**Дейкстра Э.**

# Дисциплина программирования

# Математическое обеспечение ЭВМ

**Москва**
**«Книга по Требованию»**

УДК 51
ББК 22.1
Д25

**Дейкстра Э.**
Д25 Дисциплина программирования: Математическое обеспечение ЭВМ / Дейкстра Э. – М.: Книга по Требованию, 2013. – 274 с.

**ISBN 978-5-458-33419-8**

Книга написана одним из крупнейших зарубежных специалистов в области программирования, известным советскому читателю по переводам его книг на русский язык (например, «Структурное программирование», «Мир», 1972). Она посвящена фундаментальным вопросам конструирования корректных и изящных программ для ЭВМ. В ней предлагается методика формального вывода программы из математической постановки задачи. При этом прослеживается развитие алгоритмов вплоть до создания программ. Материал излагается в форме остроумных и поучительных задач по программированию. Книга представляет значительный интерес для широкого круга программистов.

**ISBN 978-5-458-33419-8**

О структуре и стиле книги достаточно полно сказано в предисловии автора. Там же он предупреждает, что читать его книгу трудно. Причина этого заключается в сложности самих программ, послуживших для нее материалом. Я хотел бы добавить, что они сложны для нас только сейчас, когда теория программирования делает свои первые шаги. Придет время, и такие программы сможет составлять (или выводить) прямо на уроке каждый школьник. И чтобы это время приблизить, надо осваивать, внедрять и развивать теорию. А этот процесс легко не проходит.

Перевод предисловия и глав 1—7 выполнен В. В. Мартынюком, глав 8—21 — И. Х. Зусман, глав 22—27 — Л. В. Уховым.

*Э. З. Любимский*

# ПРЕДИСЛОВИЕ

Историки таких древних интеллектуальных дисциплин, как поэзия, музыка, живопись и наука, высоко оценивают роль выдающихся практиков, чьи достижения обогатили опыт и расширили представления поклонников этих дисциплин, пробудили и укрепили таланты последователей. То новое, что ими внесено, основывается на сочетании виртуозного практического мастерства и проницательного осмысливания фундаментальных принципов. Во многих случаях влияние этих людей усиливалось благодаря их высокой культуре, богатству и выразительности их речи.

В этой книге в присущем ее автору утонченном стиле представлена принципиально новая точка зрения на существо программирования. Исходя из этой точки зрения, автор разработал совокупность новых методов программирования и средств обозначения, которые демонстрируются и проверяются на многочисленных изящных и содержательных примерах. Этот труд, несомненно, будет признан одним из выдающихся достижений в разработке новой интеллектуальной дисциплины — программирования для вычислительных машин.

*Ч. А. Р. Хоар*

## ОТ АВТОРА

Уже давно мне хотелось написать книгу такого рода. Я знал, что программы могут очаровывать глубиной своего логического изящества, но мне постоянно приходилось убеждаться, что большинство из них появляются в виде, рассчитанном на механическое исполнение, но совершенно непригодном для человеческого восприятия — где уж там говорить об изяществе. Меня не удовлетворяло и то, что алгоритмы часто публикуются в форме готовых изделий, почти без упоминания тех рассмотрений, которые проводились в процессе разработки и служили обоснованием для окончательного вида завершенной программы. Сначала я задумал изложить некоторое количество изящных алгоритмов таким способом, чтобы читатель смог прочувствовать их красоту; для этого я собирался каждый раз описывать истинный или воображаемый процесс построения, который приводил бы к получению искомой программы. Я не изменил своему первоначальному намерению в том смысле, что основой этой монографии остается длинная последовательность глав, в каждой из которых ставится и решается новая задача. Тем не менее окончательный вариант книги существенно отличается от ее первоначального замысла, поскольку возложенная мною на себя обязанность представлять решения в естественной и убедительной манере повлекла за собой гораздо большее, чем я ожидал, и я навсегда сохраню чувство благодарности судьбе за то, что взялся за эту работу.

Когда начинаешь писать подобную книгу, сразу возникает проблема: каким языком программирования пользоваться? И это не только вопрос представления! Наиболее важным, но в то же время и наиболее незаметным свойством любого инструмента является его влияние на формирование привычек людей, которые имеют обыкновение им пользоваться. Когда этот инструмент — язык программирования, его влияние, независимо от нашего желания, сказывается на нашем способе мышления. Проанализировав в свете этого влияния все известные мне языки программирования, я пришел к выводу, что ни они сами, ни их подмножества не подходят для моих целей. С другой стороны, я считал себя настолько не подготов-

ленным к созданию нового языка программирования, что дал зарок не заниматься этим в ближайшее пятилетие, и я твердо знаю, что этот срок еще не вышел. (Прежде, помимо всего прочего, мне нужно было написать эту монографию.) Я попытался выбраться из этого тупика, создав лишь подходящий для моих целей мини-язык и включив в него только те элементы, которые представляются совершенно необходимыми и достаточно обоснованными.

Эти мои колебания и самоограничение, если их неправильно понять, могут разочаровать многих потенциальных читателей. Наверняка разочаруются все те, кто отождествляет трудность программирования с трудностью изощренного использования громоздких и причудливых сооружений, известных под названием «языки программирования высокого уровня» или — еще хуже! — «системы программирования». Если они сочтут себя обманутыми из-за того, что я вовсе не касаюсь всех этих погремушек и свистулек, могу ответить им только одно: «А вполне ли вы уверены, что все эти погремушки и свистульки, все эти потрясающие возможности ваших, так сказать, «мощных» языков программирования имеют отношение к процессу решения, а не к самим задачам?» Мне остается лишь надеяться, что, несмотря на употребление мною мини-языка, они все же прочтут предлагаемый текст. Тогда они, возможно, признают, что помимо погремушек и свистулек имеется очень богатое содержание и возникнет вопрос, стоило ли большинство из них вообще придумывать. А всем читателям, интересующимся преимущественно разработкой языков программирования, я могу только принести извинения в связи с тем, что еще не могу высказаться на эту тему более определенно. Тем временем эта монография, возможно, наведет их на некоторые размышления и поможет им избежать ошибок, которые они могли бы совершить, если бы не прочли ее.

Процесс работы над книгой, который явился для меня непрерывным источником удивления и вдохновения, привел к появлению текста, основательно отличающегося от первоначального замысла. Сначала у меня было вполне понятное намерение представить построение программ с помощью аппарата, чуть более формального чем тот, который я имел обыкновение использовать в своих вводных лекциях, где семантика обычно вводилась интуитивно, а доказательства правильности представляли собой смесь строгих рассуждений, жестикуляции и красноречия. Разрабатывая необходимые основы для более формального подхода, я обнаружил два неожиданных обстоятельства.

Первая неожиданность состояла в том, что так называемые «преобразователи предикатов», выбранные мною в качестве средства изъяснения, позволили прямо определять связь между начальным и конечным состояниями без каких-либо ссылок на промежуточные состояния, которые могут возникать во время выполнения программы. Я обрадовался тому, что это дает возможность провести четкое разграничение двух основных проблематик программирования: проблематики математической корректности (речь идет о проверке, определяет ли программа правильное соотношение между начальным и конечным состояниями — и преобразователи предикатов обеспечивают нам формальное средство для такого исследования без рассмотрения вычислительного процесса) и инженерной проблематики эффективности (благодаря разграничению становится очевидным, что последняя проблематика определена только в связи с реализацией). Пожалуй, самое полезное открытие состоит в том, что один и тот же текст программы всегда допускает две (в известном смысле дополняющие друг друга) интерпретации. Интерпретация в виде кода преобразователей предикатов, которая представляется более подходящей для нас, противостоит интерпретации в виде кода для выполнения — ее я предпочитаю оставлять машинам!

Второй неожиданностью оказалось то, что самые естественные и систематизированные «коды преобразователей предикатов», какие я мог себе представить, потребовали бы недетерминированной реализации, если рассматривать их как «коды для выполнения». Вначале я содрогался от мысли, что придется ввести недетерминированность уже в однопрограммном режиме (слишком хорошо мне были известны сложности, возникающие из-за этого в мультипрограммировании); однако потом я понял, что интерпретация текста как кода преобразователя предикатов имеет право на независимое существование. (Оглядываясь назад, мы можем отметить, что многие проблемы мультипрограммирования, ставившие нас прежде в тупик, являются всего лишь следствием априорной тенденции придавать детерминированности слишком большое значение.) В конце концов я пришел к тому, что стал считать недетерминированность естественной ситуацией, при этом детерминированность свелась к довольно банальному частному случаю.

Установив эти основы, я приступил, как и намеревался, к решению длинной последовательности задач. Это занятие оказалось неожиданно увлекательным. Я убедился в том, что формальный аппарат позволяет мне ухватывать существо дела гораздо четче, чем раньше. Я получил удовольствие, обнаружив, что явная постановка вопроса о завершимости может

иметь большое эвристическое значение; тут я даже начал сожалеть об излишне распространенной склонности к частичной корректности. Но самое приятное состояло в том, что большинство решенных мною ранее задач теперь увенчалось более изящными решениями. Я воспринял это как весьма ободряющее свидетельство того, что разработанная методика и в самом деле улучшила мои программистские возможности.

Как следует изучать эту монографию? Лучшéе, что я могу посоветовать: прерывайте чтение, как только усвоите постановку задачи, и пытайтесь сначала решить ее самостоятельно, затем продолжайте чтение. Попытка самостоятельного решения задачи представляется единственным способом прочувствовать, насколько она трудна; кроме того, вы можете сравнивать мое решение с вашим и получить удовлетворение, если ваше окажется лучше. Предупреждаю заранее: не огорчайтесь, когда увидите, что этот текст читается отнюдь не легко. Те, кто изучали его в рукописи, часто испытывали затруднения (но вполне вознаграждались за это). Впрочем, каждый раз при анализе их затруднений мы совместно убеждались в том, что «виновным» оказывался вовсе не текст (т. е. способ изложения), а сам излагаемый материал. Мораль этого может быть только такова: нетривиальный алгоритм и вправду нетривиален, а его окончательная запись на языке программирования слишком лаконична по сравнению с рассуждениями, обосновывающими его разработку; эта краткость окончательного текста не должна нас дезориентировать. Один из моих сотрудников внес предложение (а я довожу его до вашего сведения, поскольку оно может оказаться полезным), чтобы небольшие группы студентов изучали книгу вместе. (Здесь я должен добавить в скобках замечание по поводу «трудности» этого текста. Посвятив немало лет своей научной жизни тому, чтобы прояснить задачи программиста и сделать их более подвластными нашему интеллекту, я обнаружил с удивлением (и раздражением), что мое стремление внести ясность приводит к систематическим обвинениям в том, что я «внес в программирование трудности». Но эти трудности всегда в нем были; и только сделав их видимыми, мы сможем надеяться, что научимся разрабатывать программы с высокой степенью надежности, а не просто «лепить команды», т. е. выдавать тексты, основанные на неубедительных предположениях, несостоятельность которых может выявиться после первого же противоречащего примера. Незачем и говорить, что ни одна программа из этой монографии не проверялась на машине.)

Я должен объяснить читателю, почему я пользуюсь мини-языком, столь ограниченным, что в нем нет даже процедур и рекурсий. Поскольку каждое следующее расширение языка добавляло бы к этой книге еще несколько глав, тем самым соответственно увеличивая ее стоимость, отсутствие большинства возможных расширений (таких, как, например, мультипрограммирование) не нуждается в дополнительных оправданиях. Однако процедуры всегда занимали такое важное место, а рекурсия для вычислительной науки в такой степени считалась признаком академической респектабельности, что некоторое разъяснение представляется необходимым.

Прежде всего эта книга предназначена не для начинающих, и я рассчитываю, что мои читатели уже знакомы с указанными понятиями. Во-вторых, книга не является вводным текстом по какому-то конкретному языку программирования, так что отсутствие в ней этих конструкций и примеров их употребления не следует объяснять моей неспособностью или нежеланием ими пользоваться или же воспринимать как намек на то, что вообще лучше воздерживаться от их применения. Просто они не потребовались мне для разъяснения моей главной мысли о том, насколько существенно тщательное разграничение проблематик для разработки всесторонне высококачественных программ; скромные средства мини-языка предоставляют нам более чем достаточный простор для нетривиальных и в то же время вполне приемлемых разработок.

Можно обойтись этим объяснением, но оно все же не является исчерпывающим. Я все равно считал себя обязанным ввести повторение как самостоятельную конструкцию, поскольку мне представлялось, что это следовало сделать уже давно. Когда языки программирования зарождались, «динамическая» природа оператора присваивания казалась не очень приспособленной к «статической» природе традиционной математики. Из-за отсутствия соответствующей теории математики ощущали некоторые затруднения, связанные с этим оператором, а поскольку именно конструкция повторения создает необходимость в присваиваниях переменным, математики ощущали затруднения и в связи с повторением. Когда были разработаны языки без присваивания и без повторений — такие, как чистый ЛИСП, — многие почувствовали значительное облегчение. Они снова ощутили под ногами знакомую почву и увидели проблеск надежды превратить программирование в занятие с твердой и солидной математической основой. (До сего времени среди склонных к теоретизированию специалистов по машинной математике все еще широко распространено мнение, что рекурсивные программы «более естественны», чем программы с повторениями.)

Другого выхода из положения путем надежного и действенного математического обоснования пары понятий «повторение» и «присваивание переменной» нам предстояло ждать еще десять лет. А выход, как показано в этой монографии, заключался в том, что семантику конструкции повторения можно описать с помощью рекуррентных отношений между предикатами, тогда как для описания семантики общей рекурсии требуются рекуррентные отношения между преобразователями предикатов. Отсюда совершенно очевидно, почему я считаю общую рекурсию на порядок более сложной конструкцией, чем простое повторение; и поэтому мне больно смотреть, как семантику конструкции повторения

«**while** *B* **do** *S*»

*определяют* как семантику обращения

«*whiledo* (*B*, *S*)»

к рекурсивной процедуре (описанной в синтаксисе языка АЛГОЛ 60):

**procedure** *whiledo (условие, оператор)*;

**begin if** *условие* **then begin** *оператор*;

*whiledo (условие, оператор)* **end**
**end**

Несмотря на формальную правильность, это мне неприятно, потому что я не люблю, когда из пушки стреляют по воробьям, вне зависимости от того, насколько эффективно пушка может справляться с такой работой. Для поколения теоретиков машинной математики, которые подключались к этой тематике в течение шестидесятых годов, приведенное выше рекурсивное определение часто является не только «естественным», но даже «самым правильным». Однако ввиду того, что без понятия повторения мы не можем даже описать поведение машины Тьюринга, представляется необходимым произвести некоторое восстановление равновесия.

По поводу отсутствия библиографии я не предлагаю ни объяснений, ни извинений.

*Благодарности.* Следующие лица оказали непосредственное влияние на разработку этой книги, либо приняв участие в обсуждении ее предполагаемого содержания, либо высказав замечания относительно готовой рукописи или ее частей: К. Брон, Р. Берсталл, У. Фейен, Ч. Хоар, Д. Кнут, М. Рем, Дж. Рейнольдс, Д. Росс, К. Шолтен, Г. Зигмюллер, Н. Вирт и

М. Вуджер. Я считаю честью для себя возможность публично выразить им мою признательность за сотрудничество. Кроме того, я весьма обязан корпорации Burroughs, создавшей мне благоприятные условия и предоставившей необходимые средства, и благодарен моей жене за неизменную поддержку и одобрение.

*Э. В. Дейкстра*

**Нейен**

**Нидерланды**