Потопахин В.

Современное программирование с нуля!

ДМК Пресс
издательство
Москва, 2010

УДК 32.973.26-018.2
ББК 004.438
П64

Потопахин В.
П64 Современное программирование с нуля! – М.: ДМК Пресс, 2010. – 240 с.: ил.

ISBN 978-5-94074-608-9

Эта книга для тех, кто хочет получить хорошие навыки программирования с использованием языка Компонентный Паскаль -современной версии языка Паскаль. Изложение сопровождается большим количеством примеров, способствующих успешному усвоению материала людьми с различным уровнем подготовки – необходимо только желание и терпение.

Материал курса представлен в виде последовательности прикладных задач, нацеленных на формирование у обучаемого особой программисткой логики и дающих возможность изучить и отработать на практике все существенные особенности языка Компонентный Паскаль.

На **CD**, прилагаемом к книге, находится бесплатная среда программирования, запустив которую, вы сразу сможете начать работу!

УДК 32.973.26-018.2
ББК 004.438

Потопахин Виталий Валерьевич

Современное программирование с нуля!

Главный редактор *Мовчан Д. А.*
dm@dmk-press.ru
Перевод *Ткачев Ф. В.*
Корректор *Синяева Г. И.*
Верстка *Старцевой Е. М.*
Дизайн обложки *Харевская И. В.*

Подписано в печать 16.11.2009. Формат 70×100 $^{1}/_{16}$.
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 36. Тираж 1000 экз.
№

Web-сайт издательства: www.dmk-press.ru
Internet-магазин: www.alians-kniga.ru

ISBN 978-5-94074-608-9

© Потопахин В., 2010
© Оформление, издание, ДМК Пресс, 2010

Содержание

Предисловие 4

Глава 1. Неформальное введение 5
Кратко о главном 6
Условные циклы 12
Общая структура программы на КП 16
Условный оператор 21
Какие еще есть типы данных в КП 29
Массивы 30
Вложенные циклы 36
Многомерные массивы 49
Процедуры 57
Рекурсия 75
Записи 86
Указательные типы 94
Связные списки 96
Деревья 106
Файлы 114

Глава 2. Систематическое введение в КП 117
Введение 118
Понятие числа 127
Понятие идентификатора 127
Величины. Типы данных. Объявление и виды типов 130
Операции 142
Операторы 145
Модули 160
Полный список предопределенных процедур 161

Глава 3. Практикум 165
Раздел A. Разные задачи 181
Раздел B. Сортировки 198
Раздел C. Задачи перебора 201
Раздел Д. Графы 209

Приложение. Кратко о теории графов 226

Заключение 240

Предисловие

Книга, которую вы начинаете читать предназначена для начинающих изучать программирование, или имеющих небольшой программисткий опыт. В общем это книга для неискушенных, но желающих научится многому. Конечно, для профессионального познания любой области, одной книги всегда мало, но если у вас хватит терпения и упорства в проработке практического материала, а наша книга почти целиком посвящена практике, то можно быть уверенным, что ваш программисткий уровень станет достаточным для дальнейшего уже профессионального роста.

Книга состоит из трех глав и одного приложения. Первая глава «Неформальное введение», фактически самодостаточный логически завершенный самоучитель. Тщательное изучение неформального введения даст небольшой, но уверенный навык решения прикладных задач, и знание языка Компонентный Паскаль. Все неформальное введение от первой до последней страницы это решение задач. Каждая задача разбирается достаточно подробно, задачи используются и для рассказа о языке. Языковые конструкции вводятся по мере необходимости, тогда когда это нужно для решения очередной задачи. Это делает изучение языка хотя и несколько бессистемным (но в книге есть еще и систематическое введение), но прозрачным и понятным. Уровень сложности решаемых задач постепенное растет, но по настоящему сложных задач в неформальном введении все же нет, поэтому для его усвоения достаточно упорства и желания.

Вторая глава – систематическое введение в язык КП (Компонентный Паскаль). Здесь уже нет практики. Вся глава посвящена теории, а точнее изложению языка. Если в неформальном введении язык излагался «между делом», а главным было решение задач, то здесь главным становится язык, поэтому вторая глава отличается более строгой логикой и более трудна для понимания, но если неформальное введение вами пройдено успешно, то и вторая глава не должна создать серьезных затруднений. Еще одно важное отличие неформального введения от систематического. В первой главе используется не весь язык КП. Вторая глава излагает язык в полном объеме.

Третья глава – практикум, целиком посвящена задачам, но если в первой главе задачи использовались для объяснения, то в третьей главе задачи предлагаются для самостоятельного решения. Уровень сложности уже существенно выше. Но от вас не требуется решения с пустого места. Каждая предлагаемая в практикуме проблема снабжена пояснениями разного уровня. Где-то это описание алгоритма, где-то изложение идеи, иногда просто подсказка, иногда довольно детальная помощь. Конечно, несмотря на помощь, над каждой задачей придется основательно потрудится, но если первые две главы усвоены успешно, то и третья вполне преодолима.

Кроме трех основных глав есть еще небольшое приложение, посвященное теории графов. Оно невелико по размеру и не предназначено для хорошего изучения теории. Это приложение появилось лишь в силу того, что такая математическая конструкция как графы довольно часто используется в задачах практикума. Поэтому было сочтено полезным дать хотя бы небольшой справочный материал.

Глава 1

Неформальное введение

Кратко о главном 6
Условные циклы 12
Общая структура
программы на КП 16
Условный оператор 21
Какие еще
есть типы данных в КП 29
Массивы 30
Вложенные циклы 36
Многомерные массивы 49
Процедуры 57
Рекурсия 75
Записи 86
Указательные типы 94
Связные списки 96
Деревья 106
Файлы 114

Кратко о главном

Программирование – наука являющаяся предметом данной книги стоит на трех слонах и одной черепахе. Слоны это: постановка задачи, алгоритм, программа. Каждое из перечисленных понятий имеет сложную историю образования, различные понимания, но для начала можно отвлечься от строгой науки и запомнить что:

- постановка задачи – это описание задачи в строгой математической терминологии;
- алгоритм – это описание действий, выполняя которые некий ИСПОЛНИТЕЛЬ ***обязательно*** получит требуемый результат. Обратите внимание на выделенное слово, именно так, не может получить, а обязательно получит;
- программа – это запись алгоритма на строгом, однозначно понимаемом языке.

А черепаха, на которой стоят наши слоны – это исполнитель, способный выполнить алгоритм. Кстати совершенно не обязательно компьютер. В самом общем случае необходимо говорить о устройстве способном выполнять определенный, жестко заданный набор команд, но наша цель – программирование компьютера, поэтому далее исполнитель это всегда компьютер.

Следовательно, научится программировать, это значит научится:

- формулировать задачу в строгих математических терминах;
- находить решение в виде последовательности действий понятных компьютеру;
- записывать эту последовательность на языке программирования.

Зачем нужен язык программирования

Язык, есть способ записи мысли. Это утверждение верно, как для естественного языка, так и для любого другого, в том числе и для языка программирования. Поэтому, главная проблема любого начинающего изучать науку программирования, это алгоритмический способ мышления, некоторые специальные методы и приемы рассуждений. Мыслить алгоритмически мы все более или менее умеем, практически любой человек в состоянии понять запись алгоритма на естественном языке, если предмет алгоритма ему известен, иначе говоря, если человек является исполнителем с достаточным для данного алгоритма набором команд.

Поэтому, по крайней мере на первых порах, проблема кажущаяся второстепенной выходит на первый план. Эта проблема названа в заголовке выше. Зачем нужен специальный язык и как им пользоваться? Попробуем сейчас решить небольшую конкретную задачу и убьем двух зайцев: ответим на поставленный вопрос и получим первую информацию о языке Компонентный Паскаль, который в дальнейшем станет основой нашего движения вглубь науки о программировании (или искусства).

Задача 1. Дано множество чисел. Найти сумму положительных.

Пока мы не знаем никакого языка кроме естественного, поэтому попробуем записать необходимую последовательность действий (алгоритм) на русском языке и посмотрим чем это будет хорошо или наоборот плохо.

Вариант 1:

Для всех чисел из данного множества:
если число положительное, то прибавляем его к сумме

Наверное, это описание будет понятно каждому кто немного знаком с математикой, хотя бы в пределах арифметики. Но к сожалению, все не так просто. В «алгоритме» (пока в кавычках, так как это описание еще очень далеко от алгоритма) сказано, что некоторую операцию необходимо выполнить для всех чисел из заданного множества, но не сказано, каким образом выбирать числа из этого множества. Следовательно, предполагается, что исполнитель алгоритма умеет это делать с произвольным множеством. То есть, он в состоянии перебрать произвольное множество и ни разу не ошибиться, не взять одно и то же число дважды и ни одно число не пропустить. Пожалуй, исполнитель с такой способностью должен обладать довольно высоким уровнем интеллекта, а следовательно быть довольно сложным и дорогим устройством. Это не может быть приемлемо. Решаемая задача проста и должна решаться простым устройством. А это означает, что придется процедуру перебора множества чисел как-то описать. И естественно как-то доопределить само понятие множества.

Пусть, все элементы множества пронумерованы и пусть известно, сколько во множестве элементов, например – N. Тогда проблема решается легко. Исполнитель начинает перебор с элемента, имеющего нулевой номер, а для получения следующего элемента увеличивает текущий номер на 1.

Вариант 2:

Для Номера изменяющегося от нуля до числа N-1 с шагом 1
выполнять действие
    Если очередное число больше нуля то прибавлять его к сумме положительных.

Это уже значительно лучше, но согласитесь, выглядит очень громоздко. А сейчас пойдем по пути упрощения записи. Рассмотрим первую фразу:

Для Номера изменяющегося от нуля до числа N-1 с шагом 1
выполнять действие

Здесь описан процесс изменения некоторой переменной величины, которую мы назвали номером. Существенно в этой записи только то что:

- переменная имеет имя, и это не обязательно слово Номер;
- исходное значение переменной равно нулю;
- конечное значение переменной равно N-1;

- переменная изменяется с шагом 1;
- на каждом шаге изменения переменной выполняется некоторое действие.

Попробуем переписать фразу, так чтобы эти существенные пункты не изменили своего смысла, но фраза стала короче. Следующий вариант:

Для k=1 до **k=N-1** с шагом 1 делать

Выделенный фрагмент содержит еще одну возможность для упрощения. Записав k=0 мы уже дали исполнителю информацию о том, что в дальнейшем речь пойдет о переменной по имени k, поэтому k=N-1 это пожалуй лишний повтор, и окончательная запись окажется очень короткой:

Для k=1 до N-1 с шагом 1 делать

Последнее, запись выполнена на русском языке. Конечно, подобную запись можно выполнить и на немецком и на хинди и на китайском и любом другом языке, но так уж получилось, что в качестве основы языков программирования взят английский. Поэтому перепишем запись следующим образом:

```
FOR k:=0 TO N-1 BY 1 DO
```

и мы получим запись так называемой конструкции цикла на языке компонентный Паскаль. Если для вас этого не сложно, то постарайтесь сразу отметить, что такая форма цикла называется циклом с параметром или циклом с шагом. Далее, будем пользоваться вторым его названием.

Примечания:

- в записи k:=0 двоеточие обязательно;
- шаг 1 считается наиболее часто встречающимся. Поэтому если шаг равен 1, то в КП (Компонентный Паскаль) запись BY 1 можно опустить.

Займемся второй фразой алгоритма

если очередное число больше нуля, то прибавлять его к сумме положительных

Очередное число это величина. Все величины должны иметь имена. Так как все перебираемые числа принадлежат одному и тому же множеству, то логично их всех назвать одним именем, а различать по номеру. То есть a[1] это элемент множества «a» с номером 1. заметьте не первый элемент, а элемент с номером 1. Это существенная разница. a[k] – это соответственно элемент множества «a» с номером k. Такое множество пронумерованных элементов имеющих одно имя называется массивом.

Еще одна используемая величина это «сумма положительных», дадим ему имя sum и перепишем фразу так:

Если a[k]>0 то sum:=sum+a[k]

И перейдя на английский, получим еще одну команду языка КП

```
IF a[k]>0 THEN sum:=sum+a[k];
END;
```

Ключевое слово END не предусмотрено алгоритмом, оно является требованием языка. Правила языка требуют завершать сложную конструкцию таким ключевым словом. А условная команда является сложной конструкцией.

А теперь полная запись

Листинг 1

```
sum:=0;
FOR k:=0 TO N-1 DO
    IF a[k]>0 THEN sum:=sum+a[k];
    END;
END;
```

Здесь две сложных конструкции, поэтому два ключевых слова END. Одно из них закрывает условную команду, второе завершает цикл.

Важное замечание. В окончательной записи добавилась команда sum:=0 которой не было в исходной записи алгоритма. Для понимания необходимости этой команды посмотрим внимательнее на запись sum:=sum+a[k]. Что здесь происходит? Команда берет уже посчитанное значение величины sum (оно справа от знака :=) увеличивает это значение на величину a[k] и полученный результат присваивает снова величине sum, то есть результат становится новым sum. Таким образом, на втором шаге sum образуется от sum полученного на первом шаге. На третьем шаге sum образуется от sum полученного на втором шаге и далее все ясно. А вот чему равно самое первое sum, от которого мы должны получить новое sum на первом шаге цикла? В команде цикла об этом ничего не говориться. А раз так, то первое значение может оказаться равным чему угодно, его значение вообще говоря это какой-то числовой мусор который к моменту работы нашего фрагмента оказался в ячейках памяти хранящих величину sum.

Поэтому, программист должен перед началом процесса вычисления величины позаботится о ее исходном значении. И команда sum:=0 именно такую работу и выполняет, а называется это инициализацией, то есть определением первого, исходного значения.

Сравните полученную запись с тем, что было изначально и вы согласитесь, что язык программирования сохраняя смысл записи, позволяет ее очень существенно укоротить. Это конечно не законченная программа. До полноценных программ еще довольно далеко. Сейчас мы обсуждаем только общие вопросы. Поэтому запишем еще один короткий алгоритм и ответим на главный вопрос, для чего нужен язык программирования.

Задача 2. Найти сумму квадратов натуральных чисел от 1 до N.

Не будем тратить времени на длинные рассуждения, запишем сразу алгоритм на КП.

Листинг 2

```
sum:=0;
FOR k:=1 TO N DO
    sum:=sum+k*k;
END;
```

А теперь главный вопрос

Наверное, вы уже согласны, что запись на строгом языке сокращает текст, делает его чтение общедоступным для большого количества людей, то есть может быть общепринятым стандартом общения для специалистов занимающихся разработкой алгоритмов. Это важно, но не это главное. Самое важное в языке программирования то, что он является связующим звеном между естественным языком и языком который в действительности понятен компьютеру. Это дает возможность писать специальные программы – трансляторы способные переводить программы, написанные на алгоритмических языках в тексты уже малопонятные для человека, но исполняемые компьютером.

Для компьютера не существует таких понятий, как множество, массив, переменная. Он оперирует регистрами, адресами ячеек памяти и т.д. и т.п. То есть чем то очень далеким даже для того, кто неплохо владеет математическим аппаратом. Поэтому до появления языков – посредников программирование было уделом немногих, ибо требовало слишком больших усилий даже для написания несложных программ.

Вспомним еще раз, что программирование это постановка задачи, алгоритмизация и кодирование (запись на языке). Все три «слона» одинаково нужны для решения любой задачи, поэтому будем просто решать задачи и одновременно учиться и алгоритмизации и кодированию, а постановка задачи пока означает запись условия задачи строгими математическими терминами. Для тех кто желает заниматься программированием глубоко еще будет возможность убедиться, что постановка задачи достаточно сложный и трудоемкий процесс.

А сейчас главная проблема это расширение языкового аппарата. В общем-то весь язык программирования сводится к набору понятий для перечисления которых хватит пальцев на одной руке. Это:

- величина (переменная или константа);
- команда присваивания;
- конструкция цикла;
- условная конструкция;
- процедура.