки один учащийся может использовать бумагу и цветные карандаши, другой – текстовый редактор, третий – графический редактор (пример 16.2).

В III в. до н. э. древнегреческий математик Евклид изложил правило вычисления наибольшего общего делителя двух натуральных чисел. Это правило считают первым алгоритмом.

Эвклид



Аль-Хорезми

Термин *алгоритм* (лат. *algorithmus*) произошел от имени арабского математика Мухаммеда аль-Хорезми (787 – 850.). Он разработал правила выполнения четырех арифметических действий, применяемых и сегодня.

Пример 16.1. Подключение к сети Интернет через Wi-Fi в общественном месте:

1. Включить Wi-Fi на мобильном устройстве.
2. Выполнить поиск доступных сетей.
3. Выбрать доступную сеть.
4. Если сеть имеет ключ доступа, уточнить его у администратора.
5. Ввести ключ доступа.

Данный алгоритм состоит из 5 команд.

16.2. Понятие исполнителя алгоритма

Каждый алгоритм создается человеком или группой людей. Алгоритм выполняется

Пример 16.2 Подготовка поздравительной открытки:

1. Открыть графический редактор

исполнителями алгоритмов.

Исполнитель алгоритма – человек (группа людей) или техническое устройство, которые понимают команды алгоритма и умеют правильно их выполнять.

Исполнителем алгоритмов из примеров 16.1 и 16.2 может быть человек. Выполнять алгоритм может робот, детская игрушка, экшн-камера, автопилот и т. д. (пример 16.3).

Команды, которые понимает и может выполнить исполнитель, образуют **систему команд исполнителя.** В примере 16.4 приведена система команд исполнителя Робот-пылесос. В зависимости от площади и особенностей помещения человек может задать разные режимы работы (алгоритмы) робота-пылесоса.

Алгоритмы, предназначенные для выполнения на компьютере, записывают на языке программирования. Запись алгоритма на языке программирования называют **программой**. Исполнителем программ является компьютер.

Компьютерный исполнитель – виртуальный объект, действующий в виртуальной среде (пример 16.5).

Для некоторых исполнителей требуется определенная обстановка. Такую обстановку называют **средой обитания исполнителя** (пример 16.6).

Исполнитель *Шестиклассник* (среда Paint.

2. Нарисовать открытку.
3. Распечатать открытку.

Данный алгоритм состоит из трех команд.

Пример 16.3. Примеры исполнителей.

Пример 16.4. Система команд исполнителя *Робот-пылесос*:

- зонировать помещение;
- регулировать подачу воды;
- распознать предметы;
- выполнить уборку;
- выполнить самоочистку.

Пример 16.5. Примеры компьютерных исполнителей:

- *Чертежник, Робот, Черепаха* реализованы для различных языков программирования
- Рыжий кот из программы Scratch.

Пример 16.6. Средой обитания исполнителя алгоритма 16.1 может быть

обитания – 6-й класс) умеет:

- задумывать натуральное число,
- выполнять арифметические действия над числами,
- записывать числа,
- находить наибольшее и наименьшее число среди заданных чисел.

Ему предлагается выполнить алгоритм:

1. Задумать натуральное число.
2. Умножить задуманное число на 2.
3. К произведению прибавить 10.
4. Результат разделить на 2.
5. От частного отнять задуманное число.
6. Записать результат.

(Рассмотрите пример 16.7).

Пусть среда обитания исполнителя *Кисть* – лист бумаги. Система команд исполнителя *Кисть*:

- стрелка – исполнитель рисует отрезок некоторой длины в направлении, указанном стрелкой;
- зачеркнутая стрелка – исполнитель движется в направлении, указанном стрелкой, не оставляя следа (пример 16.8).

Рассмотрим алгоритм определения суточной амплитуды температуры воздуха (пример 16.9):

1. Определить максимальную температуру воздуха за сутки.
2. Определить минимальную температуру воздуха за сутки.
3. Найти разность между максимальным и

только среда, в которой используются мобильные телефоны. Алгоритм невозможно выполнить при отсутствии сети и точки доступа к сети Интернет.

Алгоритм из примера 16.2 нельзя выполнить, не имея компьютера и принтера.

Пример 16.7. Выполнение алгоритма исполнителем *Шестиклассник*:

Номер команды	Результат выполнения команды
1	105
2	105 × 2 = 210
3	210 + 10 = 220
4	220 : 2 = 110
5	110 – 105 = 5
6	5

Пример 16.8. Выполнение алгоритма для исполнителя *Кисть*.

Алгоритм:

Выполняя команды алгоритма, получим изображение:

Пример 16.9. Определение суточной амплитуды температуры воздуха исполнителем *Шестиклассник* по таблице.

Время наблюдения	Температура, °С
6.00	+10

минимальным значением температур.

Исполнителем этого алгоритма может оказаться любой человек, которому понятны команды алгоритма.

Алгоритм определения азимута на местности может быть таким (пример 16.10):

1. Совместить темный конец магнитной стрелки компаса с направлением на север.
2. Мысленно провести прямую линию от центра компаса к объекту.
3. Определить угол между стрелкой на север и мысленной линией к объекту по направлению часовой стрелки (азимут на север равен 0°).

Из курса математики вам известен алгоритм построения прямоугольной системы координат:

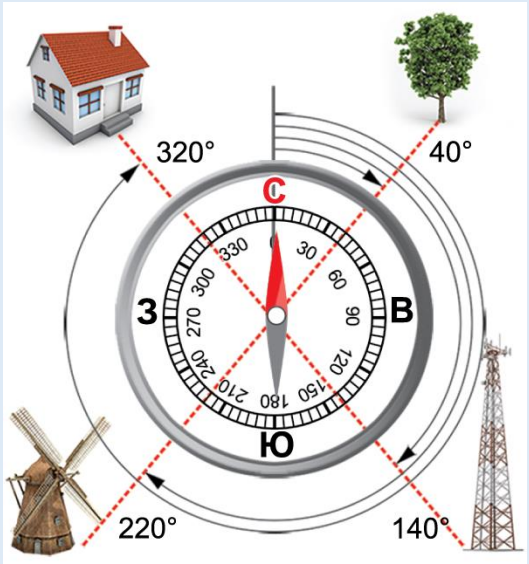
1. Построить две перпендикулярные прямые и обозначить OX и OY.
2. Выбрать положительное направление и отметить его стрелкой на каждой прямой.
3. Отметить начало координат: точку O (число 0).

Отложить единичный отрезок в положительном направлении на каждой оси.

12.00	+17
18.00	+14
24.00	+8

Максимальный результат +17 °C, минимальный – +8 °C. Амплитуда температур воздуха: 17 °C – 8 °C = 9 °C. Результат выполнения алгоритма: 9.

Пример 16.10. Определение азимута для объектов на рисунке:



Поручим выполнение алгоритма исполнителю *Шестиклассник* в предположении, что он понимает и может правильно выполнить команды алгоритма. Результат выполнения алгоритма: азимут на колодец равен 45°; азимут на мельницу равен 130°; азимут на вышку равен 300°.

1. Что такое алгоритм?
2. Что называется командой в алгоритме?
3. Что такое исполнитель алгоритма?
4. Кто может быть исполнителем алгоритма?

5. Что называют системой команд исполнителя?
6. Что такое среда обитания исполнителя?
7. Что называется программой?

Упражнения

1. Приведите примеры алгоритмов из повседневной жизни и учебной деятельности.

2. Какие из следующих процессов можно описать в виде алгоритмов:
   1. Замена колеса в автомобиле.
   2. Написание домашнего сочинения.
   3. Сложение двух дробей.
   4. Забивание гола на футбольном матче.
   5. Получение изображения белорусского орнамента, показанного на рисунке.
   6. Запись ряда всех натуральных чисел.

3. *Решите подбором задачу Аль-Хорезми: «Я к трети числа прибавил единицу и к четверти числа прибавил единицу. Перемножив эти числа, получил 20. Какое число я взял»?

4. Приведите примеры исполнителей алгоритмов.

5. Напишите систему команд одного из исполнителей примера 16.3.

6. Выполните алгоритм из примера 16.7 несколько раз для разных чисел. Сравните полученные результаты. Сделайте выводы.

7. По командам  исполнитель Кисть рисует часть окружности в указанном направлении. Определите результат выполнения алгоритма:

8. Напишите алгоритм морфологического разбора имени прилагательного в предложении. Выполните этот алгоритм для прилагательного «цифровом» из фразы «Современный человек живет в цифровом мире».

9. *Придумайте исполнителя алгоритмов со своей системой команд и напишите для него алгоритм решения некоторой задачи.

§17. *Способы записи алгоритмов*

Издавна человек стремился записывать нужные действия в краткой и понятной форме. Так в различных сферах жизни появились разнообразные инструкции: правила игры, кулинарные рецепты, методы решения математических задач, схемы вязания и т.д.

Многие из таких записей можно считать алгоритмами, так как они записаны в виде точных и понятных команд и приводят к решению задач.

Существуют следующие способы записи алгоритмов:

- словесное описание;
- графический (блок-схема);
- программный.

Словесный способ записи алгоритма – запись алгоритма на естественном языке общения.

(Рассмотрите примеры 17.1 и 17.2, в которых представлено словесное описание алгоритмов).

Графический способ записи алгоритма – запись алгоритма с помощью геометрических фигур (блоков), соответствующих командам алгоритма, и линий для соединения блоков.

В информатике для графического способа

Пример 17.1. Алгоритм игры на детском правовом сайте[1]:

1. Задумать любое двузначное число.
2. Вычесть из него составляющие его цифры.
3. Найти число разности и его символ в таблице.

Вообразить мысленно этот значок и крутить волшебное колесо.

Пример 17.2. Алгоритм приготовления белорусских драников:

1. Очистить картофель.
2. Натереть картофель на мелкой терке.
3. Натереть луковицу на мелкой терке.
4. Добавить лук в картофель.
5. Добавить яйцо, муку, соль и специи.
6. Хорошо все размешать.
7. Разогреть сковороду с растительным маслом.
8. Выложить ложкой картофельную массу на сковороду в виде лепешки.

[1] http://mir.pravo.by (дата доступа 17.01.2023).

записи алгоритма используются блок-схемы, в которых каждый блок изображается в виде некоторой геометрической фигуры и имеет свое назначение.

Блоки начала и окончания алгоритма:

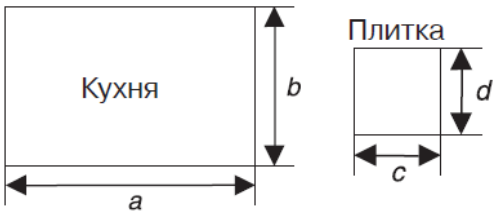
Блок для записи выполняемых команд алгоритма:

Блоки для ввода исходных данных и вывода полученных результатов:

Запись алгоритма в виде программы называется **программным способом записи алгоритма**.

Записывать алгоритмы программным способом вы научитесь на следующих уроках.

Рассмотрим такой пример. Пусть в квартире планируется проведение ремонта. Предполагается покрыть пол на кухне кафельной плиткой. Необходимо написать алгоритм определения минимального количества плиток, необходимых для ремонта.



Словесное описание и блок-схема алгоритма, позволяющего определить необходимое количество плиток для ремонта,

9. Обжарить с двух сторон.

Пример 17.3. Ремонт на кухне.

Словесное описание алгоритма:

1. С помощью рулетки измерить размеры кухни (длину a, ширину b)
2. Вычислить площадь кухни

$$S_{кухни} = ab$$

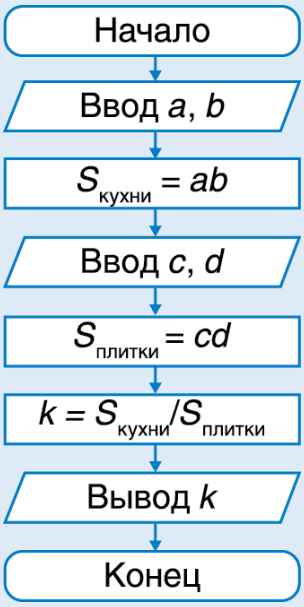
3. С помощью рулетки определить размеры одной кафельной плитки (длину c, ширину d).
4. Вычислить площадь плитки

$$S_{плитки} = cd.$$

5. Определить минимальное количество плиток $k = \frac{S_{кухни}}{S_{плитки}}$.

Результатом выполнения алгоритма является значение k.

Запись алгоритма определения количества плиток для ремонта кухни в виде блок-схемы:



Пример 17.4. Алгоритм копирования текстового фрагмента в другую часть текстового файла.

Графический способ записи

представлены в примере 17.3.

С помощью рулетки определим размеры кухни и плитки и выполним алгоритм.

1. Размеры кухни: $a = 4{,}4$ м, $b = 3{,}2$ м.

2. $S_{\text{кухни}} = 4{,}4 \cdot 3{,}2 = 14{,}08$ м^2.

3. Размеры плитки:

$c = 0{,}33$ м, $d = 0{,}33$ м.

4. $S_{\text{плитки}} = 0{,}33 \cdot 0{,}33 = 0{,}1089$ м^2.

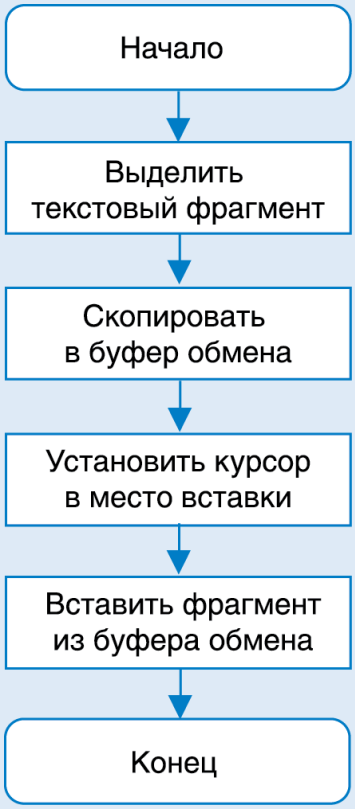
5. $k = 14{,}08 : 0{,}1089 \approx 129{,}30$ плиток.

Ответ: минимальное количество плиток для ремонта кухни – 130.

В примере 17.4 показана блок-схема копирования текстового фрагмента из одной части документа в другую. В алгоритме предполагается, что исполнитель понимает и умеет правильно выполнять все команды. В противном случае, необходимо более подробное описание отдельных команд (например, как скопировать выделенный текст в буфер обмена).

В примере 17.5 приведено словесное описание алгоритма определения особенностей периода каменного века.

алгоритма:



Пример 17.5. Алгоритм определения особенностей периодов каменного века.

1. Словесное описание:
2. Выбрать период каменного века.
3. Указать время его существования.
4. Определить основные орудия труда.
5. Указать способы добычи пропитания человеком.
6. Указать наиболее важные события и явления периода.

1. Какие способы записи алгоритмов вам известны?
2. Какой способ записи называют словесным?
3. Что такое «графическая запись алгоритма»?
4. Перечислите блоки (геометрические фигуры), которые используются для записи алгоритмов.
5. Какой способ записи алгоритма называют программным?

Упражнения

1. Запишите алгоритм перемещения текстового фрагмента из одной части текстового файла в другую.

2. Приведите графическую запись алгоритмов решения примеров 17.1 и 17.2.

3. Составьте алгоритм для выполнения синтаксического разбора простого предложения.

4. В курсе истории вы ознакомились с этапами работы над историческими источниками и документами. Запишите их в виде алгоритма.

5. Запишите для исполнителя ***Шестиклассник*** алгоритм сложения дробей $\frac{a}{b}$ и $\frac{c}{d}$. Выполните алгоритм для дробей $\frac{13}{27}$ и $\frac{12}{27}$.

6. Участок земли прямоугольной формы имеет длину a м и ширину b м. Запишите алгоритм определения площади участка и длины забора, который потребуется для ограждения участка. Исполните алгоритм на примере некоторого земельного участка.

7. В курсе биологии вы познакомились со строением растительной клетки. Запишите алгоритм, позволяющий определить строение клеток кожицы лука.

8. Составьте алгоритм действий пешехода, если красный сигнал светофора застал его на проезжей части.

9. *Запишите алгоритм, позволяющий определить толщину листа бумаги учебного пособия «Информатика, 6».

10. *Запишите алгоритм решения старинной задачи: «Требуется переправить на другой берег трех рыцарей и их оруженосцев. Имеется лодка, которая может вместить только двух человек. Известно, что ни один оруженосец не может находиться в обществе других рыцарей без своего рыцаря».

11. *Имеются кувшин емкостью 8 л, заполненный квасом, и два пустых кувшина емкостью 3 л и 5 л. Запишите алгоритм, выполняя который можно разделить квас поровну между двумя людьми (разрешается пользоваться только этими тремя кувшинами).

§18. *Среда программирования и компьютерный исполнитель*

18.1. Среда программирования

Для разработки программ используются среды программирования (интегрированные среды разработки, ИСР, англ. *IDE, Integrated Development Environment*).

Среда программирования – это комплекс программ, используемых при разработке программного обеспечения.

Среда программирования может быть рассчитана на работу с одним или несколькими языками программирования. Для работы с каким-либо языком программирования могут использоваться разные среды программирования.

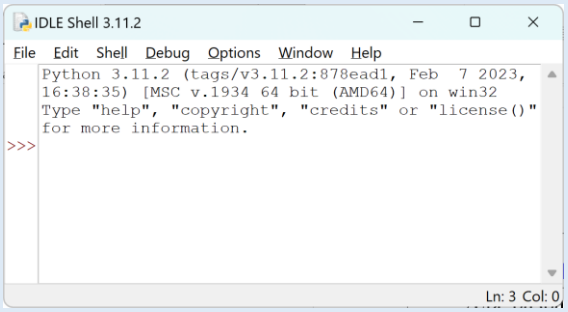
Для работы я языком программирования Python можно использовать среды программирования:

- среда IDLE устанавливается вместе языком Python[2] (пример 18.1);
- среда PyCharm[3] (пример 18.2);
- онлайновые среды[4] программирования (пример 18.3).

Для изучения основ работы с программами на языке программирования Python будем использовать среду программирования IDLE. Запустить среду можно с помощью значка или

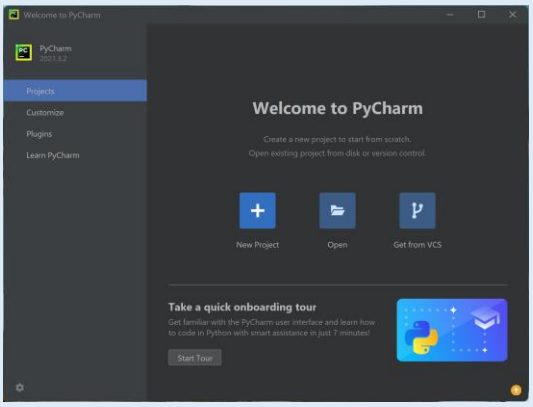
Язык **Python** появился в 1991 году. Разработчиком языка является Гвидо ван Россум, голландский программист.

Пример 18.1. Среда программирования IDLE:



Пример 18.2. Среда программирования PyCharm

Значок среды программирования IDLE



Пример 18.3. Онлайн среда программирования для исполнителя Черепаха.

[2] https://www.python.org/downloads/
[3] https://www.jetbrains.com/ru-ru/pycharm/download/other.html. Выбрать из раздела PyCharm Community Edition.
[4] Например, https://metaschool.ru/pub/konkurs/python/turtle/index.php

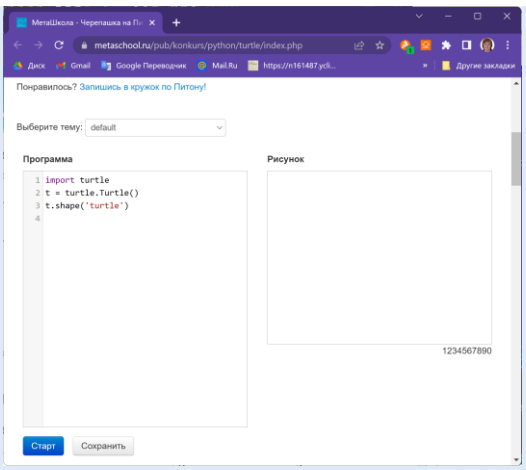
используя поисковую строку кнопки Пуск (пример 18.4).

В среду программирования входят редактор текстов, справочная система, исполнитель Черепаха и др. Для создания своей программы нужно выполнить команду **File → New File**. Будет создано новое окно, в котором можно писать программы (пример 18.5). В этом окне можно набирать и редактировать текст программы. Сохранять и открывать файлы с текстами программ можно так же, как в текстовом и графическом редакторах. При необходимости можно обратиться к справочной системе через меню **Help** (пример 18.6). Подробнее о работе со средой можно познакомиться в *Приложении* (с.).

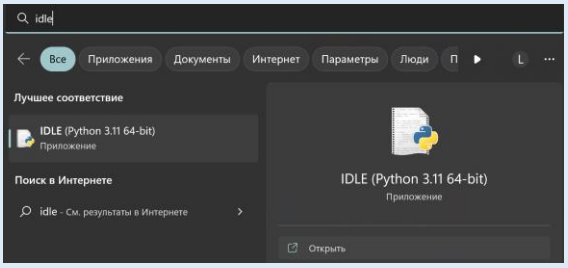
Документацию по работе в среде IDLE можно получить, выполнив команду **IDLE Doc**.

Документация по языку Python (команда **Python Docs F1**) открывается в браузере. Документация предлагается на английском языке. Перевести на русский язык можно используя контекстное меню, команда «Перевести на русский».

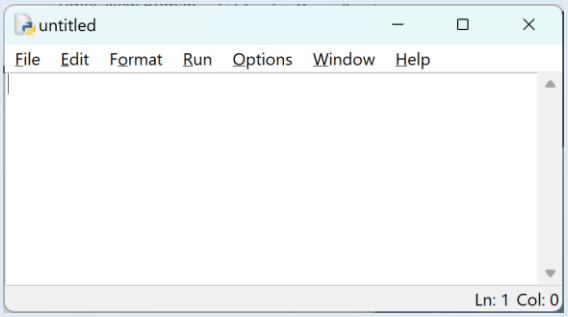
На странице **Search page/Страница поиска** (пример 18.7) можно найти интересующую информацию. В примере 18.8 показан список страниц, на которых можно



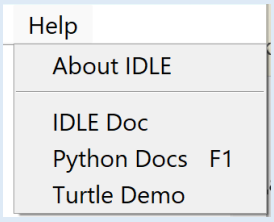
Пример 18.4. Поиск среды программирования IDLE



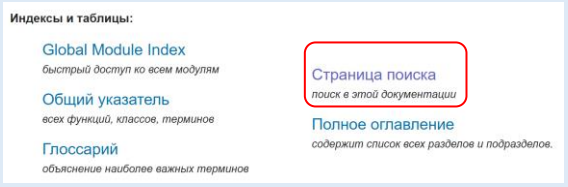
Пример 18.5. Окно для записи кода программы



Пример 18.6. Меню Help.



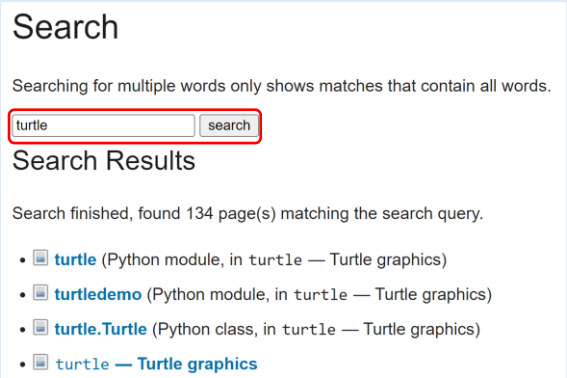
Пример 18.7. Выбор страницы поиска.



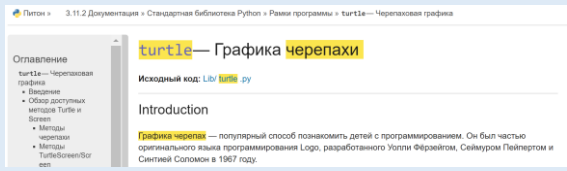
найти информацию об исполнителе Черепаха. Всего найдено 134 страницы. Переход по первой ссылке позволит подробно ознакомиться с ***Черепахой*** (пример 18.9), узнать о тех командах, которые доступны исполнителю, рассмотреть примеры программ и их результаты. Примеры могут быть скопированы с среду программирования и там выполнены. Перевод на русский язык доступен в браузере из контекстного меню.

Команда **Turtle Demo** позволяет открыть большое количество примеров для исполнителя Черепаха.

Пример 18.8. Поиск «turtle».



Пример 18.9. Часть страницы Графика Черепахи с переводом на русский.

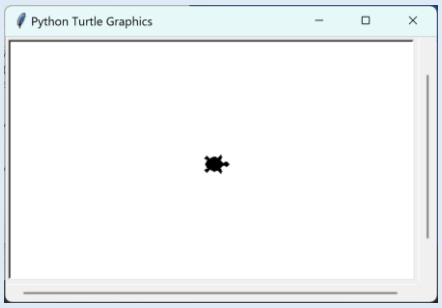


18.2. Компьютерный исполнитель Черепаха

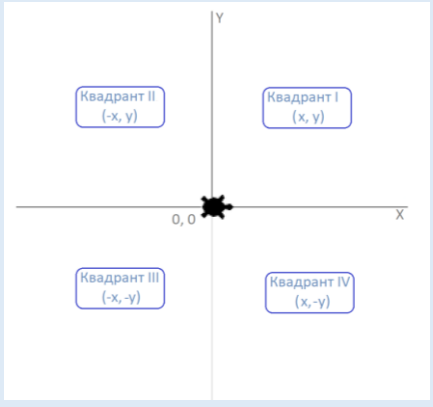
Стандартная библиотека Python содержит модуль turtle, предназначенный для обучения программированию. Этот модуль содержит набор функций, позволяющих управлять черепахой. ***Черепаха*** имеет перо, которое может поднимать, опускать, перемещать. При перемещении опущенного пера за ней остается след.

Среда обитания исполнителя ***Черепаха*** – координатная плоскость (пример 18.11). Исходное положение Черепахи – точка с координатами (0, 0), перо опущено. Координатная плоскость ***Черепахи*** по умолчанию определяется значениями x = 400, y =300. Это значит, что значения координаты x изменяются от -400 до 400, а координаты y – от -300 до 300. Размер окна ***Черепахи*** при этих

Пример 18.11. Среда обитания исполнителя ***Черепаха.***



Координатная плоскость ***Черепахи***, заданная значениями x и y



Внешний вид исполнителя

значениях 800×600.

Некоторые команды исполнителя ***Черепаха,*** с помощью которых можно написать программы, представлены в таблице:

Команда	Действие
`shape(X)`	Изменить значок черепахи
`penup()`	Не оставлять след при движении
`pendown()`	Оставлять след при движении
`forward(X)`	Пройти вперёд X пикселей
`backward(X)`	Пройти назад X пикселей
`left(X)`	Повернуться налево на X градусов
`right(X)`	Повернуться направо на X градусов

(Рассмотрите пример 18.12).

Для того, чтобы ***Черепаха*** могла выполнять команды, первой командой в программе нужно подключить модуль, содержащий команды управления ***Черепахой***:

```
import turtle
```

Каждая команда управления Черепахой записывается после ключевого слова `turtle` и отделяется от него точкой. Команды, записываемые после точки можно выбирать из выпадающего списка. Для отображения списка используется комбинация клавиш CTRL + пробел (пример 18.13). Команды Черепахи можно копировать, вырезать и вставлять также это делали в текстовом редакторе. При использовании сочетания клавиш CTRL + C, CTRL + V, CTRL + X, CTRL + Z должен быть включен английский язык.

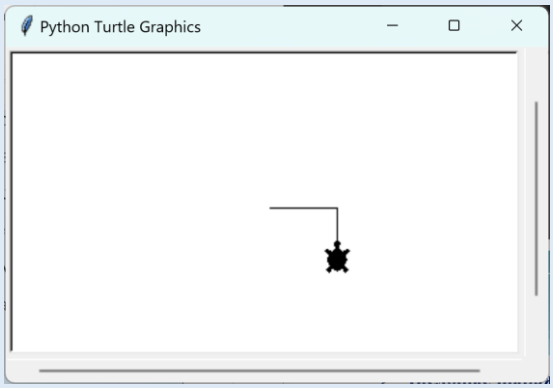
определяется значением параметра X команды `shape(X)`.

arrow	turtle	circle	square	triangle	classic
					

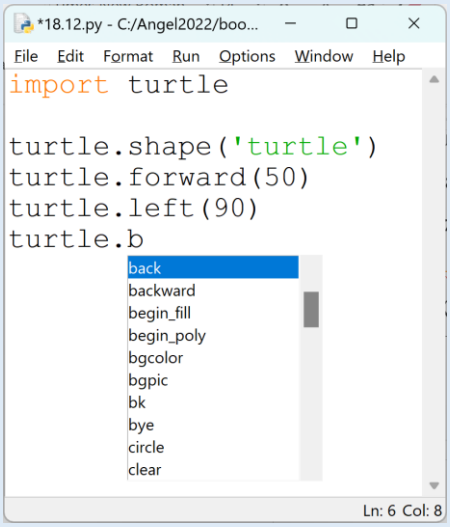
Вид исполнителя можно изменить при запуске программы. Чаще всего выбирают вид черепашки.

Пример 18.12. Использование команд `shape`, `forward`, `left`, `backward`:

```
turtle.shape('turtle')
turtle.forward(50)
turtle.left(90)
turtle.backward(40)
```



Пример 18.13.



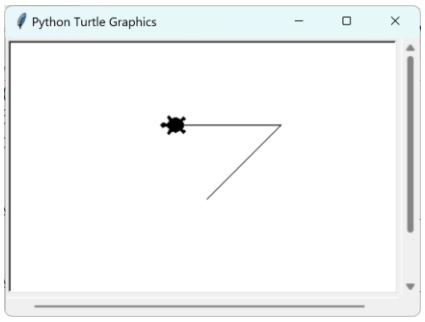
Пример 18.14. Алгоритм построения угла.

Для решения задачи с помощью исполнителя ***Черепаха*** необходимо:

1. Составить алгоритм решения задачи (пример 18.14)
2. Записать алгоритм в виде программы в окне текстового редактора среды программирования IDLE (пример 18.15).
3. Сохранить программу.
4. Выполнить программу: **Run→Run Module** (можно нажать клавишу F5).
5. Если получено неверное решение, в программу следует внести изменения и вновь ее выполнить.

Программа состоит из отдельных команд. В одной строке можно написать только одну команду. Если программа не сохранена перед выполнением, то среда предложит ее сохранить. Выполняться может только сохраненная программа.

Пример 18.14. Построить изображение угла, как на рисунке. Длины отрезков – 100, угол между ними - 45.

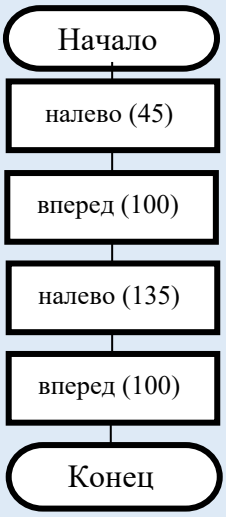


При выполнении программы могут возникать ошибки. Черепаха выполняет

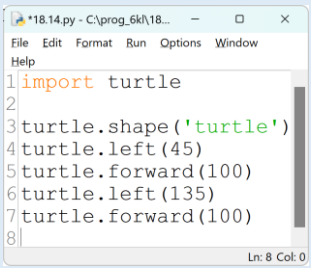
Словесное описание алгоритма:

Повернуть ***Черепаху*** на 45 налево.
Сдвинуть ***Черепаху*** на 100 вперед.
Повернуть ***Черепаху*** на 135 налево.
Сдвинуть ***Черепаху*** на 100 вперед.

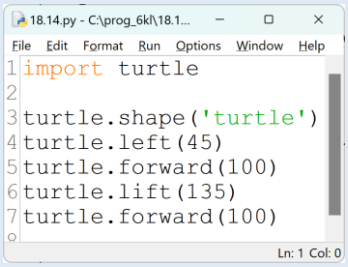
Графический способ записи алгоритма:



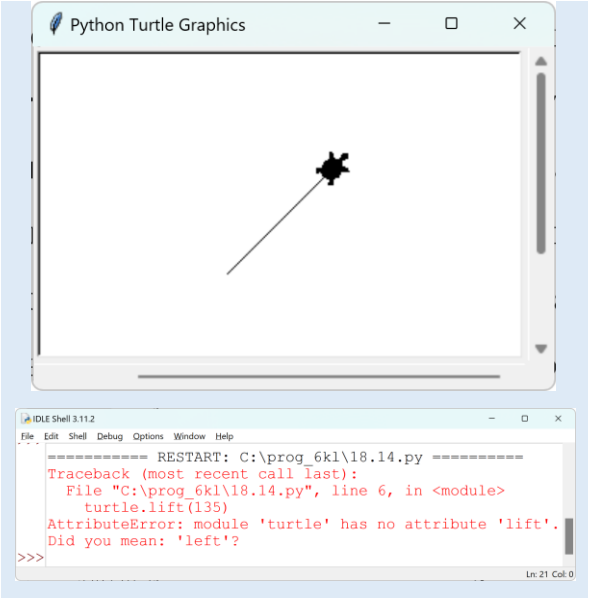
Пример 18.15. Программа рисования угла (программный способ записи алгоритма).



Пример 18.15. Программа с ошибкой в строке 6 (lift вместо left), графическое окно Черепахи, в котором выполнена часть программы до появления ошибки и реакция среды IDLE на неверную команду.



команды до первой ошибки, после чего останавливается. При этом в окне IDLE Shell выводится соответствующее сообщение (пример 18.15): номер строки, в которой допущена ошибка, ошибочная команда и предложением как ее исправить.



Показать/спрятать нумерацию строк в программе можно с помощью команды Options → Show (Hide) Lines Numbers.

1. Что такое среда программирования?
2. В какой среде программирования размещается компьютерный исполнитель ***Черепаха***?
3. Для чего предназначен исполнитель ***Черепаха***?
4. Какие команды входят в систему команд исполнителя ***Черепаха***?
5. Какое назначение команд `forward`, `backward`, `left`, `right`, `pendown`, `penup`?

Упражнения

1. С помощью справочной системы среды программирования IDLE получите справку о команде языка Python `turtle.shape`. Переведите ее на русский язык.
2. Запишите в окне редактора среды программирования IDLE текст нижеприведенной программы и определите результат ее выполнения.

```
import turtle
turtle.shape('turtle')
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(100)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
```

3. Внесите изменения в программу примера 18.14 так, чтобы ***Черепаха*** нарисовала угол в 60° между двумя отрезками, с длинами 90 и 150.

4. Составьте программу для получения изображения треугольника, проверьте правильность выполнения упражнения.
5. Напишите команды для построения изображения произвольного: а) квадрата; б) прямоугольника; в) прямоугольного треугольника; г) равнобедренного треугольника.
6. Напишите команды для построения изображений. После рисования сдвиньте Черепаху на 10 вправо от нижнего правого угла и поверните ее так, чтобы она смотрела вправо.

1. цифры 1

2. буквы Б

3. *цифры 4

§19. *Изучение и изменение готовых программ*

19.1. Дополнительные команды Черепахи

Кроме команд, которые были разобраны в предыдущем параграфе, у исполнителя Черепаха есть и другие. В таблице приведены команды исполнителя ***Черепаха.***

Команда	Действие
`color(x)`	Изменить цвет линии, которую рисует ***Черепаха***
`fillcolor(x)`	Изменить цвет заливки фигуры, которую рисует ***Черепаха***
`colormode(1.0)` `colormode(255)`	Сменить режим ввода значений цвета: 1.0 – текстом, 255 – числами
`pensize(x)`	Изменить толщину линии, которую рисует ***Черепаха***
`begin_fill()`	Команда которая прописывается перед фигурой, которую нужно будет залить

Пример 19.1. Задание цвета.

По умолчанию задан режим установки цвета с использованием текстовых значений. В 15-ой строке программы режим меняется. Теперь цвет можно задавать числовыми значениями. В 24-ой строке возвращен способ задания цвета по умолчанию.

Программа:

```
1. import turtle

2. turtle.shape('turtle')
3. turtle.pensize(2)
4. turtle.penup()
5. turtle.setpos(-100,0)
6. turtle.pendown()
7. turtle.color('darkred')
8. turtle.fillcolor('tomato')
9. turtle.begin_fill()
10. turtle.circle(50)
11. turtle.end_fill()
```

`end_fill()`	Залить фигуру, которая нарисована с помощью команд, расположенных между `begin_fill()`и `end_fill()`
`circle(r)`	Нарисовать окружность радиуса `r`
`setpos(x, y)`	Переместить ***Черепаху*** в точку с координатами `(x,y)`
`setheading(x)`	Задать направление движения ***Черепахи***
`setup(w,h)`	Изменить размеры окна ***Черепахи***: `w` – ширина окна, `h` – высота окна
`towards(x, y)`	Получить угол между текущим направлением ***Черепахи*** и прямой от ***Черепахи*** к точке `(x,y)`
`distance(x, y)`	Получить расстояние до точки `(x,y)`

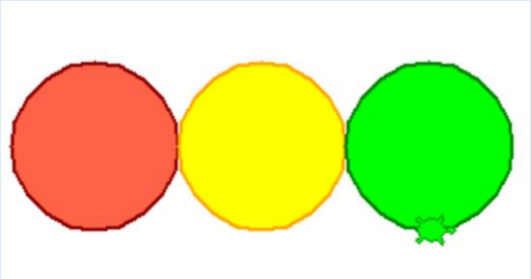
Цвет, который использует Черепаха, можно задавать с помощью английских названий цветов (см. Приложение, с.). В этом случае название цвета записывается в кавычках. Также цвет может задаваться числовыми значениями, которые определяют цвет. Посмотреть эти значения можно в графическом редакторе Paint. Если команду `color` записать с двумя параметрами – `color(x, y)`, то первый параметр *x* определит цвет линии, а второй *y* – цвет заливки. В примере 19.1 показаны разные способы задания цвета и смены режима отображения цветов.

Направление ***Черепахи*** задается величиной угла поворота. Некоторые значения приведены в примере 19.2.

Если текст программы большой, то ее сложно читать, поэтому в программе часто пишут комментарии – строки текста, которые

```
12. turtle.penup()
13. turtle.setpos(0,0)
14. turtle.pendown()
15. turtle.colormode(255)
16. turtle.color(255,165,0)
17. turtle.fillcolor(255,255,0)
18. turtle.begin_fill()
19. turtle.circle(50)
20. turtle.end_fill()

21. turtle.penup()
22. turtle.setpos(100,0)
23. turtle.pendown()
24. turtle.colormode(1.0)
25. turtle.color('green')
26. turtle.fillcolor('lime')
27. turtle.circle(50)
28. turtle.end_fill()
```

Результат выполнения программы:



Пример 19.2. Направление ***Черепахи***.

Угол поворота	Направление Черепахи
0	Вправо, на восток
90	Вверх, на север
180	Влево, на запад
270	Вниз, на юг

Пример 19.3. Комментарии в программе.

```
#желтый круг
turtle.penup()
turtle.setpos(0,0)
turtle.pendown()
turtle.colormode(255)
turtle.color(255,165,0)
turtle.fillcolor(255,255,0)
turtle.begin_fill()
turtle.circle(50)
turtle.end_fill()
```

поясняют, написанные команды. Перед текстом комментариев ставится знак # (пример 19.3). Комментарии можно писать по-русски. Исполнитель пропускает их при выполнении программы.

Здесь комментарием является строка

```
#желтый круг
```

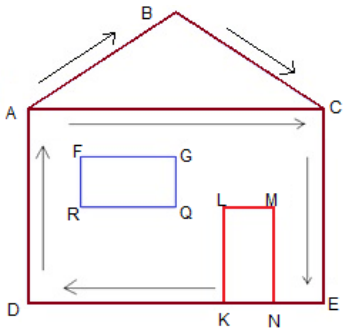
Она поясняет, что дальше расположены команды для рисования круга желтого цвета.

19.2. Изображение домика

Пример 19.4. Составить алгоритм построения изображения домика и написать программу для исполнителя ***Черепаха***.

Выберем следующий алгоритм построения изображения.

1. построим отрезки для крыши: AB, BC, CA;
2. построим отрезки для дома: AD, DE, EC;
3. построим отрезки для двери: KL, LM, MN;
4. построим отрезки для окна FG, GQ, QR, RF.



Для подбора размеров и координат желательно сначала нарисовать изображение на листе бумаги в клетку.

Словесное описание алгоритма:

```
Поднять перо
Сместиться в точку (-150, 50)
Опустить перо
Нарисовать крышу
    Налево (30)
```

Пример 19.4. Изображение домика.

Программа:

```
import turtle

turtle.shape('turtle')
turtle.pensize(2)
turtle.penup()
turtle.setpos(-150,50)
turtle.pendown()
turtle.color('brown')
#крыша
turtle.left(30)
turtle.forward(200)
turtle.right(60)
turtle.forward(200)
turtle.right(150)
turtle.forward(346)
#дом
turtle.left(90)
turtle.forward(200)
turtle.left(90)
turtle.forward(346)
turtle.left(90)
turtle.forward(200)
#окно
turtle.penup()
turtle.setpos(-100,0)
turtle.pendown()
turtle.setheading(0)
turtle.color('blue')
turtle.forward(100)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(100)
turtle.right(90)
turtle.forward(50)
#дверь
turtle.penup()
turtle.setpos(90,-150)
turtle.pendown()
turtle.setheading(90)
```

```
    Вперед (200)
    Направо(60)
    Вперед(200)
    Направо(150)
    Вперед(346)
Нарисовать дом
    Налево (90)
    Вперед (200)
    Налево (90)
    Вперед (346)
    Налево (90)
    Вперед (200)
Поднять перо.
Сместиться в точку (-100, 0).
Опустить перо и нарисовать окно.
Поднять перо.
Сместиться в точку (90, -150).
Опустить перо и нарисовать дверь.
```

Имеющиеся программы можно с некоторыми изменениями использовать для решения других задач.

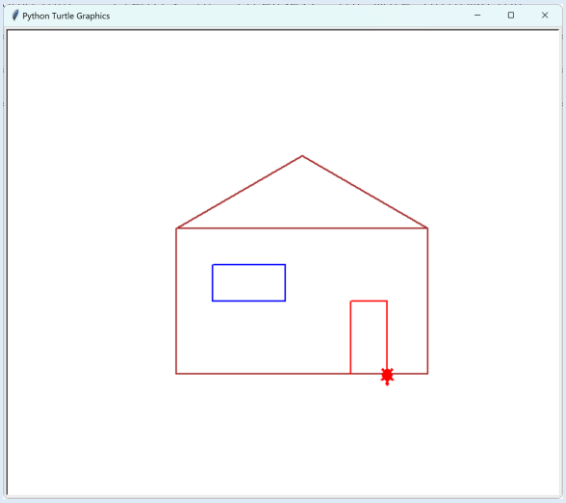
Пример 19.5. Требуется изменить изображение домика из примера 19.4.

При сравнении нового изображения и изображения из примера 19.4 обнаружим два отличия: первое – в размерах окна ***Черепахи***, второе – в форме окна. В целом рисунки похожи. Значит, для создания изображения нового домика можно использовать программу из примера 19.4. Внесем изменения в текст программы в окне редактора среды программирования, используя правила редактирования текста:

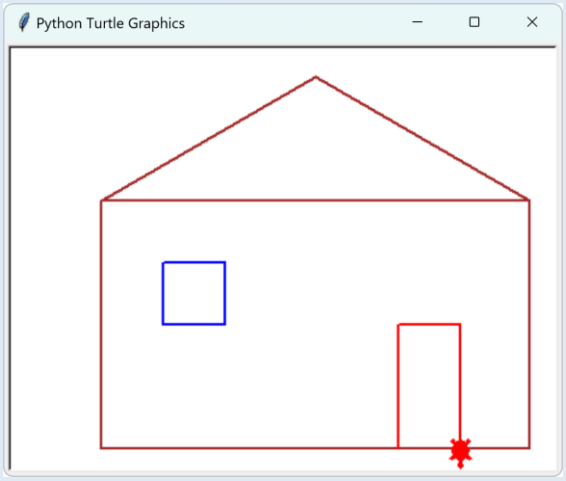
- добавим команду изменения окна начало программы;
- изменим фрагмент рисования окна домика.

```
turtle.color('red')
turtle.forward(100)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(100)
```

Результат работы программы:



Пример 19.5. Измененный рисунок.



Изменения в программе:

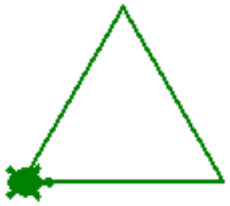
```
import turtle

turtle.shape('turtle')
turtle.setup(450, 350)
#окно
turtle.penup()
turtle.setpos(-100,0)
turtle.pendown()
turtle.setheading(0)
turtle.color('blue')
turtle.forward(100)50
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(100)50
```

Остальные команды в программе останутся неизменными.

```
turtle.right(90)
turtle.forward(50)
```

19.3. Изображение треугольников

Пример 19.6. Написать программу построения треугольника:



У треугольника три стороны одинаковой длины. Все углы равны по 60°. Угол поворота ***Черепахи*** – 180° – 60° = 120°.

Алгоритм рисования треугольника (алгоритм 1):

```
Вперед (100)
Влево (120)
Вперед (100)
Влево (120)
Вперед (100)
```

Аналогичный результат можно получить и с помощью другого алгоритма (алгоритм 2):

```
В точку(100, 0)
В точку(50, 75)
В точку(0, 0)
```

Программа данного алгоритма приведена в примере 19.7.

Если сравнить два алгоритма, то можно сделать следующие выводы.

1. Второй алгоритм имеет меньшее количество команд, чем первый.

2. Треугольник, построенный по первому алгоритму, может располагаться в

Пример 19.6. Программа получения изображения треугольника:

```
import turtle

turtle.shape('turtle')
turtle.color('green')
turtle.pensize(2)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
```

Пример 19.7. Программа получения изображения треугольника:

```
import turtle

turtle.shape('turtle')
turtle.color('green')
turtle.pensize(2)
turtle.setpos(100,0)
turtle.setpos(50,75)
turtle.setpos(0,0)
```

Пример 19.8. Программа рисования елки.

```
import turtle

turtle.shape('turtle')
turtle.color('green')
turtle.pensize(2)
#нижний треугольник
turtle.penup()
turtle.setpos(-50,-85)
turtle.pendown()
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
#средний треугольник
turtle.penup()
turtle.setpos(-50,0)
turtle.pendown()
```

любом месте плоскости. Его местоположение зависит только от начального положения ***Черепахи***.

3. Треугольник, построенный по второму алгоритму, может быть построен только в одном месте плоскости. Его местоположение определено координатами вершин.

4. Первый алгоритм, невзирая на большее количество команд, является более универсальным. Его можно использовать для построения других изображений.

Пример 19.8. Написать программу построения елки, состоящей из трех одинаковых треугольников. Треугольники рисовать используя алгоритм 1. Алгоритм рисования елки может быть следующим:

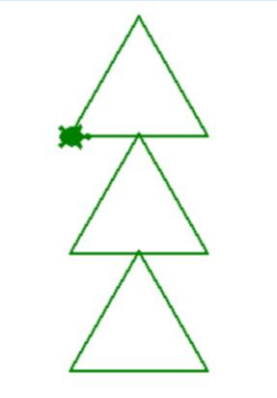
```
Поднять перо
В точку (-50,-85)
Опустить перо
Нарисовать треугольник
Поднять перо
В точку (-50,0)
Опустить перо
Нарисовать треугольник
Поднять перо
В точку (-50,85)
Опустить перо
Нарисовать треугольник
```

Для рисования треугольника в программе нужно три раза скопировать фрагмент из программы примера 19.6.

Пример 19.9. Изменить программу примера 19.6 для получения прямоугольного треугольника с углами 45, 45, 90 и длиной катета 100.

```
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
#верхний треугольник
turtle.penup()
turtle.setpos(-50,85)
turtle.pendown()
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
```

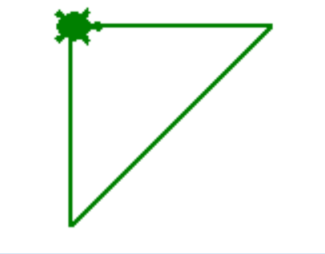
Результат работы программы:



Пример 19.9. Программа рисования прямоугольного треугольника.

```
import turtle

turtle.shape('turtle')
turtle.color('green')
turtle.pensize(2)
turtle.forward(100)
turtle.right(135)
turtle.forward(141)
turtle.right(135)
turtle.forward(100)
turtle.right(90)
```

Результат работы программы:



Определим отличия данного треугольника от треугольника в примере 19.6.

1. У этого треугольника углы не одинаковы. Черепаха должна дважды повернуться на угол 135° (180° – 45°) и затем на угол 90°.

2. Длины двух сторон треугольника нам известны, а третьей нет. Для вычисления ее длины можно воспользоваться тем, что длина гипотенузы у такого треугольника примерно в 1,41[5] раз больше катета.

Для вычисления расстояния можно использовать команду `distance`. Для этого нужно в программе заменить команду

```
turtle.forward(141)
```

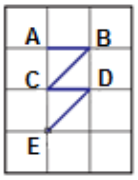
на команду

```
turtle.forward(turtle.distance(0,100))
```

Однако при таком подходе мы потеряем универсальность рисования нашего треугольника: его местоположение будет привязано к точке (0, 100).

19.4. Изображение цифры «3»

Пример 19.10. Написать программу построения изображения цифры 3 в почтовом индексе с помощью исполнителя ***Черепаха***.



Алгоритм построения изображения:

```
Сместиться в точку A(-50,60)
Опустить перо
Изобразить цифру, двигаясь по отрезкам AB, BC, CD, DE.
```

Данное изображение состоит из двух одинаковых фрагментов, каждый из которых представляет собой прямоугольный треугольник без прорисованного одного катета. Поэтому можно воспользоваться программой примера 19.9. Длина горизонтальной линии равна 30, диагональной – 30·1,41 ≈ 42.

В некоторых изображениях часто повторяются одинаковые фрагменты. Для создания программ построения таких

Пример 19.10. Программа получения изображения цифры 3.

```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
turtle.penup()
turtle.setpos(-50,60)
turtle.pendown()
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
```

Результат работы программы:

Пример 19.11. Программа построения изображения из трех одинаковых цифр разного цвета.

```
import turtle

turtle.shape('turtle')
turtle.pensize(2)
```

[5] Почему это равенство верное, вы узнаете на уроках математики.

изображений можно скопировать повторяющийся фрагмент программы и использовать его нужное число раз, так как это делали в примере 19.8 для построения елки.

Пример 19.11. Написать программу для построения изображения, состоящего из трех цифр «3», изображенных красным, зеленым и синим цветами.

Мы видим, что программу построения этого изображения можно составить на основе программы примера 19.8. Изображение первой цифры начинается от верхней точки слева, ее координаты (-50, 60). Координаты такой же точки для второй цифры (0, 60), для третьей цифры – (50, 60).

Таким образом, для создания изображения из трех цифр «3» нужно скопировать в тексте программы примера 19.10 следующий фрагмент:

```
turtle.penup()
turtle.setpos(-50,60)
turtle.pendown()
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
```

Затем следует вставить скопированный фрагмент нужное число раз, внести изменения для переходов в начальную точку и дописать команды изменения цвета.

```
#первая тройка
turtle.penup()
turtle.color("red")
turtle.setpos(-50,60)
turtle.pendown()
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
#вторая тройка
turtle.penup()
turtle.color("green")
turtle.setpos(0,60)
turtle.pendown()
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
#третья тройка
turtle.penup()
turtle.color("blue")
turtle.setpos(50,60)
turtle.pendown()
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
```

Результат работы программы:

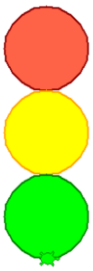
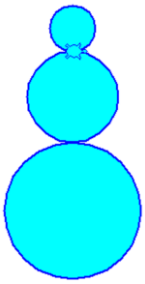
Значения параметров команд Черепахи можно записывать выражением. Например, вместо команды `turtle.forward(42)` можно записать команду `turtle.forward(30*1.41)`

1. Как получить справочную информацию о командах `setheading`, `pensize`?
2. Как задать цвет линии ***Черепахи***?
3. Как закрасить нарисованную ***Черепахой*** фигуру?
4. Какое значение должно быть у команды `forward`, если ***Черепаха*** должна переместиться по диагонали квадрата с длиной стороны 120?

Упражнения

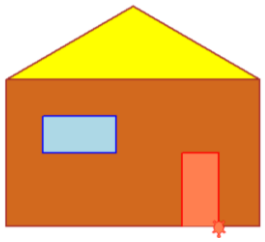
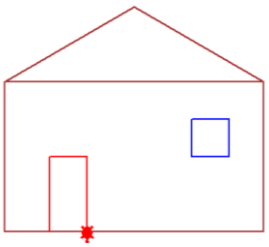
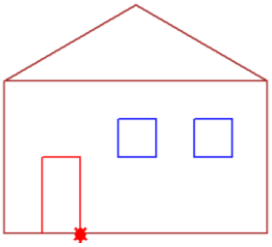
1. Измените программу примера 19.1 для получения изображений:

а) б)

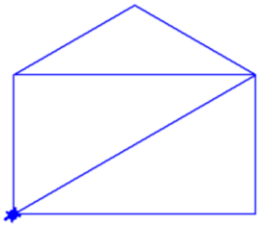
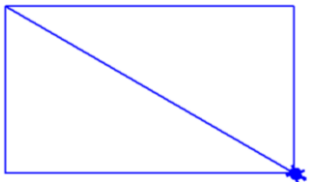
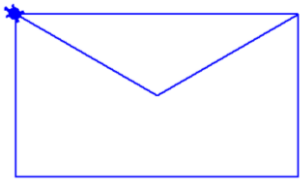



2. Измените программу из примера 19.4 для получения изображений. Конечное положение ***Черепахи*** может быть другим.

а) б) в)

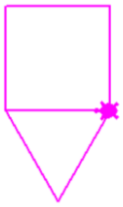




г) д) е)

3. Используйте программы примеров 19.6 и 19.8 для получения изображений:

а) б) в)


4. Вставьте в редактор среды программирования IDLE текст программы, определите результат ее выполнения:

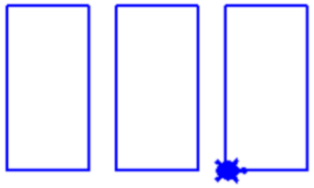
```
import turtle

turtle.shape('turtle')
#буква Б
turtle.left(90)
turtle.penup()
turtle.forward(30)
turtle.pendown()
turtle.right(90)
turtle.forward(30)
turtle.right(90)
turtle.forward(30)
turtle.right(90)
turtle.forward(30)
turtle.right(90)
turtle.forward(60)
turtle.right(90)
turtle.forward(30)
#переход
turtle.penup()
turtle.setpos(40, 0)
turtle.pendown()
#буква Г
turtle.left(90)
turtle.forward(60)
turtle.right(90)
turtle.forward(30)
#переход
turtle.penup()
turtle.setpos(80, 0)
turtle.pendown()
#буква У
turtle.forward(30)
turtle.left(90)
turtle.forward(60)
turtle.left(90)
turtle.penup()
turtle.forward(30)
turtle.pendown()
turtle.left(90)
turtle.forward(30)
turtle.left(90)
turtle.forward(30)
```

Измените программу для построения изображений

5. Скопируйте и вставьте в редактор среды программирования IDLE текст программы, определите результат ее выполнения:

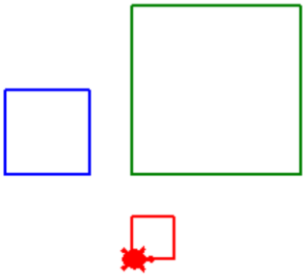
```
import turtle

turtle.shape('turtle')
turtle.color('blue')
turtle.pensize(2)
turtle.forward(60)
turtle.left(90)
turtle.forward(120)
turtle.left(90)
turtle.forward(60)
turtle.left(90)
turtle.forward(120)
turtle.left(90)
```

Измените программу для построения изображения:

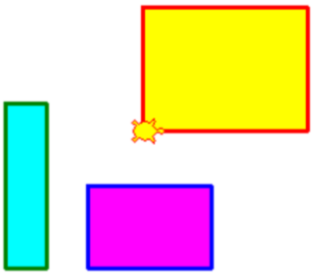


6. *Составьте программы для выполнения заданий. Подумайте, какие из уже выполненных заданий можно использовать для получения этих изображений.

а)



б)



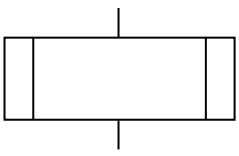
§20. *Составление программ. Использование подпрограмм (вспомогательных алгоритмов)*

20.1. Понятие вспомогательного алгоритма

Как видно из предыдущего параграфа, исполнителю ***Черепаха*** нередко приходится строить одно и то же изображение в одной программе несколько раз. Построение этого изображения удобно оформить в виде отдельного алгоритма. Такие алгоритмы называют вспомогательными.

Вспомогательный алгоритм – алгоритм, который можно целиком использовать в других алгоритмах.

Вспомогательный алгоритмы можно использовать необходимое число раз, обращаясь к его названию (имени). Для обращения к вспомогательному алгоритму в блок-схемах используется блок:



Вспомогательный алгоритм в языке

На заре создания первых компьютеров при разработке программ применялся прием проектирования «снизу вверх»: вначале создавали простейшие подпрограммы, затем их использовали в более сложных программах. В середине 60-х гг. XX в. стал применяться метод пошаговой детализации алгоритмов (проектирование «сверху вниз»). Этот метод заложен в основу процедурных языков программирования (Pascal, C, Python и др.).

Пример 20.1. Рисование елочки.

Блок-схема алгоритма рисования елочки включает:

1. Блок-схему вспомогательного алгоритма для рисования одного элемента елочки (треугольника):

Python записывается в виде функции:

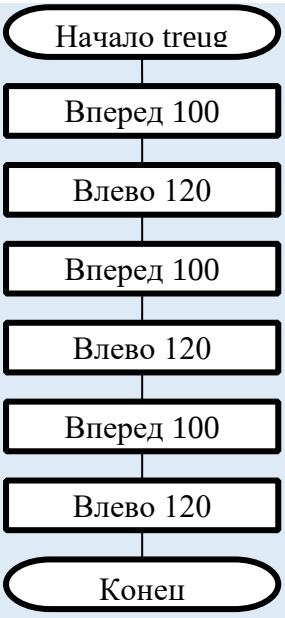
```
def имя_функции():      <- Заголовок функции
    команда1
    команда2            <- Команды (тело функции)
    …
```

Имя функции может содержать буквы латинского алфавита, цифры, знак подчеркивания (_). Первый символ в имени процедуры не может быть цифрой. После имени функции ы скобках могут указываться параметры, от которых зависит результат работы функции. Если параметров нет, то наличие скобок все равно является обязательным. После скобок ставится двоеточие – «:». Все команды, которые относятся к телу функции пишутся со сдвигом вправо. Сдвиг устанавливают клавишей Tab.

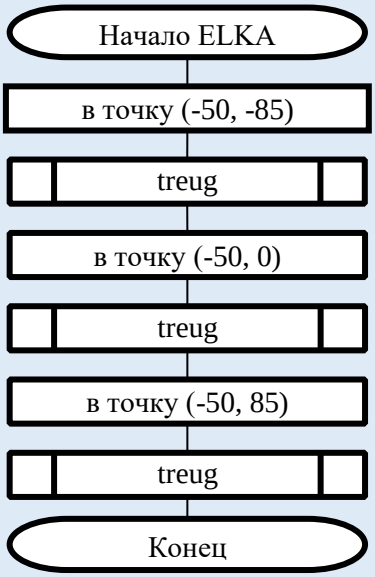
Команду выполнения вспомогательного алгоритма называют **вызовом функции**. Команду вызова записывают в основном алгоритме (программе) путем указания имени процедуры.

Пример 20.1. Написать программу для рисования елочки с использованием вспомогательного алгоритма.

В примере 19.8 мы уже строили елочку, используя изображение треугольников. Если проанализировать алгоритм, то можно увидеть, что команда построить треугольник указана трижды. Это значит, что можно не копировать три раза команды для изображения треугольника, а оформить вспомогательный



2. Блок-схему построения елочки с использованием вспомогательного алгоритма:



Программа построения елочки:

```
import turtle

turtle.shape('turtle')
turtle.pensize(2)
def treug():
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
    turtle.left(120)
#нижний треугольник
turtle.penup()
turtle.setpos(-50, -85)
turtle.pendown()
```

алгоритм, который будет строить треугольник.

Треугольник строился с помощью следующих команд:

```
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
```

Опишем эту последовательность команд в виде вспомогательного алгоритма `treug`, который будет использоваться в неизменяемом виде несколько раз (в данном случае три раза).

В этой программе есть еще три команды, которые повторяются трижды:

```
turtle.penup()
turtle.setpos(…, …)
turtle.pendown()
```

Эти команды позволяют осуществить переход от одного треугольника к другому Различие в использовании этих команд только в координатах, в которые должна переместиться ***Черепаха***. Можно оформить еще одну функцию, которая будет осуществлять переход. Эта функция будет зависеть от координат точки, в которую нужно переместит ***Черепаху***. Назовем эту функцию – `p(x, y)`. В примере 20.2 приведена измененная программа.

Если считать, что рисование треугольника зависит от точки, в которой Черепаха начинает его рисовать, то функция рисования треугольника тоже будет зависеть от параметров (x, y). Алгоритм рисования

```
treug()
#средний треугольник
turtle.penup()
turtle.setpos(-50, 0)
turtle.pendown()
treug()
#верхний треугольник
turtle.penup()
turtle.setpos(-50, 85)
turtle.pendown()
treug()
```

Пример 20.2. Измененная программа рисования елки.

```
import turtle

turtle.shape('turtle')
turtle.pensize(2)
def treug():
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
    turtle.left(120)
def p(x, y):
    turtle.penup()
    turtle.setpos(x,y)
    turtle.pendown()
#нижний треугольник
p(-50, -85)
treug()
#средний треугольник
p(-50, 0)
treug()
#верхний треугольник
p(-50, 85)
treug()
```

Пример 20.3. Новые изменения в программе рисования елки.

```
import turtle

turtle.shape('turtle')
turtle.pensize(2)
def treug(x, y):
    p(x, y)
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
def p(x, y):
    turtle.penup()
    turtle.setpos(x, y)
    turtle.pendown()
    turtle.setheading(0)
```

треугольника в этом случае буде начинаться с команды перевода Черепахи в точку с координатами (x, y). Программа приведена в примере 20.3. В данном примере функция `p` вызывается из функции `treug`.

```
#нижний треугольник
treug(-50, -85)
#средний треугольник
treug(-50, 0)
#верхний треугольник
treug(-50, 85)
```

Здесь у функции `treug` убрали последнюю команду `turtle.left(120)`, которая разворачивала ***Черепаху*** в начальное положение. Вместо нее в функцию `p` добавлена команда `turtle.setheading(0)`, которая устанавливает угол поворота, равный 0.

20.2. Решение практических задач с использованием функций

Пример 20.4. Оформить с помощью функций программу рисования трех троек на почтовом конверте из примера 19.11.

Функция для рисования одной тройки `cifr_3` будет функцией, которая зависит от начального положения Черепахи – координат (x, y). Функцию p(x, y) , устанавливающую Черепаху в нужную точку возьмем из примера рисования елочки.

Поскольку каждая тройка должна рисоваться своим цветом, то добавим еще один параметр `c` в описание функции.

На примере рисования елочки и трех троек мы увидели, что вспомогательные алгоритмы позволяют сократить код программы, поскольку не приходится одинаковые части программы записывать несколько раз. Однако вспомогательные алгоритмы используют и тогда, когда задача

Пример 20.4. Программа построения изображения из примера 19.11:

```
import turtle

turtle.shape('turtle')
turtle.pensize(2)
def cifr_3(x, y, c):
    p(x, y)
    turtle.color(c)
    turtle.forward(30)
    turtle.right(135)
    turtle.forward(42)
    turtle.left(135)
    turtle.forward(30)
    turtle.right(135)
    turtle.forward(42)
def p(x, y):
    turtle.penup()
    turtle.setpos(x, y)
    turtle.pendown()
    turtle.setheading(0)
#первая тройка
cifr_3(-50, 60, 'red')
#вторая тройка
cifr_3(0, 60, 'green')
#третья тройка
cifr_3(50, 60, 'blue')
```

Пример 20.5. Программа построения изображения

```
import turtle
```

разбивается на части – подзадачи. Вспомогательный алгоритм решает некоторую подзадачу основной задачи.

Пример 20.5. Написать программу для построения изображения, состоящего из трех цифр 1, 3 и 7, изображенных красным, зеленым и синим цветами.

В данном случае в изображении нет повторяющихся фрагментов. Однако задача разбивается на три подзадачи:

1. нарисовать цифру 1;
2. нарисовать цифру 3;
3. нарисовать цифру 7.

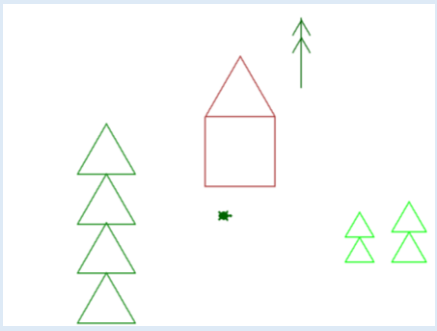
Функцию для рисования цифры 3 можно взять из предыдущего примера. Остается написать функции для рисования цифр 1 и 7.

При решении реальных задач над проектом могут работать несколько человек. Каждый выполняет свою часть работы и оформляет ее как отдельный вспомогательный алгоритм. Важно, чтобы вспомогательные алгоритмы, написанные разными людьми, правильно выполнялись в одном проекте. Для этого устанавливаются определенные договоренности, позволяющие определить единые подходы к написанию текста программы.

Для исполнителя ***Черепаха*** такой договоренностью условимся считать следующее правило: рисование любой фигуры начинается с перевода ***Черепахи*** в начальную

```
turtle.shape('turtle')
turtle.pensize(2)
def cifr_1(x, y, c):
    p(x, y)
    turtle.color(c)
    turtle.left(45)
    turtle.forward(42)
    turtle.right(135)
    turtle.forward(60)
def cifr_3(x, y, c):
    p(x, y)
    turtle.color(c)
    turtle.forward(30)
    turtle.right(135)
    turtle.forward(42)
    turtle.left(135)
    turtle.forward(30)
    turtle.right(135)
    turtle.forward(42)
def cifr_7(x, y, c):
    p(x, y)
    turtle.color(c)
    turtle.forward(30)
    turtle.right(135)
    turtle.forward(42)
    turtle.left(45)
    turtle.forward(30)
def p(x,y):
    turtle.penup()
    turtle.setpos(x,y)
    turtle.pendown()
    turtle.setheading(0)
cifr_1(0, 30, 'red')
cifr_3(50, 60, 'green')
cifr_7(100, 60, 'blue')
p(150,0)
```

Пример 20.6. «Пейзаж»



Пример 20.7. Список функций для рисования «пейзажа».

- `treug` – для построения треугольника;
- `p` – для перевода Черепахи в начальную точку;
- `b_el` – для рисования большой ели;

точку. Начальной точкой договоримся считать нижний левый угол изображения. В начальной точке ***Черепаха*** смотрит вправо (на восток).

Рассмотрим следующий пример. Пусть нескольким шестиклассникам поручили разработать программу рисования некоторого «пейзажа». «Пейзаж» состоит из следующих элементов: дом, три ели (одна большая и две маленькие) и сосна (пример 20.6).

Шестеро шестиклассников могут распределить работу между собой следующим образом:

- первый пишет вспомогательный алгоритм рисования сосны;
- второй пишет вспомогательный алгоритм рисования дома;
- третий пишет вспомогательный алгоритм рисования треугольника. В этом алгоритме желательно добавить еще один параметр – длина стороны треугольника;
- четвертый пишет вспомогательный алгоритм для маленькой елки, основываясь на вспомогательном алгоритме рисования треугольника. Координата x у обоих треугольников одинакова, координата y может быть вычислена по формуле $y + 0.86 \cdot d$, где d – длина стороны треугольника;
- пятый пишет вспомогательный алгоритм для большой елки, основываясь на
- `m_el` – для рисования маленькой ели;
- `dom` – для рисования дома;
- `sosna` – для рисования сосны.

Пример 20.8. Программа для рисования «пейзажа».

```
import turtle

turtle.shape('turtle')
turtle.pensize(2)
def treug(x, y, d):
    p(x,y)
    turtle.forward(d)
    turtle.left(120)
    turtle.forward(d)
    turtle.left(120)
    turtle.forward(d)
def p(x,y):
    turtle.penup()
    turtle.setpos(x,y)
    turtle.pendown()
    turtle.setheading(0)
def m_el(x, y, d):
    treug(x, y, d)
    treug(x, y + d * 0.86, d)
def b_el(x, y, d):
    m_el(x, y, d)
    m_el(x, y + d * 1.72, d)
def dom(x, y, d):
    p(x,y)
    turtle.forward(d)
    turtle.left(90)
    turtle.forward(d)
    turtle.left(60)
    treug(x, y + d, d)
    turtle.left(30)
    turtle.forward(d)
def sosna(x, y, d):
    p(x + d / 8 , y)
    turtle.left(90)
    turtle.forward(d)
    p(x, y + d / 2)
    turtle.left(60)
    turtle.forward(d / 4)
    turtle.right(120)
    turtle.forward(d / 4)
    p(x, y + 3 * d / 4)
    turtle.left(60)
    turtle.forward(d / 4)
    turtle.right(120)
    turtle.forward(d / 4)
turtle.color('green')
b_el(-250,-285, 100)
turtle.color('lime')
m_el(210, -180, 50)
m_el(290, -180, 60)
```

вспомогательном алгоритме рисования маленькой елки;

- шестой пишет основной алгоритм, размещая элементы пейзажа на поле ***Черепахи***.

В примере 20.7 приведен список функций для рисования «пейзажа», а в примере 20.8 размещена программа рисования «пейзажа». Она содержит функции:

Имея в своем распоряжении вспомогательные алгоритмы, можно легко изобразить и другие «пейзажи». При этом не придется переписывать сами вспомогательные алгоритмы. Достаточно выбрать место размещения объекта на поле ***Черепахи***, указав координаты начальной точки объекта.

Например, можно сохранить все функции из примера 20.8, а в текст основной части программы внесли изменения (пример 20.9).

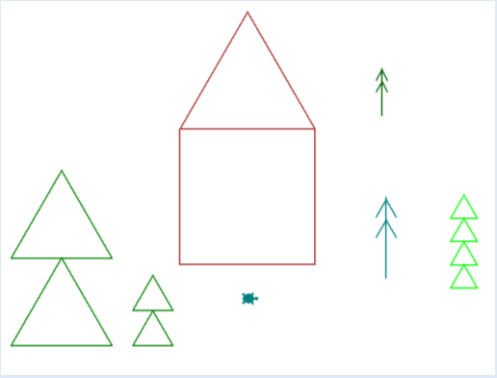
Результат работы программы после изменений показан в примере 20.10.

```
turtle.color('brown')
dom(-30, -50, 120)
turtle.color('darkgreen')
sosna(120, 120, 120)
p(0,-100)
```

Пример 20.9. Изменения в программе.

```
turtle.color('green')
m_el(-350, -220, 150)
m_el(-170, -220, 60)
turtle.color('lime')
b_el(300,-135, 40)
turtle.color('brown')
dom(-100, -100, 200)
turtle.color('darkgreen')
sosna(190, 120, 70)
turtle.color('teal')
sosna(190, -120, 120)
p(0,-150)
```

Пример 20.10. Результат изменений.



1. Какие алгоритмы называются вспомогательными?
2. Для чего нужны вспомогательные алгоритмы?
3. Какое ключевое слово используется для описания функции?
4. Как выполнить вызов функции?
5. Как понять, какие команды составляют тело функции?

Упражнения

1. Измените программу рисования елочки так, чтобы треугольник рисовался закрашенным.

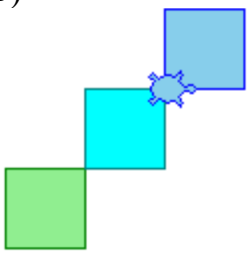
2. Используя составленные ранее программы для исполнителя ***Черепаха,*** как вспомогательные, составьте программу для построения изображения:

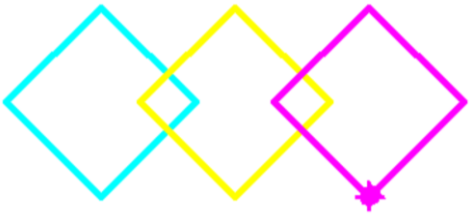
3. Используя функции из примера 20.8, придумайте свой пейзаж.
4. Составьте программы построения изображения всех цифр от 0 до 9.
5. Запишите программы написания слов для исполнителя ***Черепаха***. Используйте вспомогательные алгоритмы рисования букв.

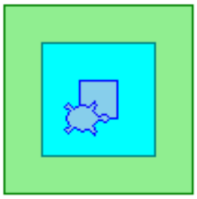
а)  б) 

6. Составьте программы построения изображений. Что могут обозначать эти изображения?

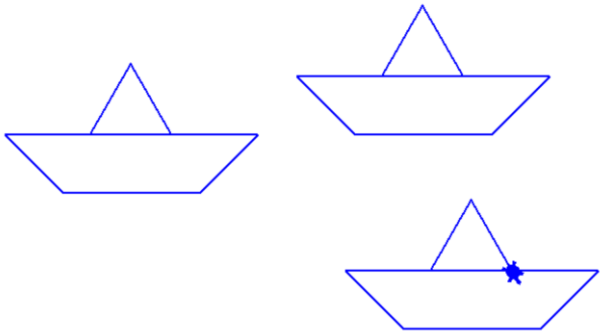
а) 

б) 

в) 

г) 

7. Составьте программу для построения изображения:

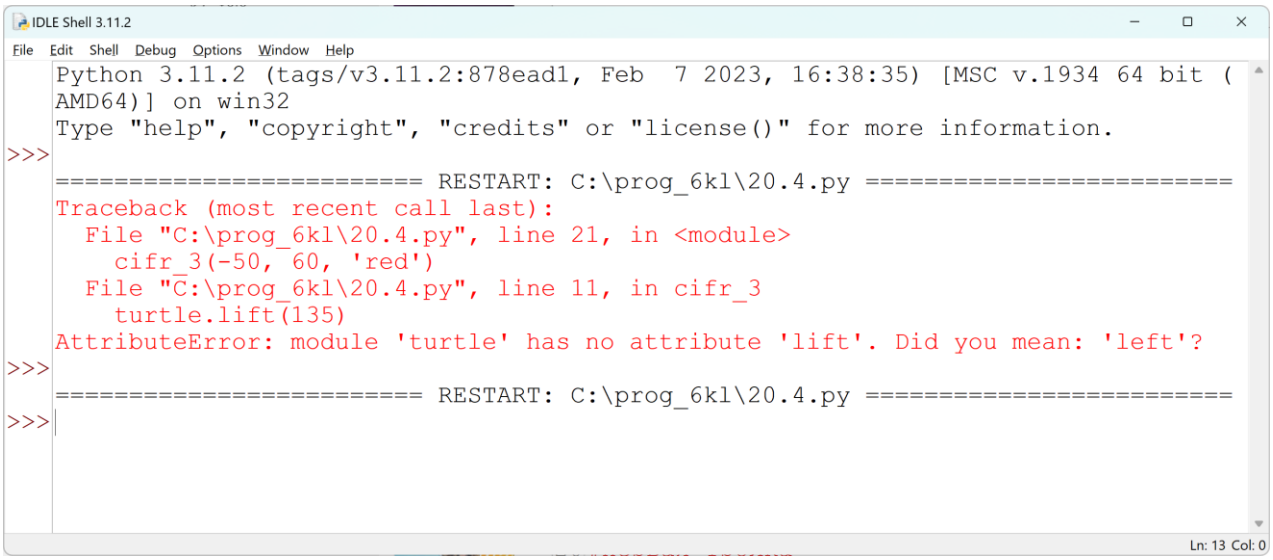


Приложение к главе 6. Работа в среде программирования IDLE

IDLE – среда программирования, ориентированная поддержку работы с языком Python. Среда имеет минималистический интерфейс, присущий простому текстовому редактору. Среда имеет главное окно (IDLE Shell), в котором выводятся сообщения о выполнимой программе и сообщения об ошибках, если они появляются при выполнении программы. Для каждой программы создается отдельное окно, в котором можно работать с текстом программы: набирать, редактировать, выполнять.

Главное окно среды программирования IDLE

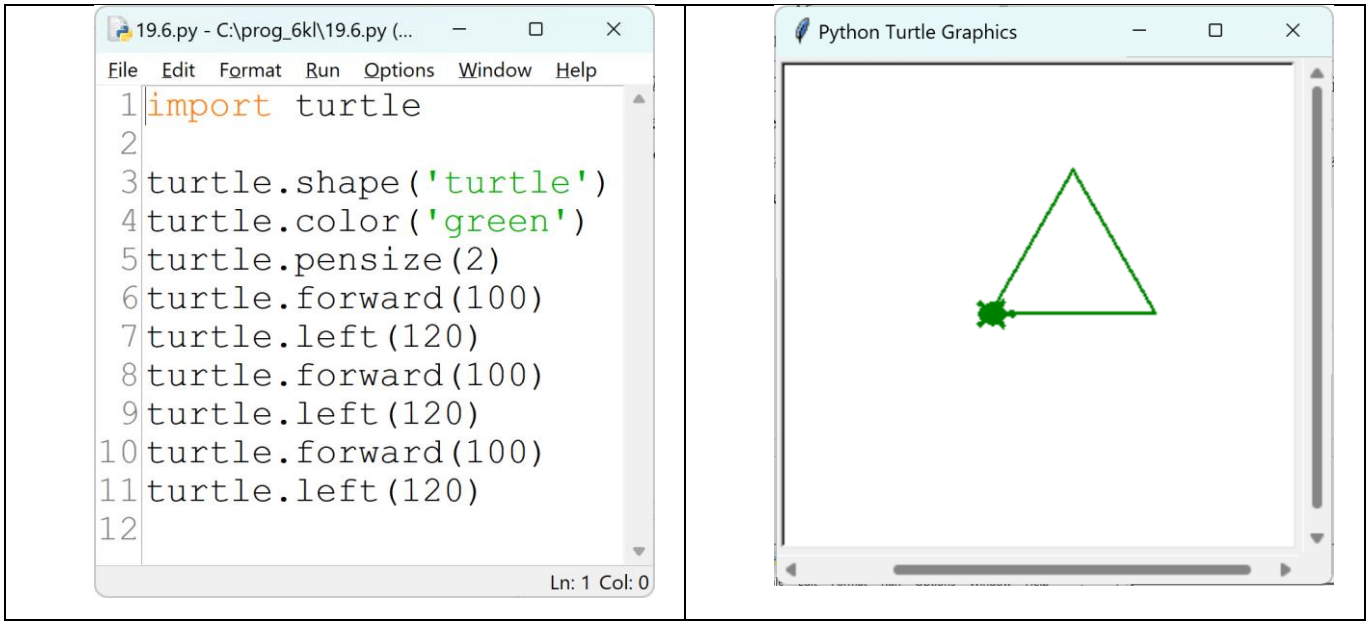
На рисунке приводится вид главного окна программы. Сообщение красным цветом – это сообщение о найденной ошибке. При появлении ошибки выполнение программы останавливается. Следующее сообщение говорит о том, что программа была успешно выполнена.



Окно с текстом программы на языке Python и окно Черепахи

Для создания программ выполняется команда File → New File, которая создает новое окно. После сохранения программы окно получает имя, соответствующее имени файла. Результат работы программы для исполнителя ***Черепаха*** отображается в окне Python Turtle Graphics.

Окно с текстом программы	Окно ***Черепахи***



Главное окно, а также окно, предназначенное для набора текстов программы, имеют меню. С помощью мыши можно открыть тот или иной пункт меню. Меню File, Edit, Options, Windows, Help у них общие. У главного окна есть меню Shell и Debug, у окна с программой меню Run. Рассмотрим некоторые команды меню.

Меню File

Назначение: выполнение операций с файлами текстов программ.

Команда	*Комбинация клавиш*	*Назначение*
New File	Ctrl +N	Открытие окна для нового файла
Open…	Ctrl +O	Открытие (загрузка) файла
Recent Files		Список недавно открытых файлов
Save	Ctrl +S	Сохранение файла с прежним именем
Save	Ctrl +Shift + S	Сохранение файла с новым именем
Close Windows	Alt + A4	Закрыть текущее окно файла
Exit IDLE	Ctrl + Q	Выход из системы программирования

Меню Edit

Назначение: выполнение операций редактирования текста программы.

Команда	*Комбинация клавиш*	*Назначение*
Undo	Ctrl + Z	Отмена последней операции редактирования текста программы (откатка)
Redo	Ctrl + Shift + Z	Восстановление предыдущей операции редактирования текста программы
Select All	Ctrl + A	Выделить все
Cut	Ctrl + X	Перемещение выделенного фрагмента текста из окна редактора в текстовый буфер обмена (промежуточную память)
Copy	Ctrl +C	Копирование выделенного фрагмента из окна редактора в буфер обмена
Paste	Ctrl + V	Копирование выделенного текста из буфера обмена в окно редактора
Find…	Ctrl + F	Поиск текста
Replace…	Ctrl + H	Поиск текста и замена его новым текстом
Show Completions	Ctrl + пробел	Показать контекстное меню с подсказками команд

Меню Run

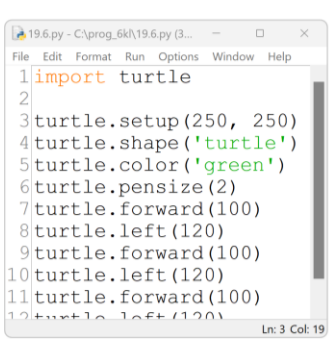
Назначение: выполнение команд управления ходом выполнения программы.

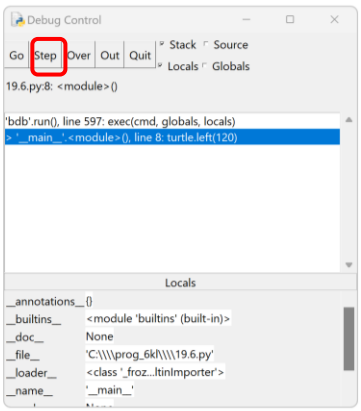
Команда	*Комбинация клавиш*	*Назначение*
Run Module	F5	Выполнение программы

Меню Debug (главное окно, IDLE Shell)

Назначение: отладка (пошаговое выполнение программ).

Команда	*Назначение*
Debuger	Открыть окно Debug Control для управление пошаговым выполнением программы. Для выполнения текущей команды программы используется кнопка Over. В окне Черепахи можно наблюдать за выполнение программы по шагам.





Список некоторых цветов Черепахи[6]

Цвет	Название	Код	Цвет	Название	Код
	white	255, 255, 255		black	0, 0, 0
	lightgray	211, 211,211		darkgray	169,169,169
	silver	192,192,192		gray	128,128,128
	aliceblue	240,248,255		royalblue	65,105,225
	blue	0,0,255		navy	0,0,128
	lightblue	173,216,230		deepskyblue	0,191,255
	lightskyblue	135,206,250		skyblue	135,206,235
	cyan	0,255,255		darkcyan	0,139,139
	teal	0,128,128		turquoise	64,224,208
	aquamarine	127,255,212		seagreen	46,139,87
	lightgreen	144,238,144		green	0,128,0
	lime	0,255,0		darkgreen	0,100,0
	olive	128,128,0		ivory	255,255,240
	lightyellow	255,255,224		yellow	255,255,0
	gold	255,215,0		peru	205,133,63
	orange	255,165,0		chocolate	210,105,30
	brown	65,42,42		maroon	128,0,0
	coral	255,127,80		darkred	139,0,0
	tomato	255,99,71		red	255,0,0
	salmon	250,128,114		rosybrown	188,143,143
	pink	255,192,203		deeppink	255,20,147
	violet	139,0,255		purple	128,0,128
	magenta	255,0,255		blueviolet	138,43,226
	slateblue	106,90,205		indigo	75,0,130

[6] Другие цвета можно посмотреть, например, здесь - https://barzunov.ru/python/turtle/color/colors-list/