Глава 6
АЛГОРИТМЫ И ИСПОЛНИТЕЛИ

§ 16. Понятие алгоритма и исполнителя

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

Евклид

Аль-Хорезми

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

Пример 16.1. Звонок маме по мобильному телефону:

1. Открыть список контактов.
2. Выбрать нужный номер.
3. Нажать кнопку «Звонок».
4. Дождаться соединения.

Данный алгоритм состоит из 4 команд.

Пример 16.2. Написание поздравительной открытки:

1. Открыть графический редактор Paint.
2. Нарисовать открытку.
3. Распечатать открытку.

Данный алгоритм состоит из 3 команд.

16.1. Понятие алгоритма

В повседневной жизни нам приходится решать много задач, простых и сложных. Простыми могут быть звонок по мобильному телефону, чистка картофеля и др. Более сложно получить отметку 10 по информатике, испечь торт «Наполеон» и др.

Для решения любой задачи необходимо выполнить определенные действия (пример 16.1).

Понятная и конечная последовательность точных действий (команд), формальное выполнение которых позволяет получить решение поставленной задачи, называется **алгоритмом**.

Команда в алгоритме — указание на выполнение конкретного действия.

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

16.2. Понятие исполнителя алгоритма

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

Человек может быть исполнителем алгоритмов из примеров 16.1 и 16.2, алгоритма по сборке робота и др. Выполнять алгоритм может не только человек, но и робот, фотоаппарат и др. (пример 16.3).

Команды, которые понимает и может выполнить исполнитель, образуют **систему команд исполнителя.** В примере 16.4 приведена система команд исполнителя Стиральная машина-автомат. В зависимости от степени загрязнения и типа белья человек может задать разные режимы работы (алгоритмы) стиральной машины.

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

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

Пример 16.3. Примеры исполнителей.

Пример 16.4. Система команд исполнителя Стиральная машина-автомат:

- замачивание,
- стирка,
- полоскание,
- отжим,
- температура.

Пример 16.5. Примеры компьютерных исполнителей:

- Чертежник, с которым вы познакомитесь в § 18;
- Рыжий кот из программы Scratch.

Пример 16.6. Средой обитания исполнителя алгоритма 16.1 может быть только та среда, в которой используются мобильные телефоны. Данный алгоритм невозможно было бы выполнить 30 лет назад или при отсутствии сети.

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	17
2	$17 \cdot 2 = 34$
3	$34 + 10 = 44$
4	$44 : 2 = 22$
5	$22 - 17 = 5$
6	5

Пример 16.8. Выполнение алгоритма для исполнителя Кисть.

Алгоритм:

Выполняя команды алгоритма, получим изображение:

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

Исполнитель Шестиклассник (среда обитания — 6-й класс) умеет:

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

Ему предлагается выполнить алгоритм:

1. Задумать некоторое натуральное число.
2. Умножить задуманное число на 2.
3. К полученному 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:

- 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. Для этого требуется:

1. Определить максимальную температуру воздуха за сутки.
2. Определить минимальную температуру воздуха за сутки.
3. Найти разность между максимальным и минимальным значениями температур (пример 16.9).

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

Алгоритм определения азимута может быть таким:

1. Совместить окрашенный конец 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°) (пример 16.10).

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

Время наблюдения	Температура, °С
6.00	+8
12.00	+15
18.00	+14
24.00	+6

Максимальный результат +15 °С, минимальный — +6 °С. Амплитуда температур воздуха: 15 °С – 6 °С = 9 °С. 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 дерево равен 40°; азимут на вышку сотовой связи равен 140°; азимут на мельницу равен 220°; азимут на дом равен 320°.

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?

Упражнения

1 Приведите примеры алгоритмов из повседневной жизни и учебной деятельности.

2 Измените алгоритм из примера 16.1 (с. 112) для телефона с функцией голосового управления.

3 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 справа.
6. Запись ряда всех натуральных чисел.

4* Решите методом подбора задачу аль-Хорезми: «Я к трети числа прибавил единицу и к четверти числа прибавил единицу. Перемножив эти числа, получил 20. Какое число я взял?»

5 Приведите примеры исполнителей алгоритмов.

6 Напишите систему команд одного из исполнителей примера 16.3 (с. 113).

7 Выполните алгоритм из примера 16.7 (с. 114) несколько раз для разных чисел. Сравните полученные результаты. Сделайте выводы.

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

9 Напишите алгоритм морфологического разбора прилагательного. Выполните этот алгоритм для прилагательного *цифровом* из предложения *Современный человек живет в цифровом мире.*

10* 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.

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

1. Нажать оранжевую кнопку с надписью «Начать игру».
2. Выбрать «Играть без регистрации».
3. Выбрать игровое место «Скамейка».
4. Играть 15 мин.
5. Выйти из игры.

Для победы нужно набрать наибольшее количество баллов и подружиться со всеми персонажами.

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. Жарить на разогретой сковороде.

[1] http://mir.pravo.by/welcome (дата доступа 28.12.2017).

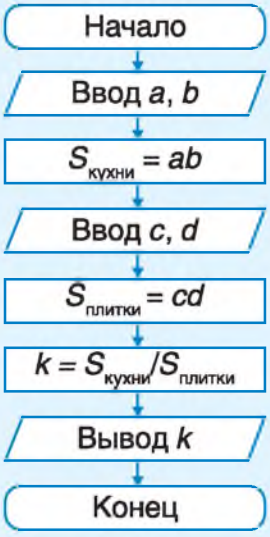
Пример 17.3. Ремонт на кухне.

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**:



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

Блоки начала и окончания алгоритма: Начало, Конец.

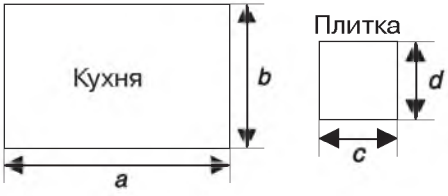
Блок для записи выполняемых команд алгоритма: Команда.

Блоки для ввода исходных данных и вывода полученных результатов: Ввод, Вывод.

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 записать алгоритм определения минимального количества плиток, необходимых для ремонта.



Словесное описание и блок-схема алгоритма, позволяющего определить необходимое количество плиток для ремонта, 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. $k = 14{,}08/0{,}1089 \approx 129{,}29$ (пл.).

Ответ: минимальное количество плиток для ремонта кухни — 130.

В примере 17.4 показана блок-схема алгоритма 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 текстовый фрагмент в буфер обмена).

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

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

Графический способ записи алгоритма:

Пример 17.5. Алгоритм определения особенностей периода каменного века.

Словесное описание:

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

1. Какие способы записи алгоритмов вам известны?
2. Какой способ записи называют словесным?
3. Что такое графическая запись алгоритма?
4. Какой способ записи алгоритма называют программным?

Упражнения

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

2 Приведите графическую запись алгоритмов решения примеров 17.1 и 17.2 (с. 117).

3 Составьте алгоритм для выполнения синтаксического разбора простого предложения.

4 В курсе истории вы познакомились с этапами работы над историческими источниками. 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, а также длины забора, который потребуется для ограждения участка. Выполните 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* Запишите алгоритм, позволяющий определить толщину листа бумаги учебного пособия «Информатика, 6».

9* 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».

10* 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 PascalABC.NET

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

Среда программирования — комплекс программ, используемых при разработке других программ.

При запуске среды программирования открывается одно окно, в котором можно выполнить весь процесс разработки: ввести текст программы, отредактировать его, выполнить программу и т. д. В среду PascalABC.NET входят: редактор текстов; справочная система; исполнитель Чертежник и др. Для создания своей программы нужно запустить PascalABC.NET (пример 18.1). Элементы окна среды (пример 18.2) рассмотрены в *Приложении* (с. 165).

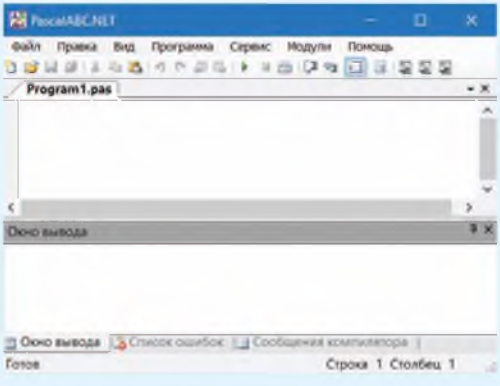
В окне PascalABC.NET можно набирать и редактировать текст программы. Сохранять и открывать файлы с текстами программ можно так же, как в текстовом и графическом редакторах. При необходимости можно обратиться к справочной системе через меню **Помощь** (пример 18.3).

Среда **PascalABC.NET** — совместная разработка российских и немецких программистов (2009 г.).

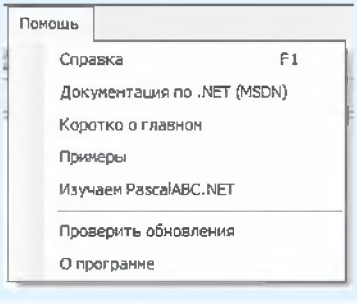
Пример 18.1. Запуск среды программирования PascalABC.NET.

Для запуска программы PascalABC.NET нужно использовать значок  .

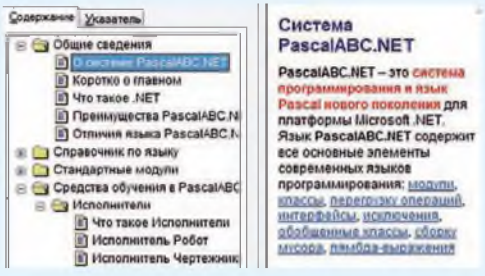
Пример 18.2. Окно программы PascalABC.NET.



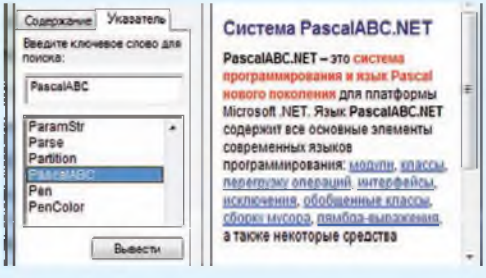
Пример 18.3. Меню справочной системы.



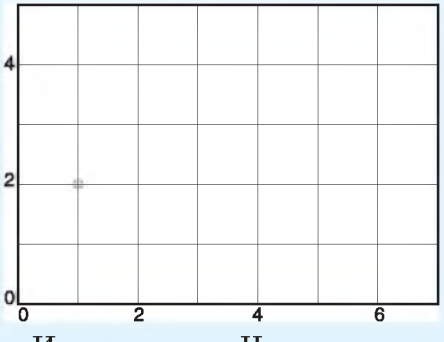
Пример 18.4. Вкладка **Содержание** окна **Справка**.



Пример 18.5. Содержание вкладки **Указатель** окна **Справка**.



Пример 18.6. Среда обитания исполнителя Чертежник.



Исполнитель Чертежник в среде обитания изображается маленьким квадратом, как на рисунке в точке (1, 2). Если перо опущено, размеры квадрата становятся меньше, как на рисунке в точке (5, 4).

Окно **Справка** содержит две вкладки: **Содержание** и **Указатель**. Для чтения нужной информации на вкладке **Содержание** следует выполнить щелчок мышью на названии раздела. Информация отобразится в правом окне (пример 18.4).

Вкладка **Указатель** позволяет осуществить поиск справочной информации по ключевому слову. Слово нужно ввести в специальную строку или выбрать из предложенного списка (пример 18.5).

18.2. Компьютерный исполнитель Чертежник

Исполнитель Чертежник предназначен для построения рисунков и чертежей на координатной плоскости. Он имеет перо, которое может поднимать, опускать, перемещать. При перемещении опущенного пера за ним остается след.

Средой обитания исполнителя Чертежник является часть координатной плоскости (пример 18.6). На уроках математики вы узнали, что координатная плоскость задается системой координат на плоскости. Это позволяет задать координаты любой точки на плоскости.

Исходное положение пера исполнителя Чертежник: поднято и находится над точкой (0, 0) —

началом координат. После выполнения программы перо должно быть также поднято.

Команды исполнителя Чертежник, с помощью которых можно записать программы для него, представлены в таблице.

Команда	Действие
`Field(n,m)`	Создать поле размером $n \times m$
`ToPoint(x,y)`	Переместить перо Чертежника в точку (x, y)
`PenUp`	Поднять перо Чертежника
`PenDown`	Опустить перо Чертежника

Здесь n, m — натуральные числа, x, y — целые неотрицательные числа.

(Рассмотрите пример 18.7.)

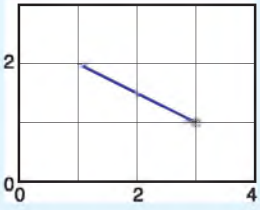
При работе со средой программирования можно использовать встроенный задачник с проверяемыми заданиями (пример 18.8). Для вызова проверяемого задания для исполнителя Чертежник используется следующий шаблон программы:

Вызов исполнителя Чертежник

```
uses Drawman;
begin
 Task('имя задания');  ← Вызов задания
 ...  ← Команды
end.
```

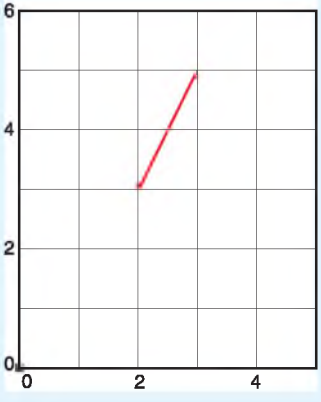
Пример 18.7. Пример использования команд `ToPoint`, `PenDown` и `PenUp`.

Команды для построения изображения отрезка, соединяющего точки (1, 2) и (3, 1):




```
ToPoint(1,2);
PenDown;
ToPoint(3,1);
PenUp;
```

Пример 18.8. Окно выполнения задания а1.



При вызове задания открывается окно исполнителя Чертежник. В этом окне отображается условие задачи и на координатной плоскости — требуемый результат. Цвет отрезка — красный. При выполнении программы отрезки изображаются синим цветом.

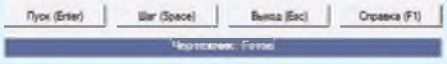
Пример 18.9. Алгоритм выполнения задания а1. Словесное описание:

1. Переместить перо в точку (2, 3).
2. Опустить перо.
3. Переместить перо в точку (3, 5).
4. Поднять перо.
5. Переместить перо в точку (0, 0).

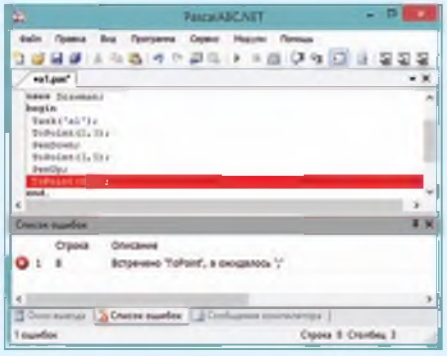
Пример 18.10. Программа выполнения задания а1.

```
Program1.pas*
uses Drawman;
begin
  Task('a1');
  ToPoint(2,3);
  PenDown;
  ToPoint(3,5);
  PenUp;
  ToPoint(0,0);
end.
```

Пример 18.11. Меню внизу окна исполнителя Чертежник.



Пример 18.12. Реакция PascalABC.NET на пропущенный символ «;» (точка с запятой).



Программа состоит из отдельных команд. В одной строке можно записывать несколько команд, которые отделяются друг от друга символом «точка с запятой» (;). Программа заканчивается символом «точка» (.). Команды алгоритма размещаются в теле программы между словами **begin** ... **end**.

Для решения задачи с помощью исполнителя Чертежник нужно:

1. Составить алгоритм решения задачи (пример 18.9).
2. Записать алгоритм в виде программы в окне текстового редактора среды программирования PascalABC.NET (пример 18.10).
3. Выполнить программу: **Программа** → **Выполнить** (можно нажать клавишу F9 или кнопку ▶ на панели инструментов).

С помощью меню в нижней части окна исполнителя Чертежник (пример 18.11) или соответствующих клавиш на клавиатуре можно выполнить программу целиком, по шагам, выйти из окна исполнителя, получить справку.

При выполнении программы в среде PascalABC.NET могут возникать ошибки. Ошибка может оказаться на месте нахождения курсора или в предыдущей строке. При этом в окне вывода результатов выводится соответствующее сообщение (пример 18.12).

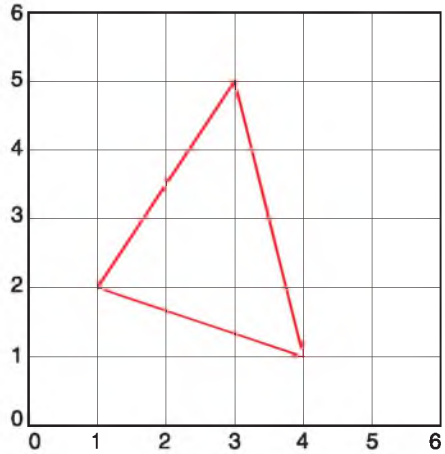
1. Что такое среда программирования?
2. В какой среде программирования размещается компьютерный исполнитель Чертежник?
3. Как получить справочную информацию об исполнителе Чертежник?
4. Как получить справочную информацию о команде `ToPoint`?
5. Для чего предназначен исполнитель Чертежник?

Упражнения

1 С помощью справочной системы среды программирования PascalABC.NET получите справку о команде `Field`.

2 Запишите в окне редактора среды программирования PascalABC.NET текст нижеприведенной программы и определите результат ее выполнения.

```
uses Drawman;
begin
   Field(4,4);
   ToPoint(1,1);
   PenDown;
   ToPoint(2,1);  ToPoint(2,3);
   ToPoint(3,3);  ToPoint(3,2);
   ToPoint(1,2);  ToPoint(1,1);
   PenUp;
end.
```



3 Запишите команды для примера 18.7 (с. 123) при условии, что Чертежник начинает перемещение от точки (3, 1) и смещается в точку (1, 2).

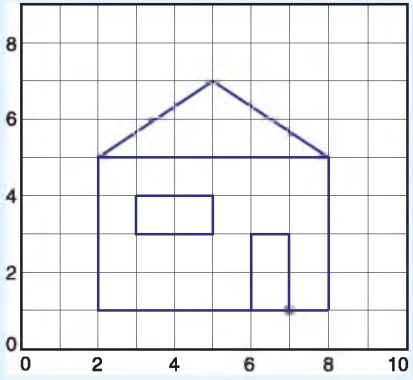
4* Составьте программу для построения изображения треугольника. Проверьте правильность выполнения упражнения на странице https://goo.gl/vCBL9U (зайдите гостем).

5 Запишите команды для построения изображения произвольного: а) квадрата; б) прямоугольника; в) прямоугольного треугольника; г) равнобедренного треугольника.

6 Запишите команды для построения изображения цифры 1.

§ 19. Изучение и изменение готовых программ

Пример 19.1. Изображение домика.



Программа:

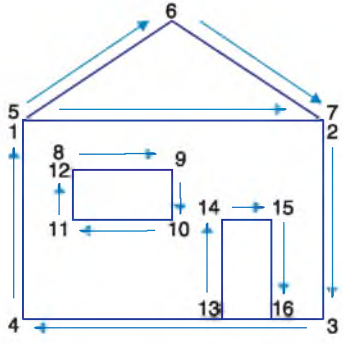
```
uses Drawman;
begin
  Field(10,9);
  ToPoint(2,5); PenDown;
  ToPoint(8,5);
  ToPoint(8,1);
  ToPoint(2,1);
  ToPoint(2,5);
  ToPoint(5,7);
  ToPoint(8,5); PenUp;
  ToPoint(3,4);
  PenDown;
  ToPoint(5,4);
  ToPoint(5,3);
  ToPoint(3,3);
  ToPoint(3,4);
  PenUp;
  ToPoint(6,1);
  PenDown;
  ToPoint(6,3);
  ToPoint(7,3);
  ToPoint(7,1);
  PenUp;
end.
```

Изображение окна

Изображение двери

Запишем алгоритм построения изображения домика для исполнителя Чертежник.

Выберем следующий алгоритм построения изображения:



1. Создать поле Чертежника размером 10 × 9.
2. Сместиться в точку (2, 5).
3. Опустить перо.
4. Сместиться в точку (8, 5).
5. Сместиться в точку (8, 1).
6. Сместиться в точку (2, 1).
7. Сместиться в точку (2, 5).
8. Сместиться в точку (5, 7).
9. Сместиться в точку (8, 5).
10. Поднять перо.
11. Сместиться в точку (3, 4).
12. Опустить перо и получить изображение окна.
13. Поднять перо.
14. Сместиться в точку (6, 1).
15. Опустить перо и получить изображение двери.
16. Поднять перо.

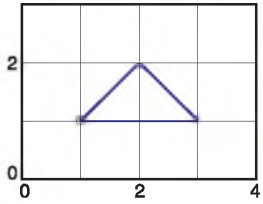
Готовое изображение домика и программа для его построения представлены в примере 19.1.

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

Рассмотрим пример 19.2. В нем требуется изменить изображение домика, представленного в примере 19.1.

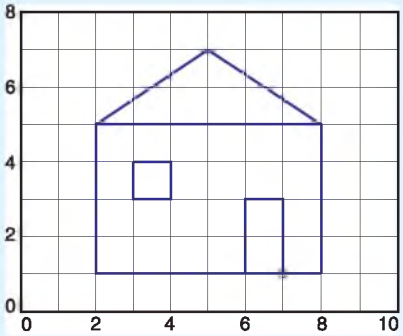
При сравнении нового изображения и изображения из примера 19.1 обнаружим два отличия: первое — в размерах поля Чертежника, второе — в форме окон. В целом рисунки похожи. Значит, для создания нового домика можно использовать программу из примера 19.1. Внесем изменения в текст программы в окне редактора среды программирования PascalABC.NET, используя правила редактирования текста.

Приведем такой пример. Нужно записать программу построения изображения треугольника:



Мы видим, что изображение похоже на крышу домика из примеров 19.1 и 19.2, но с другими

Пример 19.2. Программа построения измененного изображения.




```
uses Drawman;
begin
  Field(10,8);
  ToPoint(2,5);
  PenDown;
  ToPoint(8,5);
  ToPoint(8,1);
  ToPoint(2,1);
  ToPoint(2,5);
  ToPoint(5,7);
  ToPoint(8,5);
  PenUp;
  ToPoint(3,4);
  PenDown;
  ToPoint(5,4);
  ToPoint(5,3);
  ToPoint(3,3);
  ToPoint(3,4);
  PenUp;
  ToPoint(6,1);
  PenDown;
  ToPoint(6,3);
  ToPoint(7,3);
  ToPoint(7,1);
  PenUp;
end.
```

Пример 19.3. Программа:

```
uses Drawman;
begin
  Field(4,3);
  ToPoint(1,1); PenDown;
  ToPoint(2,2); ToPoint(3,1);
  ToPoint(1,1); PenUp;
end.
```

Пример 19.4. Фрагмент программы из примера 19.1 с комментарием.

```
  //окно
  ToPoint(3,4); PenDown;
  ToPoint(5,4);
  ToPoint(5,3); ToPoint(3,3);
  ToPoint(3,4); PenUp;
```

Пример 19.5. Пояснительный рисунок к команде `OnVector(a,b)`.

Пример 19.6. Программа построения изображения из примера 19.3 с помощью команды `OnVector`.

```
uses Drawman;
begin
  Field(4,3);
  ToPoint(1,1); PenDown;
  OnVector(1,1); OnVector(1,-1);
  OnVector(-2,0); PenUp;
end.
```

размерами поля Чертежника (4×3) и координатами точек. Покажем разницу в координатах:

Старые координаты	Новые координаты
(2, 5)	(1, 1)
(5, 7)	(2, 2)
(8, 5)	(3, 1)

Вставим новые координаты в уже известную нам программу для изображения треугольника (пример 19.3). Таким образом, известные программы можно использовать для построения новых изображений.

Чтобы сделать программы более понятными, их снабжают пояснительными текстами — комментариями. Комментарий начинается с символов // (пример 19.4).

Познакомимся еще с одной командой исполнителя Чертежник.

`OnVector(a,b)` — переместить перо Чертежника на вектор (a, b), т. е. на a единиц вдоль оси x и на b — вдоль оси y.

При перемещении вправо $a > 0$, вверх $b > 0$. При перемещении влево $a < 0$, вниз $b < 0$.

На рисунке в примере 19.5 показано, как найти значения измененных координат для построения изображения прямоугольного треугольника.

Соответствующие команды для построения этого изображения с помощью исполнителя Чертежник:

```
ToPoint(1,2);
PenDown;
OnVector(2,0);
OnVector(0,3);
OnVector(-2,-3);
```

Запишем программу решения примера 19.3 с помощью команды OnVector (пример 19.6). Эту программу можно использовать для построения похожего изображения (пример 19.7).

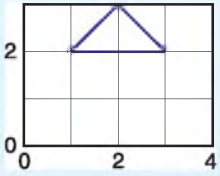
Запишем программу построения изображения цифры 3 в почтовом индексе с помощью исполнителя Чертежник (по образцу, представленному в примере 19.8).

Алгоритм построения изображения:

1. Задать поле для исполнителя Чертежник размером 3 × 4.
2. Сместиться в точку (1, 3).
3. Опустить перо.
4. Изобразить цифру, двигаясь по отрезкам 1—2, 2—3, 3—4, 4—5.
5. Поднять перо.

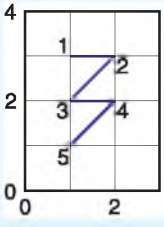
В некоторых изображениях повторяются одинаковые фрагменты. Для создания программ построения таких изображений можно скопировать повторяющийся

Пример 19.7. Модификация программы из примера 19.6.




```
uses Drawman;
begin
  Field(4,3);
  ToPoint(1,2); PenDown;
  OnVector(1,1); OnVector(1,-1);
  OnVector(-2,0); PenUp;
end.
```

Пример 19.8. Программа построения изображения цифры 3.



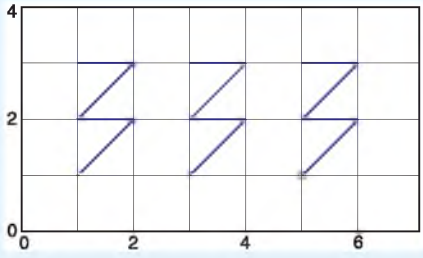
1. С использованием команды ToPoint:

```
uses Drawman;
begin
  Field(3,4);
  ToPoint(1,3); PenDown;
  ToPoint(2,3); // 1-2
  ToPoint(1,2); // 2-3
  ToPoint(2,2); // 3-4
  ToPoint(1,1); // 4-5
  PenUp;
end.
```

2. С использованием команды OnVector:

```
uses Drawman;
begin
  Field(3,4);
  ToPoint(1,3); PenDown;
  OnVector(1,0); // 1-2
  OnVector(-1,-1); // 2-3
  OnVector(1,0); // 3-4
  OnVector(-1,-1); // 4-5
  PenUp;
end.
```

Пример 19.9. Программа построения изображения из трех одинаковых цифр.




```
  uses Drawman;
begin
  Field(7,4);
  //первая тройка слева
  ToPoint(1,3);
  PenDown;
  OnVector(1,0);
  OnVector(-1,-1);
  OnVector(1,0);
  OnVector(-1,-1);
  PenUp;
  //средняя тройка
  ToPoint(3,3);
  PenDown;
  OnVector(1,0);
  OnVector(-1,-1);
  OnVector(1,0);
  OnVector(-1,-1);
  PenUp;
  //тройка справа
  ToPoint(5,3);
  PenDown;
  OnVector(1,0);
  OnVector(-1,-1);
  OnVector(1,0);
  OnVector(-1,-1);
  PenUp;
end.
```

фрагмент программы и использовать его нужное число раз.

Так, в примере 19.9 требуется записать программу для построения изображения, состоящего из трех цифр «3».

Мы видим, что программу построения изображения можно составить на основе программы из примера 19.8. Изображение первой цифры начинается от верхней точки слева, ее координаты (1, 3). Координаты такой же точки для второй цифры (3, 3), для третьей цифры (5, 3).

Таким образом, для создания изображения из трех цифр «3» нужно скопировать в тексте программы примера 19.8 следующий фрагмент:

```
ToPoint(1,3);
PenDown;
OnVector(1,0);
OnVector(-1,-1);
OnVector(1,0);
OnVector(-1,-1);
PenUp;
```

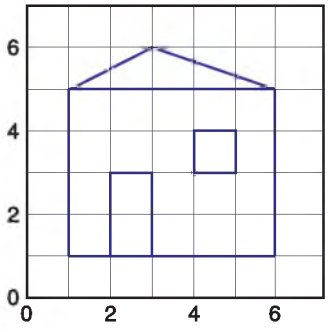
Затем следует вставить скопированный фрагмент нужное количество раз и внести изменения.

1. Как получить справочную информацию о командах `ToPoint`, `OnVector`?
2. В чем отличие между командами Чертежника `ToPoint` и `OnVector`?

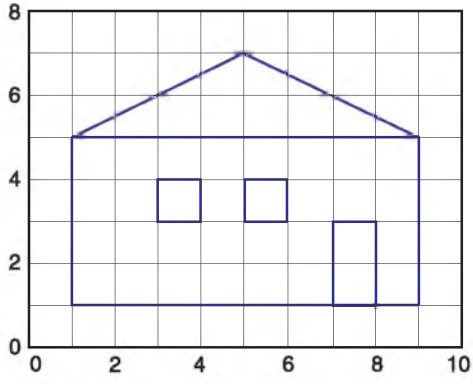
Упражнения

1 Измените программу из примера 19.1 (с. 126) для построения изображений:

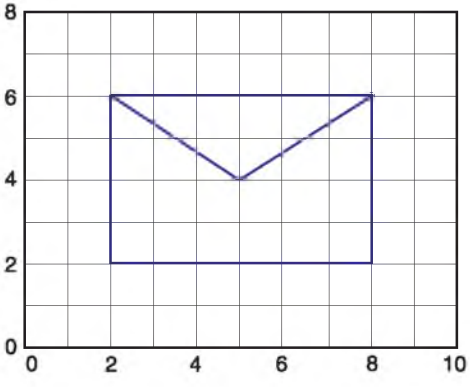
а



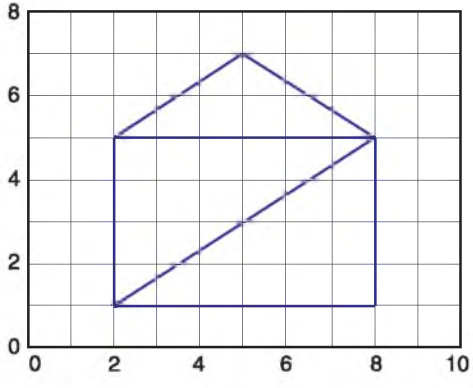
б



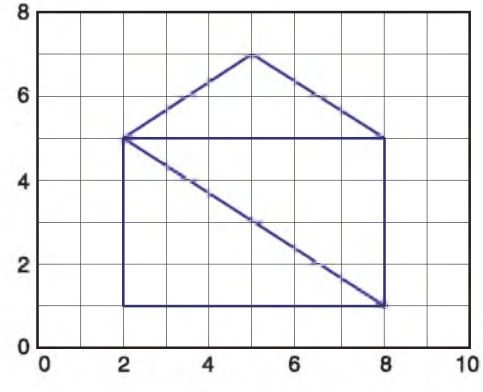
в



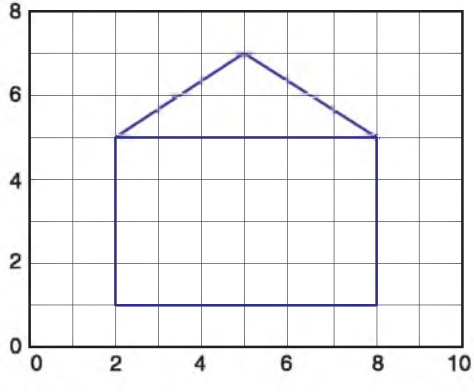
г



д

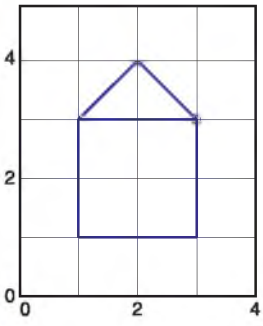


е

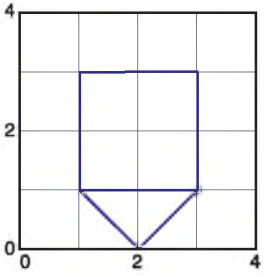


2 Измените программу из примера 19.6 (с. 128) для построения изображений:

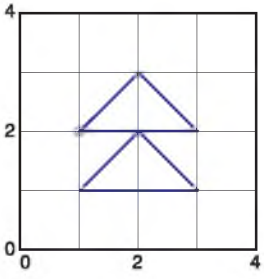
а



б



в



3 Откройте файл с текстом программы и определите результат ее выполнения.

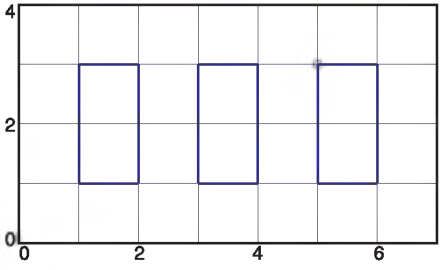
```
uses Drawman;
begin
   Field(10,5);
   //буква Б
   OnVector(2,4);  PenDown;
   OnVector(-1,0);  OnVector(0,-3);
   OnVector(2,0);  OnVector(0,2);
   OnVector(-2,0);  PenUp;
   //буква Г
   OnVector(5,1);  PenDown;
   OnVector(-2,0);  OnVector(0,-3);
   PenUp;
   //буква У
   OnVector(3,3);  PenDown;
   OnVector(2,-2);  PenUp;
   OnVector(0,2);  PenDown;
   OnVector(0,-3);  OnVector(-1,0);
   PenUp;
end.
```

Измените программу для построения изображения:

4 Откройте файл с текстом программы и определите результат ее выполнения.

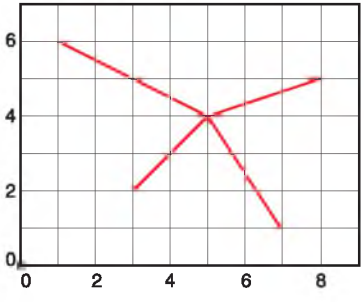
```
uses Drawman;
begin
    Field(7,4); ToPoint(1,3); PenDown;
    OnVector(1,0); OnVector(0,-2);
    OnVector(-1,0); OnVector(0,2);
    PenUp;
end.
```

Измените программу для построения изображения:



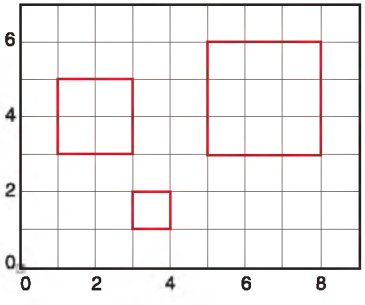
5 Составьте программы для выполнения проверяемых заданий.

Задание а2

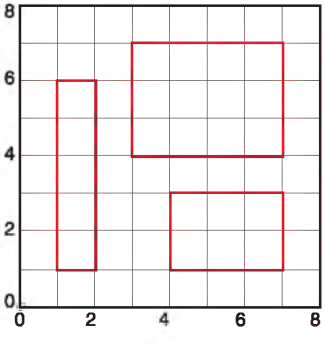


Задание а3

Задание а4

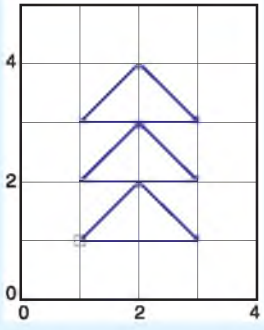


Задание а5



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

Пример 20.1. Рисование елочки.



Программа построения изображения:

```
uses Drawman;
begin
  Field(4,5);
  ToPoint(1,3);
  PenDown;
  OnVector(1,1);
  OnVector(1,-1);
  OnVector(-2,0);
  PenUp;
```

Верхний треугольник

```
  ToPoint(1,2);
  PenDown;
  OnVector(1,1);
  OnVector(1,-1);
  OnVector(-2,0);
  PenUp;
```

Средний треугольник

```
  ToPoint(1,1);
  PenDown;
  OnVector(1,1);
  OnVector(1,-1);
  OnVector(-2,0);
  PenUp;
end.
```

Нижний треугольник

Как видно из предыдущего параграфа, исполнителю Чертежник 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 алгоритм можно использовать необходимое число раз, обращаясь к его названию (имени). Для обращения к вспомогательному алгоритму в блок-схемах используется блок:

Вспомогательный алгоритм в среде PascalABC.NET записывается в виде процедуры:

```
procedure имя процедуры;
begin
  ...
  ...
end;
```

Заголовок процедуры
Начало процедуры
Команды (тело процедуры)
Конец процедуры

В программе процедура записывается ниже команды `uses Drawman;`

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

В примере 19.6 (с. 128) мы записали команды для рисования треугольника. Этот треугольник может быть элементом елочки. Будем изображать элементы елочки начиная с точек (1,3), (1,2), (1,1). Наши действия:

1. Сместиться в точку (1,3).
2. Нарисовать треугольник.
3. Сместиться в точку (1,2).
4. Нарисовать треугольник.
5. Сместиться в точку (1,1).
6. Нарисовать треугольник.

Изображение елочки и программа его построения представлены в примере 20.1.

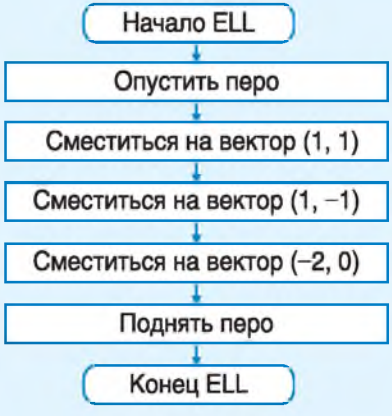
Мы видим, что в каждом выделенном фрагменте программы есть повторяющиеся команды:

```
PenDown;
OnVector(1,1);
OnVector(1,-1);
OnVector(-2,0);
PenUp;
```

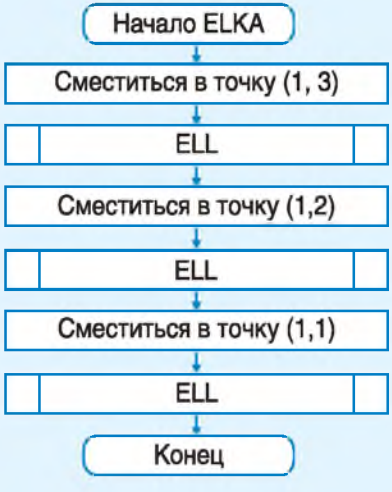
Опишем эту последовательность команд в виде вспомогательного алгоритма ELL, который будет использоваться в неизменяемом виде несколько раз (в данном случае три раза). В примере 20.2

Пример 20.2. Блок-схема алгоритма рисования елочки включает:

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



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



Пример 20.3. Программа построения елочки:

```
uses Drawman;
procedure ELL;
//процедура для рисования
//одного элемента елочки —
//треугольника
begin
  PenDown;
  OnVector(1,1);
  OnVector(1,-1);
  OnVector(-2,0);
  PenUp;
end;
begin
  Field(4,5);
  ToPoint(1,3);
  ELL;
  ToPoint(1,2);
  ELL;
  ToPoint(1,1);
  ELL;
end.
```

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 и др.).

представлены блок-схемы алгоритмов для рисования елочки. Пример 20.3 содержит программу рисования данного изображения.

Вспомогательный алгоритм решает некоторую подзадачу основной задачи (примеры 20.4 и 20.5).

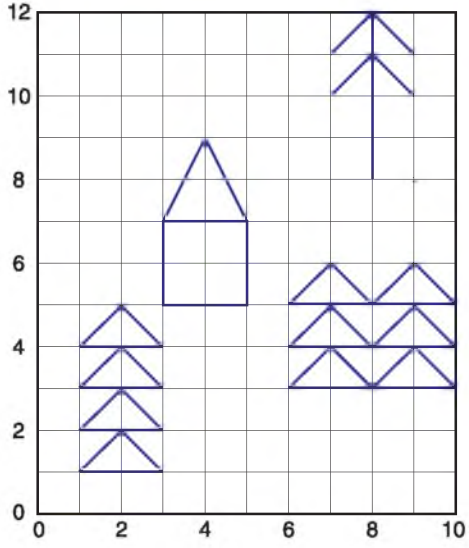
При решении реальных задач над проектом могут работать несколько человек. Каждый выполняет свою часть работы и оформляет ее как отдельный вспомогательный алгоритм. Важно, чтобы вспомогательные алгоритмы, написанные разными людьми, правильно выполнялись в одном проекте. Для этого устанавливаются определенные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

Для исполнителя Чертежник такой договоренностью условимся считать следующее правило:

в исходном положении Чертежника перо поднято, и в таком же положении должно быть перо после выполнения программы.

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.



Пятеро шестиклассников могут распределить работу между собой следующим образом:

1. Вспомогательный алгоритм построения треугольника.

2. Вспомогательные алгоритмы для большой и маленькой елей, основанные на вспомогательном алгоритме построения треугольника.

3. Вспомогательный алгоритм рисования дома.

4. Вспомогательный алгоритм рисования сосны.

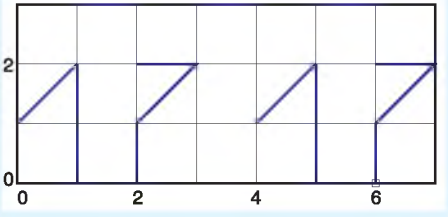
5. Основной алгоритм с размещением элементов «пейзажа» на поле Чертежника.

В примере 20.6 на с. 138 размещена программа рисования «пейзажа». Она содержит процедуры:

Пример 20.4. Программа построения изображения из примера 19.9 (с. 130).

```
uses Drawman;
procedure three;
begin
  PenDown;
  OnVector(1,0); OnVector(-1,-1);
  OnVector(1,0); OnVector(-1,-1);
  PenUp;
end;
begin
  Field(7,4);
  ToPoint(1,3); three;
  ToPoint(3,3); three;
  ToPoint(5,3); three;
end.
```

Пример 20.5. Программа построения изображения из цифр.




```
uses Drawman;
procedure _17;
begin
  PenDown;
  OnVector(1,1);// 1
  OnVector(0,-2);PenUp;
  OnVector(1,2);
  PenDown;
  OnVector(1,0);// -
  OnVector(-1,-1);// /
  OnVector(0,-1); // |
  PenUp;
end;
begin
  Field(7,3);
  ToPoint(0,1); _17;
  ToPoint(4,1); _17;
end.
```

Пример 20.6. Программа:

```
uses Drawman;
procedure treug;
begin
//треугольник
  PenDown; OnVector(1,1);
  OnVector(1,-1); OnVector(-2,0);
  PenUp; OnVector(2,0);
end;
procedure b_el;
begin
//большая ель
  treug; OnVector(-2,1);
  treug; OnVector(-2,1);
  treug; OnVector(-2,1);
  treug; OnVector(0,-3);
end;
procedure m_el;
begin
//маленькая ель
  treug; OnVector(-2,1);
  treug; OnVector(-2,1);
  treug; OnVector(0,-2);
end;
procedure dom;
begin
//дом
  PenDown; OnVector(2,0);
  OnVector(0,2); OnVector(-1,2);
  OnVector(-1,-2); OnVector(2,0);
  OnVector(-2,0); OnVector(0,-2);
  PenUp; OnVector(2,0);
end;
procedure sosna;
begin
//сосна
  OnVector(2,0); PenDown;
  OnVector(0,4); PenUp;
  OnVector(-1,-1); PenDown;
  OnVector(1,1); OnVector(1,-1);
  PenUp; OnVector(-2,-1);
  PenDown; OnVector(1,1);
  OnVector(1,-1); PenUp;
  OnVector(0,-2);
end;
begin
  Field(10,12); ToPoint(1,1); B_El;
  ToPoint(6,3); m_El; m_El;
  ToPoint(3,5); dom;
  ToPoint(6,8); Sosna;
end.
```

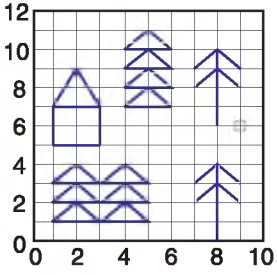
`treug` — для построения треугольника; `b_el` — для рисования большой ели; `m_el` — для рисования маленькой ели; `dom` — для рисования дома; `sosna` — для рисования сосны.

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 Чертежника, указав координаты нижнего левого угла объекта.

Например, сохраним все процедуры из примера 20.6, а в текст программы внесем изменения:

```
begin
  Field(10,12);
  ToPoint(4,7);  B_El;
  ToPoint(1,1);  m_El;  m_El;
  ToPoint(1,5);  dom;
  ToPoint(6,0);  Sosna;
  ToPoint(6,6);  Sosna;
end.
```

При запуске программы получим рисунок:

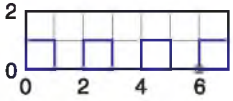


1. Какие алгоритмы называются вспомогательными?
2. Для чего нужны вспомогательные алгоритмы?

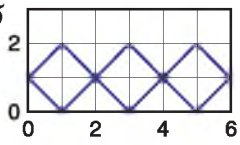
Упражнения

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

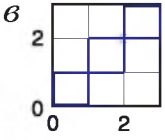
а



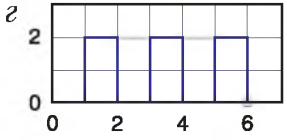
б



в

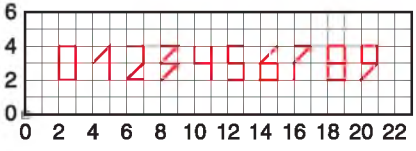


г

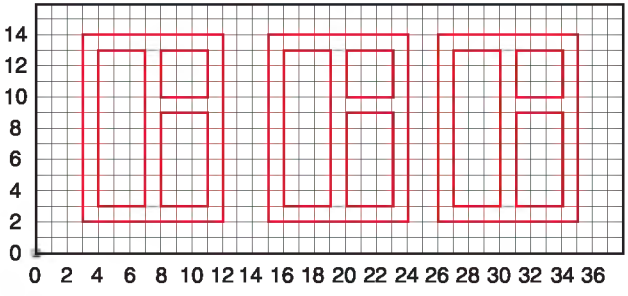


2 Запишите программы для выполнения проверяемых заданий:

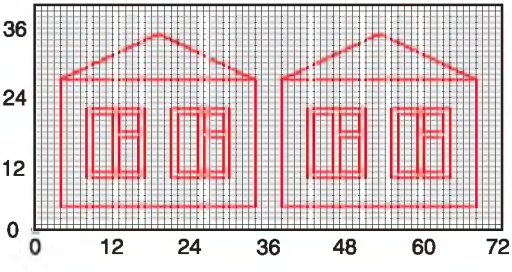
Задание р1



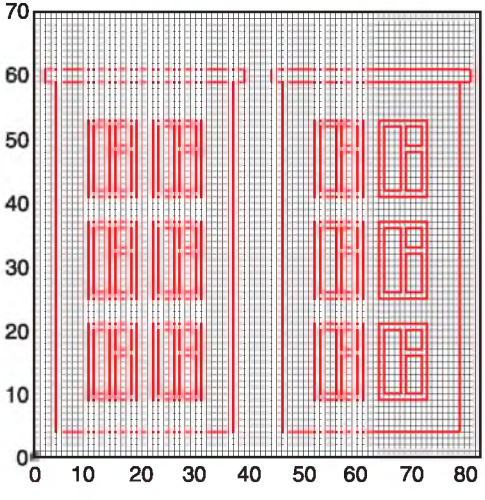
Задание р2



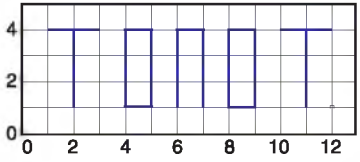
Задание р3

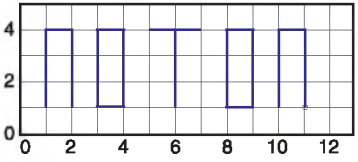


Задание р4

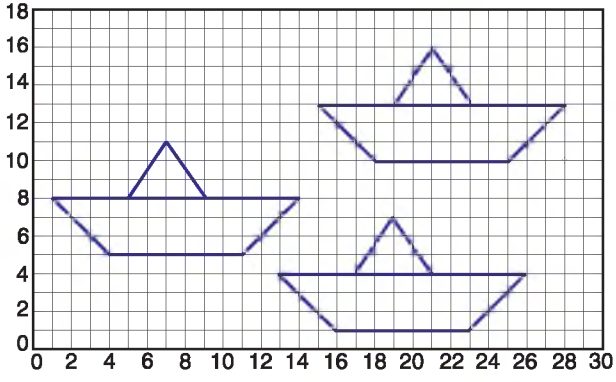


3 Запишите программы написания слов для исполнителя Чертежник. Используйте вспомогательные алгоритмы.





4 Составьте программу для построения изображения:



5* Используя составленные ранее программы для исполнителя Чертежник как вспомогательные, придумайте свой «пейзаж» и реализуйте его.

6 Используя составленные ранее программы для исполнителя Чертежник как вспомогательные, составьте программу для построения изображения: