EESTI NSV TEADUSTE AKADEEMIA TOIMETISED. 19. KÖIDE
FÜÜSIKA * MATEMAATIKA. 1970, NR. 1

ИЗВЕСТИЯ АКАДЕМИИ НАУК ЭСТОНСКОЙ ССР. ТОМ 19
ФИЗИКА * МАТЕМАТИКА. 1970, № 1

https://doi.org/10.3176/phys.math.1970.1.01

Б. ТАММ

НЕКОТОРЫЕ КОНЦЕПЦИИ СИСТЕМНОГО ПРОГРАММИРОВАНИЯ

Введение

Условимся, как и в [1–3], считать язык кодов машинных команд языком программирования первого уровня, языки программирования в символических адресах, автокоды и коды ассамблеров — языками второго уровня, универсальные языки программирования типа АЛГОЛ, КОБОЛ и т. п. — языками третьего уровня или языками с процедурной ориентацией, а специализированные языки программирования, как САП, АПРОКС, АПТ и т. д., с ярко выраженными декларативными выразительными средствами — языками четвертого уровня или языками с проблемной ориентацией.

Когда речь идет о программах, написанных на языках четвертого уровня, будем вместо слова «программа» использовать понятие «языковая модель». Языковая модель — это программа, написанная на языке с проблемной ориентацией, где алгоритм решения проблемы зафиксирован в инженерных операциях — макроинструкциях, а не на уровне вычислительных терминов.

Одной из важнейших целей теории автоматического программирования является создание средств программирования, позволяющих расширить область применения ЦВМ и существенно увеличить количество людей, могущих общаться с цифровой машиной. В этой связи простота использования языков программирования приобретает огромную важность. Одной из причин, ограничивающих использование систем третьего уровня, является сложность их употребления: для освоения языка потребителю приходится учиться месяцами, а для достижения высоких результатов требуется довольно высокая математическая квалификация. Универсальность и простота использования — условия, с которыми системному программисту всегда придется сталкиваться, — взаимно противоречивы. От правильного совмещения этих условий во многом зависит значимость результатов его работы. Поэтому, создатель системы должен ответить не только на вопрос: какие классы задач данная система программирования должна решать? но и еще на вопрос: для каких потребителей она предназначается? [3].

В данной статье рассматриваются основные концепции создания систем программирования четвертого уровня, а также интегрированных систем программирования с проблемной ориентацией.

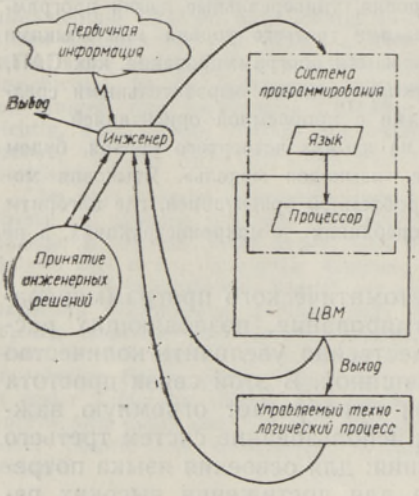
Специализированные алгоритмические языки с проблемной ориентацией претендуют на универсальность только в пределах того класса проблем, для решения которого они предназначены. Эти классы довольно узкие и относятся, как правило, к различным областям инженерной деятельности. Поэтому их выразительные средства базируются на

терминологии данной области и команды программ или языковых моделей состоят из инженерных, притом довольно декларативных макроинструкций. Благодаря этому подобные языки осваиваются специалистами, для которых они предназначены, быстро и широко используются.

Для успешной реализации систем четвертого уровня при их создании необходимо строго придерживаться концепций, от которых этот успех зависит. Эти концепции относятся как к окружающей среде и языку, так и к процессору системы программирования и формулируются в виде пяти принципов.

1. Системный принцип

Каждая система четвертого уровня является частью некоторой большей системы, куда, как правило, входят человек, цифровая вычислительная машина и инженерно-технический или технологический процесс. В связи с этим в каждом конкретном случае следует четко сформулировать функции каждого из этих основных звеньев и их взаимные, как аппаратурные, так и информационные связи (рис. 1). При этом необходимо учитывать следующее:



1. Какие функции целесообразно возложить на данную систему программирования с точки зрения технического процесса, имея в виду, что наряду с ЦВМ для управления технологической подготовкой данного процесса или для ее автоматизации могут применяться еще и другие средства.

2. Какими периферийными устройствами, осуществляющими связь с человеком, с одной стороны, и с процессом — с другой, располагает предполагаемая цифровая вычислительная система; какие основные характеристики имеет сама ЦВМ и какие дополнительные устройства требуются для промежуточного процессирования информации и контроля результатов.

3. Как распределить функции между человеком (т. е. инженером-программистом) и ЦВМ (т. е. системой программирования). Такое распределение должно являться результатом содержательного анализа того класса проблем, для которых предназначается проектируемая система программирования. При этом подлинно творческая часть, т. е. принятие основных инженерных решений, должна быть возложена на инженера, а принятие вытекающих из них частных решений, синтез подробного алгоритма и его осуществление на ЦВМ — на систему программирования.

Только совместный учет изложенных концепций в каждом конкретном случае позволяет соблюдать системный принцип, т. е. реализовать естественную интеграцию вычислительной машины в систему человек — ЦВМ — процесс.

Необходимо подчеркнуть, что недооценка системного принципа привела к приостановлению ряда начатых в данном направлении проектов как за рубежом, так и в нашей стране.

2. Принцип ориентированности

Все специализированные системы программирования, аналогично специализированным вычислительным машинам, предназначены для решения определенного класса задач или проблем. Этот факт подчеркивается также в названии рассматриваемых систем, именуемых системами с проблемной ориентацией.

Ориентация на определенный класс проблем определяет математико-логический аппарат системы программирования. В результате анализа этих проблем системный программист выясняет, чему должен соответствовать внутренний аппарат системы, чтобы решать все возможные проблемы данного класса, и на основе этого разрабатывает надлежащие вычислительные процедуры, подпрограммы, устанавливает информационно-логические связи и определяет некоторые другие функции процессора системы.

Наряду с этим принцип ориентированности имеет еще вторую, не менее важную сторону, заключающуюся в том, что система программирования должна быть ориентирована еще на определенный круг потребителей. От этой стороны принципа ориентированности во многом зависит выбор средств моделирования проблем, т. е. выбор выразительных средств языка программирования. От последних, разумеется, зависят и некоторые части процессора.

В отличие от систем программирования с процедурной ориентацией и от других видов математического обеспечения, система с проблемной ориентацией должна полностью соответствовать требованиям, диктуемым использователями данной системы. Дело в том, что если при разработке многих видов программных средств можно рассчитывать на высокий уровень специальных знаний потребителей, то в данном случае это исключается. Кроме того, каждая область инженерных процессов имеет свою специфику, которая должна быть учтена при выборе средств моделирования так, чтобы последние наилучшим образом соответствовали этой специфике. Эти соображения частично вытекают из системного принципа и, в свою очередь, частично отражаются в следующем принципе — принципе декларативности.

Следовательно, принцип ориентированности систем с проблемной ориентацией заключается в учете специфики того класса проблем, для которого система предназначена, а также в учете специфики круга потребителей системы. Эти равноценные стороны принципа ориентированности требуют от системного программиста ясного ответа на два изложенных в вводной части статьи вопроса.

3. Принцип декларативности

Характерным для составления программ на языках третьего уровня является необходимость определения процедур вычисления, требующихся для решения задачи, и описания тел этих процедур. В языковых моделях систем четвертого уровня программисту этого не нужно делать. Он не обязан заботиться о том, какие вычислительные процедуры и в какой последовательности должны быть осуществлены для решения его проблемы. Инженер-программист после принятия основных инженерных (творческих) решений моделирует свою проблему для ЦВМ в терминах базовых инженерных операций. Принятие надлежащих решений об использовании тех или иных процедур обеспечивается процессором системы, находящимся в ЦВМ. В этом заключается одно из существен-

ных различий между системами с процедурной и с проблемной ориентацией.

На языках первого уровня, т. е. в машинных кодах, необходимо фиксировать в программе все подробности вычислительного процесса и поэтому степень декларативности здесь равняется нулю. В программах, записанных на автокодах, уже наблюдается некоторая декларативность. Она еще увеличивается в языках третьего и четвертого уровней. Наряду с резким увеличением декларативности при переходе от третьего уровня на четвертый существенно увеличивается и простота использования этих языков. Последняя достигается еще и специализацией указанных языков.

Благодаря высокой декларативности и простоте выразительные средства языка четвертого уровня станут мощным и универсальным (в отношении проблем данного класса) аппаратом моделирования, позволяющим составлять различные языковые модели для решения одной и той же проблемы, аналогично, например, тому, как аппарат алгебры позволяет решать данную задачу различными методами и путями.

Язык с проблемной ориентацией $\mathcal{X}$ можно представить состоящим из множества слов $x, x \in \mathcal{X}$. Языковая модель $\mathcal{Y}$ является некоторым, имеющим смысл отображением Γ множества $\mathcal{X}$, таким, что каждому слову $x \in \mathcal{X}$ ставится в соответствие, в общем, элемент некоторого подмножества $\Gamma x \subset \mathcal{Y}$; при этом не исключается возможность $\Gamma x = \varnothing$. Для каждого данного языка существует большое множество отображений, превращающих $\mathcal{X}$ в $\mathcal{Y}$, но при составлении языковой модели конкретной проблемы количество этих отображений существенно уменьшается. Объясняется это наложением определенных условий на языковую модель со стороны множества принятых до составления модели инженерных решений $\mathcal{S}_\alpha$ по данной проблеме α, и со стороны множества первичной информации $\mathfrak{G}_\alpha$, находящейся в распоряжении инженера. Таким образом, языковая модель решения данной проблемы, написанная на языке $\mathcal{X}$, является отображением

$$\Gamma(x, \mathcal{S}_\alpha, \mathfrak{G}_\alpha) \subset \mathcal{Y}.$$

Важным свойством языков с проблемной ориентацией, вытекающим из принципа декларативности, является тот факт, что

$$\Gamma(x, \mathcal{S}_\alpha, \mathfrak{G}_\alpha) > 1,$$

т. е. они допускают составление более чем одной результативной языковой модели, инвариантной по отношению к получаемому на выходе ЦВМ результату. Чем больше количество возможных языковых моделей для описания желаемого решения данной проблемы, тем лучше язык. В лучших языках четвертого уровня, таких, как АПТ, САП, АПРОКС, ЕКСАПТ,

$$\Gamma(x, \mathcal{S}_\alpha, \mathfrak{G}_\alpha) \gg 1.$$

4. Принцип командной структуры

Языки с проблемной ориентацией имеют командную структуру [4, 5]. Это значит, что они позволяют составлять языковые модели из команд, написанных в инженерных макроинструкциях. Принцип командной структуры, частично вытекающий из двух предыдущих принципов, позволяет инженеру сформулировать свою проблему вычислительной ма-

шине таким же или почти таким же способом, как он это сделал бы, общаясь со своим коллегой по профессии.

Принцип командной структуры подчеркивает важность языкового моделирования на уровне базовых инженерных операций, в естественных декларативных инженерных терминах, где каждая команда имеет для инженера данной области определенный конечный смысл. Он подчеркивает важность отказа от всяких формальных форматов представления моделей, может быть, более простых с точки зрения вычислительной машины, но усложняющих труд инженера.

Аналогично тому, как инструкция автокода вызывает действие нескольких машинных кодов, как оператор языка третьего уровня активизирует действие нескольких подпрограмм, команда языка четвертого уровня может инициировать работу целого ряда вычислительных и логических процедур, необходимых для ее реализации.

5. Принцип промежуточных языков

Благодаря причинам ориентированности, декларативности и командной структуры языковые модели моделируемых процессов отличаются простотой, на их составление затрачивается на порядок меньше времени, чем на составление адекватных программ, написанных на языках более низких уровней, не говоря о ручных расчетах.

Но независимо от того, на каком уровне написана исходная программа, вычислительный процесс все равно происходит на уровне воплощенных в металле машины микроопераций. Следовательно, чем выше степень декларативности исходной программы, тем сложнее процесс преобразования ее в ЦВМ. В моделях или программах, записанных на языках с проблемной ориентацией, как уже отмечалось, не определены процедуры в явном виде, не описаны их тела и поэтому, в отличие от трансляторов процедурных языков, программирующей программе системы с проблемной ориентацией приходится взять эти функции на себя. Таким образом, эта программа должна выполнять задачи, далеко выходящие за пределы трансляции в обычном смысле. Чтобы отличать указанные трансляторы от трансляторов систем с процедурной ориентацией, многие авторы называют их процессорами.

Преобразование информации в процессорах проходит ряд последовательных, а иногда еще и параллельных этапов [4]. Поэтому правильная организация работы процессора имеет исключительно важное значение.

Принцип промежуточных языков определяет два основных положения построения процессоров:

а) блочную структуру;

б) организацию иерархии внутрипроцессорных промежуточных языков.

Эти положения предполагают, что различные этапы преобразования информации организованы в виде отдельных самостоятельных программных блоков и что информационная связь между ними осуществляется с помощью внутрипроцессорных промежуточных языков.

Благодаря блочной структуре процессоры могут быть реализованы на ЦВМ с малыми оперативными запоминающими устройствами, что особенно важно при решении инженерных задач, поскольку технические учреждения и заводы снабжены в основном ЦВМ, входящими в класс малых машин.

Инженерно-технические процессы время от времени меняются и совершенствуются. Эти изменения должны отражаться и в системах про-

граммирования. Блочная структура и промежуточные языки позволяют легко учитывать эти изменения, поскольку они, как правило, вызывают переделки лишь одного или нескольких блоков и требуют внесения соответствующих поправок в промежуточные языки.

Весьма часто данный инженерный процесс реализуется управляющими машинами различного типа, требующими входной информации в различных системах кодирования. В отношении процессоров, построенных по принципу промежуточных языков, это можно легко осуществить, подготовив соответствующее число различных последних блоков процессора (постпроцессоров) и связав их с остальной частью процессора через промежуточный язык предпоследнего (внутрипроцессорного) уровня [4].

Принцип промежуточных языков позволяет легко проверить работу процессора по частям и во многом облегчает отладку как отдельных частей, так и процессора в целом.

6. Об интегрированных системах с проблемной ориентацией

Одним из новейших и перспективных направлений развития математического обеспечения с проблемной ориентацией является создание больших многоцелевых систем программирования. Идея таких систем заключается в интеграции различных подсистем с проблемной ориентацией для решения комплексных инженерных задач, требующих участия специалистов различного профиля, без усложнения при этом языков программирования, предназначенных для потребителей. Основные функции и принципиальная схема такой системы описаны в [3, 4, 6].

При создании интегрированных систем с проблемной ориентацией также необходимо учитывать концепции, вытекающие из изложенных выше принципов. В данном случае они коротко сводятся к следующему:

1. К правильному прилаживанию интегрированной системы программирования к внешней среде, т. е. к четкому определению взаимоотношений системы с человеком (т. е. с различными категориями программистов), с одной, и с непосредственно управляемыми процессами, с другой стороны (системный принцип).

2. К соответствующему выбору изобразительных средств и внутреннего математико-логического аппарата для всех подсистем интегрированной системы (принцип ориентированности).

3. К максимизации степени декларативности ориентированных на потребителя языков, включая языки для определения команд и данных (принцип декларативности).

4. К необходимости организовать семантику и синтаксисы подсистем таким образом, чтобы они позволяли осуществлять моделирование процессов на уровне базовых инженерных операций каждой данной области в виде команд и избегать использования сложных машинно-ориентированных форматов представления языковых моделей или их частей (принцип командной структуры).

5. К блочной структуре всей системы, позволяющей осуществлять необходимые последовательные и параллельные информационные связи между отдельными модулями системы, и к использованию внутрисистемных промежуточных языков различного типа, среди которых отдельные должны обладать способностью осуществлять связи с некоторым множеством модулей системы, например с прекомпайлерами различных подсистем (принцип промежуточных языков).

ЛИТЕРАТУРА

1. Barton R. S., AFIPS Conf. Proc., **2**, 169 (1963).
2. Тамм Б., Изв. АН ЭССР, Физ. Матем., **16**, 267 (1967).
3. Тамм Б. Г., Тр. Первой всесоюзной конференции по программированию, вып. Ж, Киев, 1969, с. 109.
4. Pruuden J., Tamm B., IFIP-68 Congr. Proc., Booklet H, H 62-66, Edinburgh, 1968.
5. Тамм Б., Изв. АН ЭССР, Физ. Матем., **17**, 260 (1968).
6. Roos D., AFIPS Conf. Proc., **27**, Part 1 (1965).

Институт кибернетики
Академии наук Эстонской ССР

Поступила в редакцию
5/IX 1969

B. TAMM

MÕNED SÜSTEEMPROGRAMMEERIMISE KONTSEPTSIOONILISED KÜSIMUSED

Artiklis vaadeldakse probleemorientatsiooniga programmeerimissüsteemide loomise põhikontseptsioone ning formuleeritakse viis printsiipi, mis autori arvates peaksid olema nende süsteemide koostamise aluseks. Neist printsiipidest lähtudes on loodud süsteemid SAP ja APROKS. Käesolevatest kontseptsioonidest kinnipidamist võib täheldada ka parimates välismaistes uurimustes. Lühidalt käsitletakse nende printsiipide interpretatsiooni kompleksete tehniliste probleemide lahendamiseks loodavate integreeritud insenerlike programmeerimissüsteemide puhul.

B. TAMM

SOME CONCEPTUAL MATTERS OF SYSTEM-PROGRAMMING

The basic conceptions of the establishment of problem-oriented programming systems are viewed and five principles are formulated, which, in the author's opinion, should be the fundamentals of composing the above-mentioned systems. Proceeding from these principles, the SAP and APROKS systems have been established, and an adherence to these conceptions can be observed with the best foreign works, too. Briefly, the interpretation of these principles on the creation of integrated engineering programming systems for solving complex technical problems is dealt with.