

Тема 13 **Представление о программе. Классификация программ**

Изучив эту тему, вы узнаете:

- что такое Исполнитель и каково его назначение;
- что означают понятия программы и программирования на алгоритмических языках;
- в чем отличие процедурного и объектного подходов к программированию;
- как классифицируется программное обеспечение;
- как программное обеспечение соотносится с аппаратной частью.

13.1. Исполнитель алгоритма

Действия, описываемые в алгоритме, прежде всего должны быть понятны самому разработчику алгоритма. Только тогда алгоритм можно преобразовать в форму, понятную тому, кто будет его выполнять.

Поэтому разработка алгоритма практически всегда осуществляется в две стадии, как было сказано в п. 12.8. На первой стадии человек приблизительно описывает последовательность выполнения действий объектом, который будет претворять в жизнь заданную в алгоритме идею. Возможно, этим объектом будет сам разработчик. На этой стадии человек должен ясно представить себе, что же он хочет получить и каким образом. На следующей стадии алгоритм претерпевает некоторые изменения для того, чтобы в нем были учтены особенности среды, в которой предполагается выполнение этого алгоритма.

Объект, который будет выполнять разработанный человеком алгоритм, обычно называют *Исполнителем*. Его предназначение — точно выполнить предписания алгоритма, подчас не задумываясь о результатах и целях. Например, Исполнителем может быть:



- ◆ солдат в армии, который обязан беспрекословно выполнять приказы старших по званию чинов;
- ◆ собака, которая должна выполнять команды хозяина;
- ◆ животные в цирке, которые должны точно исполнять требования дрессировщика;
- ◆ робот, производящий измерения в космосе, выполняет команды, поступающие от космического центра;
- ◆ летчик, который должен точно выполнять распоряжения диспетчера аэропорта.

Во всех приведенных выше примерах объект, исполняющий действия алгоритма, не обязан:

- ◆ понимать цели и методы достижения этой цели;
- ◆ пропускать действия или менять их порядок по своему усмотрению;
- ◆ искать какую-то замену, если действие выполнить невозможно.



Исполнитель — объект, который выполняет алгоритм.



Идеальными Исполнителями являются машины, роботы, компьютеры. Они в состоянии выполнять указанные команды, не обсуждая их целесообразность. Человек тоже может поставить себя в положение Исполнителя алгоритма, хотя бы для провер-

ки его правильности. При этом человек формально, не стараясь понять поставленную задачу, выполняет команду за командой.

Исполнитель способен выполнять только ограниченное количество команд. Поэтому алгоритм, переписанный на второй стадии под конкретного Исполнителя, должен еще раз пройти дополнительное преобразование. Алгоритм дорабатывается и детализируется так, чтобы в нем присутствовали только те команды и конструкции, которые может выполнить Исполнитель.

Так появляется *третья стадия*, на которой алгоритм должен быть представлен в форме, понятной Исполнителю. Исполнитель, как и любой объект, находится в определенной среде и может выполнять только допустимые в ней действия. Если Исполнитель встретит в алгоритме неизвестную ему команду, то выполнение алгоритма прекратится.

13.2. Понятие программы

На третьей стадии разработки алгоритма необходимо познакомиться с командами, доступными Исполнителю, и с правилами их записи. Так, игра в шахматы теряет всякий смысл, если Исполнитель не представляет себе правил поведения в среде «шахматное поле».

Алгоритм, представленный на понятном Исполнителю языке, называют *программой*. Обычно понятие программы связывают с компьютером, и тогда процесс создания программы называют *программированием* или *кодированием*. Программа должна быть составлена так, чтобы каждый блок компьютера выполнял задуманное человеком действие в соответствии с алгоритмом.



Программа — упорядоченная последовательность команд (инструкций), необходимых компьютеру для решения поставленной задачи.

Программирование (кодирование) — процесс составления программы для компьютера.

Для первых электронно-вычислительных машин программы записывались в виде последовательности элементарных операций. Это была очень трудоемкая и неэффективная работа. Для



исправления любой ошибки приходилось переделывать всю программу и снова записывать ее в память.

Поэтому впоследствии были разработаны специальные языки программирования, которые позволяют удобно и наглядно представить алгоритм

решения задачи. Первым алгоритмическим языком для создания компьютерных программ был Фортран. Очень скоро появились и другие языки: Алгол 58, Алгол 60, ПЛ, Бейсик, Паскаль и др. Каждый из них нес в себе какую-нибудь особую идею по более рациональному использованию ресурсов компьютера и усовершенствованию формы представления программы.

В настоящее время существует множество искусственных языков для составления программ. Однако так и не удалось создать идеального алгоритмического языка, который устроил бы всех, как не удалось создать и искусственный разговорный язык, который удовлетворил бы все страны и народы. Одним нравится программировать на Паскале, другим — на С, третьи, прекрасно овладев правилами составления программ на Бейсике, вполне довольны получающимися результатами. Алгоритм, представленный с помощью языка программирования, чем-то похож на математическую формулу.

Программы, так же как и алгоритмы, обладают свойствами *дискретности* и *детерминированности*. Верно составленные программы должны быть *конечны* и *правильны*. Хорошие программы обладают свойством *массовости*.

Программа хранится в памяти компьютера. При запуске программы компьютер выполняет команды в том порядке, в котором они записаны.

В каждом алгоритмическом языке программирования имеется набор встроенных команд — инструментов для выполнения разнообразных действий. В состав языка обязательно включены команды (операторы) для создания в программе циклов и разветвляющихся конструкций. Любой язык содержит правила для

разработки и применения вспомогательных программ, называемых процедурами.



Процедура — вспомогательная программа, которая вызывается из другой программы.

В течение жизни человек постоянно расширяет свой словарный запас. Услышав или прочитав новое слово, вы иногда не понимаете, о чем идет речь, пока не сопоставите это слово с уже знакомыми понятиями. Столкнувшись с новым явлением, мы стараемся подобрать для него название, иногда придумывая его, но обязательно объясняя с помощью уже известных слов. Таким образом увеличиваются наши знания.

Разрабатывая программу для компьютера, вы даете ей уникальное *имя*. Записывая в память программу под этим именем, вы расширяете словарный запас компьютера. Применяя специальные конструкции языка программирования, можно разработанную программу использовать в качестве процедуры при создании новых программ. Обращение к процедуре происходит по ее имени.

13.3. Подходы к созданию программы*

При разработке алгоритмов, а затем соответствующих им программ можно использовать разные подходы. Проиллюстрируем это на примерах создания алгоритмов, в которых применяется процедурный и объектный подходы.

При *процедурном подходе* в результате выполнения любого действия, описанного в алгоритме, должен быть получен промежуточный результат. Цепочка промежуточных результатов в конечном счете должна привести к ожидаемому конечному результату.

Покажем это на примере. Надо составить алгоритм вычисления на калькуляторе следующего выражения:

$$\frac{1}{\frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3}}.$$

Здесь числа, над которыми в калькуляторе будут совершаться действия, заменены именами переменных. Эти переменные могут принимать различные значения. Если вводимые числа или значение суммы в знаменателе окажется равным нулю, то действие алгоритма окажется невыполнимым. Следовательно, в алгоритм должна быть включена проверка условия «равно ли значение знаменателя нулю?» и предусмотрены действия для обхода такой ситуации. В результате алгоритм вычисления будет следующим:

1. Если $R1 = 0$, или $R2 = 0$, или $R3 = 0$, или $R4 = 0$, то выбрать другие числа. Если все числа не равны нулю, то перейти к п. 2.
2. Вычислить $1/R1$ и записать Результат1.
3. Вычислить $1/R2$ и записать Результат2.
4. Вычислить $1/R3$ и записать Результат3.
5. Вычислить $1/R4$ и записать Результат4.
6. Сложить Результат1, Результат2, Результат3 и Результат4 и запомнить сумму как Результат5.
7. Если Результат5 не равен нулю, то разделить 1 на Результат5, и полученный Результат6 будет искомым ответом.
8. Если Результат5 равен 0, то выбрать другие числа и перейти к п. 1 или прекратить вычисления.

При *объектном подходе* рассматривается класс объектов, их свойства, параметры, действия, среда. Для создания конкретного экземпляра класса объектов надо указать соответствующие значения параметров. Вы не должны подробно описывать, как создавать каждый элемент выбранных объектов. Обработка параметров конкретного объекта уже заключена в действиях алгоритма.



Рассмотрим пример, иллюстрирующий объектный подход.

Предположим, в альбоме имеются красочные иллюстрации, на которых изображены разнообразные цветы. Кроме того, приведены шаблоны для рисования частей каждого цветка: стебля, листа,

венчика цветка. Эти шаблоны рекомендуется вырезать, чтобы, используя их, можно было достаточно быстро рисовать существующие или выдуманные цветы.

Составим описание класса «Цветок» в виде таблицы 13.1 так, как это было показано в теме 7.

Таблица 13.1. Описание класса «Цветок»

Параметр		Действие
Название	Значение	
Название цветка	Роза, ромашка, тюльпан и др.	Выращивать Рисовать Вырезать из бумаги Лепить
Вид стебля	Стебель розы, стебель ромашки, стебель тюльпана и т. д.	
Вид листа	Лист розы, лист ромашки, лист тюльпана и т. д.	
Вид венчика цветка	Венчик розы, венчик ромашки, венчик тюльпана и т. д.	

Предположим, вы хотите нарисовать ромашку, используя шаблоны ее составных частей. Это означает, что вы выбираете из таблицы 13.1 действие «Рисовать».

Прежде всего надо создать необходимую для рисования среду, а именно: найти место на столе, взять бумагу, цветные карандаши, подготовить шаблоны. Затем вы определяете конкретные значения параметров объекта, который будете рисовать: название цветка — ромашка; вид стебля, листа и венчика цветка — определяются выбранным шаблоном. Далее составляете алгоритм рисования:



1. *Найти в книге изображение **ромашки**.*
2. *Нарисовать по шаблону стебель **ромашки**, задав нужный размер.*
3. *Нарисовать на стебле по шаблону несколько листов **ромашки**, задав им нужный размер.*
4. *Нарисовать в верхней части стебля по шаблону венчик цветка **ромашки**, задав нужный размер.*



По указанному алгоритму вы сможете нарисовать любое количество ромашек разного размера с разным количеством листочков.

Если вы решите рисовать другой объект, например розу, то алгоритм останется тем же. Изменяются только значения параметров: название объекта — роза; вид стебля, вид листа, вид венчика цветка — типовые для розы и определяются выбранным шаблоном. Обратите внимание на изменения в алгоритме:

1. *Найти в книге изображение **розы**.*
2. *Нарисовать по шаблону стебель **розы**, задав нужный размер.*
3. *Нарисовать на стебле по шаблону несколько листов **розы**, задав им нужный размер.*
4. *Нарисовать в верхней части стебля по шаблону венчик цветка **розы**, задав нужный размер.*



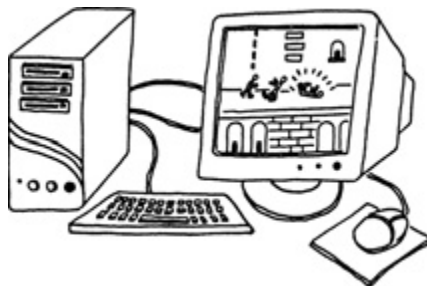
Допустим, вы решили вырезать из бумаги цветок розы, используя для этого готовые шаблоны. Тогда нужно немного изменить среду (условия) работы — найти ножницы, цветные листы бумаги, клей. В алгоритме действие «Рисовать» будет заменено на «Вырезать». Параметры останутся прежними:

1. *Найти в книге изображение **розы**.*
2. *Вырезать по шаблону стебель **розы**, задав нужный размер.*
3. *Вырезать по шаблону несколько листов **розы**, задав им нужный размер.*
4. *Вырезать по шаблону венчик цветка **розы**, задав нужный размер.*
5. *Склеить вырезанные части.*

Современные методы разработки алгоритмов основываются на понятиях объектов, их параметров и действий, среды. Приступая к созданию алгоритма, прежде всего ориентируются на некую среду существования объектов. В среде должны иметься инструменты для выполнения действий над объектами и изме-

нения их параметров. Алгоритм преобразования объектов содержит описание действий, выполняемых этими инструментами. Например, при рисовании ромашки такими инструментами являются: «Нарисовать по шаблону стебель», «Нарисовать на стебле по шаблону несколько листьев», «Нарисовать в верхней части стебля по шаблону чашечку цветка».

Объектный подход к разработке алгоритма часто используется при создании компьютерных игр. Алгоритм игры описывает действия героев, параметры которых можно задавать (вводить) в процессе выполнения алгоритма (игры). Благодаря этому в начале игры вы имеете возможность выбрать своего героя, обеспечить ему снаряжение и указать другие характеристики. В соответствии с введенными значениями компьютер отображает вашего героя на экране монитора.



13.4. Классификация программного обеспечения*

Совокупность используемых в компьютере программ принято называть программным обеспечением. Любая программа создает на компьютере определенную среду для работы и включает в себя инструментарий, с помощью которого вы имеете возможность создавать компьютерные объекты. Разнообразие сред определяется составом программного обеспечения компьютера. Приведем упрощенное определение программного обеспечения.



Программное обеспечение — совокупность всех используемых в компьютере программ.

В жизни все объекты можно сгруппировать по определенным признакам и составить для себя представление о том, где можно использовать того или иного представителя данной группы

(класса). То же самое можно сделать и по отношению к компьютерным программам.

Для того чтобы ясно понимать, где и какую программу вам следует использовать для преобразования информации и получения желаемого результата, необходимо иметь представление об имеющихся разновидностях программ. Все программное обеспечение принято разделять на три класса (рисунок 13.1): системное, прикладное, инструментальный программирования.



Рис. 13.1. Классификация программного обеспечения

Системное программное обеспечение

Этот класс программного обеспечения является необходимой принадлежностью компьютера, так как обеспечивает взаимодействие человека, всех устройств и программ компьютера.

Этот комплекс программ определяет на компьютере *системную среду* и правила работы в ней. Чем более совершеннее системное программное обеспечение, тем комфортнее мы чувствуем себя в системной среде.



Самой важной системной программой является *операционная система*, которая обычно хранится на жестком диске. При включении компьютера ее основная часть переписывается с жесткого диска во внутреннюю память и там находится на протяжении всего сеанса работы компьютера. Если вы включили компьютер и при