В. В. Трофимов, Т. А. Павловская

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

УЧЕБНИК ДЛЯ СПО

Под редакцией **В. В. Трофимова**

Рекомендовано Учебно-методическим отделом среднего профессионального образования в качестве учебника для студентов образовательных учреждений среднего профессионального образования

Книга доступна в электронной библиотечной системе biblio-online.ru

Москва ▪ Юрайт ▪ 2018

УДК 681.3(075.32)
ББК 32.81я723
Т76

***Авторы*:**

Трофимов Валерий Владимирович — доктор технических наук, профессор, заслуженный деятель науки Российской Федерации, заведующий кафедрой информатики факультета информатики и прикладной математики Института экономики Санкт-Петербургского государственного экономического университета, профессор кафедры информатики и прикладной математики — 1 факультета компьютерных технологий и управления Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики;

Павловская Татьяна Александровна — профессор, кандидат технических наук, работала в должности профессора на кафедре информатики Санкт-Петербургского государственного экономического университета.

***Рецензенты*:**

Песоцкая Е. В. — доктор экономических наук, профессор Санкт-Петербургского государственного экономического университета;

Стельмашонок Е. В. — доктор экономических наук, профессор Санкт-Петербургского государственного инженерно-экономического университета;

Гаспариан М. С. — доктор экономических наук, профессор.

Трофимов, В. В.

Т76 Основы алгоритмизации и программирования : учебник для СПО / В. В. Трофимов, Т. А. Павловская ; под ред. В. В. Трофимова. — М. : Издательство Юрайт, 2018. — 137 с. — (Серия : Профессиональное образование).

ISBN 978-5-534-07321-8

В учебнике, представляющем собой один из модулей дисциплины «Информатика», рассмотрены модели решения функциональных и вычислительных задач, алгоритмизация и программирование, языки программирования высокого уровня, технологии программирования.

Соответствует актуальным требованиям Федерального государственного образовательного стандарта среднего профессионального образования и профессиональным требованиям.

Для студентов образовательных учреждений среднего профессионального образования, обучающихся по экономическим специальностям, преподавателей, специалистов организаций любого уровня и сферы хозяйствования.

УДК 681.3(075.32)
ББК 32.81я723

ISBN 978-5-534-07321-8

Оглавление

Предисловие

Алгоритмизация и программирование — один из модулей дисциплины «Информатика». Материал учебника подобран таким образом, чтобы в нем содержались ответы на большинство вопросов, предлагаемых на экзамене. Как показал опыт проведения интернет-экзамена среди студентов экономических специальностей, вопросы по данному модулю вызывают наибольшие трудности. Поэтому материал модуля дан максимально подробно и изложен по принципу «от простого к сложному». Дополнительный материал дан в минимальном объеме и предназначен только для облегчения усвоения основного.

После изучения материалов учебника студент должен освоить:

трудовые действия

✓ владения навыками составления программ на ПАСКАЛе, содержащих подпрограммы;

необходимые умения

✓ составлять программы на одном из языков структурного программирования;

✓ описывать и использовать объекты в программах на ПАСКАЛе;

необходимые знания

✓ понятия интегрированной среды программирования;

✓ целей, принципов и базовых конструкций структурного программирования;

✓ управляющих операторов языка ПАСКАЛЬ, реализующих базовые конструкции;

✓ понятия «парадигма программирования», «технология программирования»;

✓ общих представлений о современных технологиях создания программного обеспечения;

✓ моделей жизненного цикла разработки программного обеспечения.

Учебник предназначен для студентов СПО экономических (гуманитарных) специальностей и преподавателей, осуществляющих

подготовку к экзамену, проводимому в Интернете. Он может быть полезен и школьникам выпускных классов, готовящимся к поступлению в вузы с углубленным изучением информатики. Учебник может быть использован аспирантами гуманитарных специальностей, самостоятельно изучающими данный раздел информатики. Специалисты организаций любого уровня и сферы хозяйствования тоже могут здесь найти для себя полезный материал, позволяющий повысить уровень их квалификации.

ГЛАВА 1

Основы алгоритмизации

Задачи главы

1. Изучить понятие алгоритма.
2. Изучить свойства алгоритма.
3. Рассмотреть типы алгоритмических моделей.
4. Освоить способы описания алгоритмов.
5. Получить представление о методах разработки алгоритмов.

После изучения главы 1 бакалавр должен:

знать

- современные возможности реализации алгоритмов;
- принципы построения алгоритмов;

уметь

- составлять блок-схемы алгоритмов;
- составлять программы на алгоритмическом языке высокого уровня;
- работать в интегрированной среде изучаемых языков программирования;

владеть

- практическими навыками разработки и реализации алгоритмов обработки различных данных.

1.1. Понятие алгоритма и его свойства

Алгоритмом называют точное предписание, которое задается вычислительному процессу и представляет собой конечную последовательность обычных элементарных действий, четко определяющую процесс преобразования исходных данных в искомый результат. Приведенная фраза представляет собой не определение, а объяснение сути, поскольку понятие алгоритма является фундаментальным и не может быть выражено через другие, поэтому его следует рассматривать как неопределяемое.

В качестве примера алгоритма приведем алгоритм Евклида для нахождения наибольшего общего делителя (НОД) двух натуральных чисел. Вот одна из возможных формулировок этого алгоритма, описанная по шагам:

1) присвоить переменным X и Y значения, НОД которых ищется;
2) если $X > Y$, то перейти на шаг 5;
3) если $X < Y$, то перейти на шаг 6;
4) здесь $X = Y$. Выдать X в качестве результата. Конец работы;
5) заменить пару (X, Y) парой $(X – Y, Y)$ и вернуться на шаг 2;
6) заменить пару (X, Y) парой $(X, Y – X)$ и вернуться на шаг 2.

Выяснение того, какие объекты и действия над ними следует считать точно определенными, какими возможностями обладают комбинации элементарных действий, что можно и чего нельзя сделать с их помощью, — все это предмет теории алгоритмов и формальных систем, которая первоначально возникла в рамках математики и стала важнейшей ее частью. Как указывал В. А. Успенский, самым главным открытием в науке об алгоритмах, безусловно, было открытие самого понятия алгоритма в качестве новой и отдельной сущности.

Вычисления протекают во времени и в пространстве. Каждый шаг алгоритма выполняется за какое-то конечное время. Для размещения данных необходимо пространство — память. Рассмотрим основные свойства алгоритма.

Свойства алгоритма

Каждый алгоритм имеет дело с данными — входными, промежуточными и выходными.

Конечность. Понимается двояко: во-первых, алгоритм состоит из отдельных элементарных шагов, или действий, причем множество различных шагов, из которых составлен алгоритм, конечно. Во-вторых, алгоритм должен заканчиваться за конечное число шагов. Если строится бесконечный, сходящийся к искомому решению процесс, то он обрывается на некотором шаге и полученное значение принимается за приближенное решение рассматриваемой задачи. Точность приближения зависит от числа шагов.

Элементарность (понятность). Каждый шаг алгоритма должен быть простым, чтобы устройство, выполняющее операции, могло выполнить его одним действием.

Дискретность. Процесс решения задачи представляется конечной последовательностью отдельных шагов, и каждый шаг алгоритма выполняется за конечное (не обязательно единичное) время.

Детерминированность (определенность). Каждый шаг алгоритма должен быть однозначно и недвусмысленно определен и не должен допускать произвольной трактовки. После каждого шага либо указывается, какой шаг делать дальше, либо дается команда остановки, после чего работа алгоритма считается законченной.

Результативность. Алгоритм имеет некоторое число входных величин — аргументов. Цель выполнения алгоритма состоит в получении конкретного результата, имеющего вполне определенное отношение к исходным данным. Алгоритм должен останавливаться после конечного числа шагов, зависящего от данных, с указанием того, что считать результатом. Если решение не может быть найдено, то должно быть указано, что в этом случае считать результатом.

Массовость. Алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется *областью применимости алгоритма*.

Эффективность. Одну и ту же задачу можно решить по-разному и соответственно за разное время и с различными затратами памяти. Желательно, чтобы алгоритм состоял из минимального числа шагов и при этом решение удовлетворяло бы условию точности и требовало минимальных затрат других ресурсов.

Точное математическое определение алгоритма затрудняется тем, что интерпретация предусмотренных предписаний не должна

зависеть от выполняющего их субъекта. В зависимости от своего интеллектуального уровня он может либо не понять, что имеется в виду в инструкции, либо интерпретировать ее непредусмотренным образом.

Можно обойти проблему интерпретации правил, если наряду с формулировками предписаний описать конструкцию и принцип действия интерпретирующего устройства. Это позволяет избежать неопределенности и неоднозначности в понимании одних и тех же инструкций. Для этого необходимо задать язык, на котором описывается множество правил поведения, либо последовательность действий, а также само устройство, которое может интерпретировать предложения, сделанные на этом языке, и выполнять шаг за шагом каждый точно определенный процесс. Оказывается, что такое устройство (машину) можно выполнить в виде, который остается постоянным независимо от сложности рассматриваемой процедуры.

В настоящее время можно выделить три основных типа универсальных алгоритмических моделей. Они различаются исходными посылками относительно определения понятия алгоритма.

Первый тип связывает понятие алгоритма с наиболее традиционными понятиями математики — вычислениями и числовыми функциями. *Второй тип* основан на представлении об алгоритме как о некотором детерминированном устройстве, способном выполнять в каждый отдельный момент лишь весьма примитивные операции. Такое представление обеспечивает однозначность алгоритма и элементарность его шагов. Кроме того, такое представление соответствует идеологии построения компьютеров. Основной теоретической моделью этого типа, созданной в 1930-х гг. английским математиком Аланом Тьюрингом, является машина Тьюринга.

Третий тип — это преобразования слов в произвольных алфавитах, в которых элементарными операциями являются подстановки, т.е. замены части слова (под словом понимается последовательность символов алфавита) другим словом. Преимущества этого типа моделей состоят в его максимальной абстрактности и возможности применить понятие алгоритма к объектам произвольной (необязательно числовой) природы. Примеры моделей третьего типа — канонические системы американского математика Эмиля Л. Поста и нормальные алгоритмы, введенные советским математиком А. А. Марковым.

Модели второго и третьего типа довольно близки и отличаются в основном эвристическими акцентами, поэтому не случайно говорят о машине Поста, хотя сам Пост такое название не вводил.

Запись алгоритма на некотором языке представляет собой программу. Если программа написана на специальном алгоритмическом языке (например, на ПАСКАЛе или БЕЙСИКе), то говорят об *исходной программе*. Программа, написанная на языке, который непосредственно понимает компьютер (как правило, это двоичные коды), называется *машинной,* или *двоичной*.

Любой способ записи алгоритма подразумевает, что всякий описываемый с его помощью предмет задается как конкретный представитель некоторого класса объектов, которые можно описывать данным способом.

Средства, используемые для записи алгоритмов, в значительной мере определяются тем, кто будет исполнителем.

Если исполнителем будет человек, запись может быть не полностью формализована, на первое место выдвигаются понятность и наглядность. В этом случае можно использовать словесную форму записи или схемы алгоритмов.

Для записи алгоритмов, предназначенных для исполнителей-автоматов, необходима формализация, поэтому в таких случаях применяют формальные специальные языки. Преимущество формального способа записи состоит в том, что он дает возможность изучать алгоритмы как математические объекты; при этом формальное описание алгоритма служит основой, позволяющей интеллектуально охватить этот алгоритм.

Для записи алгоритмов используют самые разнообразные средства. Выбор средства определяется типом исполняемого алгоритма. Выделяют следующие **основные способы записи алгоритмов**:

- *вербальный* — алгоритм описывается на человеческом языке;
- *символьный* — алгоритм описывается с помощью набора символов;
- *графический* — алгоритм описывается с помощью набора графических изображений.

Общепринятыми способами записи алгоритма являются *графическая запись* с помощью схем алгоритмов (блок-схем) и *символьная запись* с помощью какого-либо алгоритмического языка.

Для описания алгоритма с помощью схем изображают связанную последовательность геометрических фигур, каждая из которых подразумевает выполнение определенного действия алгоритма. Порядок выполнения действий указывается стрелками.

В схемах алгоритмов используют следующие типы графических обозначений.

Начало и *конец* алгоритма обозначают с помощью одноименных символов (рис. 1.1).

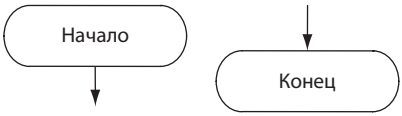

Рис. 1.1. Обозначения начала и конца алгоритма

Шаг алгоритма, связанный с присвоением нового значения некоторой переменной, преобразованием некоторого значения с целью получения другого значения, изображается символом «*процесс*» (рис.1.2).

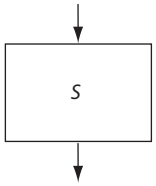

Рис. 1.2. Обозначение процесса

Выбор направления выполнения алгоритма в зависимости от некоторых переменных условий изображается символом «*решение*» (рис. 1.3).

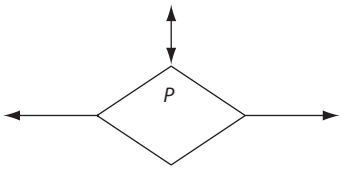

Рис. 1.3. Обозначение решения

Здесь *P* означает предикат (условное выражение, условие). Если условие выполнено (предикат принимает значение ИСТИНА),

то выполняется переход к одному шагу алгоритма, а если не выполнено, то к другому.

Имеются примитивы для операций ввода и вывода данных, а также другие графические символы. В настоящий момент они определены стандартом ГОСТ 19.701—90 (ИСО 5807—85) «Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». Всего сборник ЕСПД содержит 28 документов.

По схеме алгоритма легко составить исходную программу на алгоритмическом языке.

В зависимости от последовательности выполнения действий в алгоритме выделяют алгоритмы линейной, разветвленной и циклической структуры.

В алгоритмах **линейной структуры** действия выполняются последовательно одно за другим.

В алгоритмах **разветвленной структуры** в зависимости от выполнения или невыполнения какого-либо условия производятся различные последовательности действий. Каждая такая последовательность действий называется *ветвью алгоритма*.

В алгоритмах **циклической структуры** в зависимости от выполнения или невыполнения какого-либо условия выполняется повторяющаяся последовательность действий, называющаяся *телом цикла*. Вложенным называется цикл, находящийся внутри тела другого цикла. Итерационным называется цикл, число повторений которого не задается, а определяется в ходе выполнения цикла. Одно повторение цикла называется *итерацией*.

1.2. Методы разработки алгоритмов[1]

Ключевым подходом в алгоритмизации является сведение задачи к подзадачам. Это естественно, так как предстоит превратить один шаг в последовательность элементарных шагов. Само это преобразование также может состоять из нескольких этапов, на которых единственный шаг разбивается на несколько более простых, но еще не элементарных. Эти более простые шаги соответствуют

[1] Использованы материалы сайта http://it.kgsu.ru (автор — Т. А. Никифорова).

частным задачам (подзадачам), совокупное решение которых приводит к решению исходной задачи.

Различные методы разработки алгоритмов отличаются тем, на какие подзадачи производится разбиение и как эти подзадачи соотносятся между собой. Хотя общепринятой классификации методов разработки алгоритмов нет, тем не менее некоторые общие соображения на этот счет могут быть высказаны.

Любая задача может быть сформулирована как функция преобразования исходных данных в выходные данные: $f(X) = Y$. Как исходные данные, так и функция могут быть достаточно сложными. Например, X — число, вектор, текст на каком-либо языке, БД, рисунок, информация. То же можно сказать и о выходных данных. Функция может быть суммой чисел, тригонометрической функцией, корнем уравнения, переводом текста с русского на английский язык, раскраской полутонового рисунка и т.д.

Разбивая задачу на подзадачи, можно: разбивать исходные и выходные данные на части или упрощать их; производить декомпозицию функции, т.е. превращать ее в суперпозицию более простых.

Разбиение данных

Под **разбиением данных** понимается разделение структуры данных на части, например разделение вектора из десяти компонентов на два вектора по пять компонентов или разделение текста на предложения. Под **упрощением данных** понимаются такие ситуации, когда, например, X — число и его нельзя разбить на части, но его можно разложить, скажем, на сумму $X = X_1 + X_2$ так, что результаты $f(X_1)$, $f(X_2)$ отыскиваются проще, чем $f(X)$.

При разработке алгоритма может использоваться отдельно первый подход, отдельно второй подход или оба подхода совместно.

Разложение задачи на подзадачи

Разложение задачи в **последовательность разнородных подзадач** иногда называют методом «разделяй и властвуй».

В этом методе обычно выделяется относительно небольшое число подзадач. Например: задача — выполнить программу на компьютере; подзадачи — ввести исходный текст программы; транслировать программу в машинные команды; подсоединить к машинному коду стандартные процедуры из библиотеки; загрузить программу

в оперативную память; запустить процесс выполнения; завершить процесс выполнения программы.

Результаты решения первой подзадачи становятся исходными данными для второй подзадачи и т.д. Таким образом, здесь использован второй подход в чистом виде (декомпозиция функции, задание ее в виде суперпозиции более простых). Заметим также, что такая суперпозиция может быть задана последовательным соединением машин Тьюринга. На алгоритмическом языке этот метод может быть выражен записью последовательно вызываемых процедур.

Важный частный случай предыдущего метода — разложение задачи в **последовательность однородных подзадач** приобретает новое качество за счет того, что задача *P* сводится к *n* экземплярам более простой задачи *R* и к простой задаче *Q*, объединяющей *n* решений.

Пример

Вычислим скалярное произведение двух векторов — *A* и *B*:

```
S:=0;              {Задача Q — подготовка места для суммирования.}
for i:= 1 to n do
S:=S+A[i]*B[i]; {Задача R — перемножение компонент и суммирование.}
```

Этот алгоритм использует разбиение исходных данных на части — отдельные компоненты векторов.

Однородность подзадач позволяет значительно сократить длину текста алгоритма за счет применения операторов повторения. Итерация на уровне крупных подзадач или отдельных небольших операторов встречается в большинстве реальных алгоритмов и служит основным источником эффективного использования компьютера по сравнению с другими вычислительными средствами (например, непрограммируемым калькулятором).

Рекурсия

Рекурсия — это сведение задачи к самой себе. Задача так же, как и в предыдущем методе, сводится к более простой. Но эта более простая задача имеет ту же формулировку, что и исходная, с той лишь разницей, что решаться она должна для более простых исходных данных. Это чистый вариант упрощения исходных данных.