А. Е. КУЛИНКОВИЧ, Е. Л. ЮЩЕНКО

О БАЗОВОМ АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

УДК 681.142.001 : 51

Электронные цифровые вычислительные машины (ЭЦВМ) имеют две широкие области применения: решение математических задач и обработка и хранение различной информации. На первом этапе ЭЦВМ применялись, главным образом, для решения задач вычислительной математики — с помощью машин стало возможно ликвидировать колоссальный разрыв между множеством разработанных аналитических методов и небольшим числом задач, решение которых удавалось доводить до численного результата. Ориентируясь именно за этот круг задач, и разрабатывались схемы и конструкции ЭЦВМ.

В настоящее время начинается второй этап в применении ЭЦВМ, характеризующийся все более широким их использованием для обработки и хранения больших массивов информации. Этот этап требует, во-первых, соответствующего совершенствования ЭЦВМ как по линии увеличения объема различных ступеней иерархической памяти, так и по линии автономии различных узлов с тем, чтобы операции ввода — вывода и обращения к внешним устройствам памяти не препятствовали работе функциональных (арифметических) устройств. Примером ЭЦВМ такого типа являются машины «Гамма-60» [1], «Стреч» [2] и т. д. Во-вторых, основная трудность, с которой приходится сталкиваться при постановке задач на ЭЦВМ, заключается в сложности и громоздкости машинных (внутренних) языков, с одной стороны, и в большом разнообразии алгоритмов переработки специальной информации, с другой. Таким образом, центральное место на данном этапе занимает вопрос упрощения связи человек—машина.

В связи с этим, наряду с широким фронтом работ по усовершенствованию вводных и выводных устройств машин (буквенный и многоколонный ввод—вывод, читающие автоматы, ввод — вывод графической и звуковой информации и др.), исключительно остро стоит вопрос разработки алгоритмических языков и основанных на них систем автоматического программирования.

Разработка алгоритмического языка имеет и самостоятельную цель — четкое и лаконичное описание алгоритмов обработки специальной информации. Такое описание может быть не только использовано для перевода его с помощью транслятора в машинный код, но также и для связи, с одной стороны, специалистов (лингвистов, геологов и др.), заинтересованных в автоматизации обработки получаемой информации, но не обладающих знаниями в области вычислительной техники (за исключением алгоритмического языка), и, с другой стороны, математиков-программистов и конструкторов.

Более того, алгоритмический язык может быть использован и для записи четкой инструкции, которая может быть выполнена не только машиной, но и «вручную» (можно привести большое число примеров, когда инструкции, особенно по обработке специальных видов информации, обладают недостатками именно потому, что они составлены не в алгоритмическом языке).

Развитие вычислительных машин в значительной мере идет по такой схеме: создается класс ЭЦВМ, а затем подбираются алгоритмы, которые могут быть эффективно реализованы на них. Однако более последователен другой путь: вначале составляются алгоритмы, которые необходимо реализовать на ЭЦВМ, и, в соответствии с данным классом алгоритмов, создаются машины, наиболее эффективные с точки зрения реализации этих алгоритмов. В настоящее время все большее значение приобретает второй путь. В связи с этим резко возрастает роль алгоритмизации процессов преобразования информации и, соответственно, становится еще более актуальной разработка алгоритмических языков программирования.

В настоящее время известен ряд алгоритмических языков, более или менее успешно применяемых для автоматизации программирования. Наибольшее признание получил международный язык АЛГОЛ-60, весьма удобный для описания алгоритмов задач вычислительной математики, но менее эффективный, а часто и вовсе неудобный для описания алгоритмов переработки информации. Можно сказать, что АЛГОЛ-60 является *алгоритмическим языком первого этапа применения ЭЦВМ*.

Для целей описания алгоритмов переработки информации предложен ряд специализированных языков — COBOL (для обработки деловой информации), COMIT (для перевода с одного языка на другой) и многие другие.

Процесс создания большого числа не связан-

ных друг с другом специализированных языков идет полным ходом, и вряд ли все аспекты этого процесса можно считать положительными. Целесообразно отметить следующие моменты.

Во-первых, разделение круга задач на задачи математические (вычислительные) и задачи, связанные с переработкой информации, весьма и весьма условно, поскольку при обработке многих видов информации приходится решать (и нередко довольно сложные) вычислительные задачи.

Во-вторых, специальные языки, предназначенные для обработки определенных видов информации, имеют много общего между собой, и желательно иметь некоторый язык, который служил бы общей основой для специализированных языков.

Таким образом, в настоящее время резко ощущается необходимость в создании общей теории формальных языков и на ее основе создания единой базовой схемы алгоритмических языков, по отношению к которой все алгоритмические языки являлись бы частными реализациями, то есть конкретными разновидностями языков, получаемых из нее посредством определенной интерпретации и наложения естественных ограничений на использование возможных изобразительных средств, выбора определенной формы записи из некоторого множества возможных форм и т. д. Создание такой схемы, пригодной для описания любого класса алгоритмов, имеющего практическое значение, можно считать актуальной задачей, соответствующей второму этапу применения электронных цифровых вычислительных машин.

Базовой схеме алгоритмических языков могут быть предъявлены следующие требования в отношении порождаемых ею языков.

1. Языки должны быть абсолютно четкими, не допускающими (при правильном переводе) неоднозначности при реализации записанных в них алгоритмов на ЭЦВМ.

2. Языки должны быть лаконичными; избыточная информация в алгоритмическом тексте должна ограничиваться определенным минимумом.

3. Языки должны быть удобными для использования их при составлении алгоритмов человеком и достаточно «наглядными» для максимального облегчения чтения человеком.

4. Языки должны быть такими, чтобы алгоритмические тексты допускали достаточно эффективное преобразование в программы для ЭЦВМ, а также другие виды обработки алгоритмических текстов, о которых будет сказано ниже.

5. Языки должны быть возможно более гибкими и богатыми, — класс алгоритмов, описываемых на данном языке, должен быть достаточно большим.

6. Языки должны быть строго формализованными, — синтаксис и семантика алгоритмического языка должны быть четко описаны.

Анализ структуры базовой схемы алгоритмических языков естественно начать с рассмотрения видов его частных реализаций. В качестве частных реализаций могут быть рассмотрены уровни, специализации, стили, диалекты и варианты алгоритмических языков. Рассмотрим более подробно эти виды частных реализаций.

Уровни алгоритмического языка. *Элементами* алгоритмического языка (как, впрочем, и многих других языков) являются *имена* и *предложения**. *Денотатами имен* являются *содержимые* элементов (или частей) памяти машины, *денотатами предложений — события***. События в ЭЦВМ представляют собой операции, производимые над содержимым элементов памяти машины. Каждый алгоритм описывает некоторую цепь событий (цепочку операций). Если алгоритмический язык обладает достаточными изобразительными средствами для описания каждой команды, реализуемой на ЭЦВМ, то любая последовательность событий, происходящих в процессе работы машины (то есть

---

* Алгоритмический язык, как и всякий формальный язык, задается совокупностью четырех элементов: 1) алфавита; 2) синтаксиса, определяющего множество допустимых в данном языке слов; 3) предметного множества объектов, являющихся денотатами (десигнатами) допустимых слов и 4) семантики, определяющей соответствие между допустимыми словами языка и элементами предметного множества.

** А. Черч [3] считает, что денотатами предложений являются истиностные значения (истина, ложь). Такой подход очень часто (особенно при анализе алгоритмических языков) оказывается неудовлетворительным. Предложениям в реальной действительности соответствуют определенные явления — события, множество которых и образует предметное множество «языка предложений». Среди многих операций над языками можно, например, выделить такую: подмножества предметного множества, которым соответствуют различные слова x_1, x_2, ... языка, объединяются, после чего слова x_1, x_2, ... рассматриваются как синонимы (то есть как слова, которые имеют один и тот же денотат). Такую операцию назовем *укрупнением языка*. Нетрудно видеть, что концепция Черча соответствует «максимальному укрупнению» языка событий, когда все истинные события объединяются в одно множество $M_{ис}$ и все предложения (истинные предложения), денотаты которых принадлежат к $M_{ис}$, рассматриваются как синонимы.

любой алгоритм, реализуемый на данной машине), может быть в нем описана. Такой язык будет соответствовать машинному уровню. Поскольку алгоритмический язык машинного уровня должен совпадать с программой в кодах машины (с точностью до перекодировки), самостоятельная ценность его невелика.

Необходимость в алгоритмическом языке возникает при более высоком уровне — общемашинном («мета-машинном»). ЭЦВМ различных марок обладают, вообще говоря, различными наборами реализуемых операций. Однако многие операции могут быть реализованы на всех марках машин (из некоторой группы марок ЭЦВМ). Если алгоритмический язык располагает средствами описания всех этих операций, то алгоритм, записанный на таком языке, нетрудно преобразовать в машинный код любой марки ЭЦВМ (из данной группы). Таким образом, уже при переходе на общемашинный уровень появляется возможность точного описания алгоритмов (программ) до выяснения того, на какой из машин алгоритм будет реализован, что является существенным преимуществом программирования в этом языке.

Запись на алгоритмическом языке общемашинного уровня является весьма детальной, что является определенным недостатком, так как чем детальнее запись, тем она более громоздка. Чтобы сделать запись алгоритмического текста более лаконичной, некоторые последовательности предложений можно заменять одним предложением.

Так, например, сложная запись, состоящая из нескольких операторов (предложений алгоритмического языка):

$$«i := 1;$$

$$k : \text{if } i > n \text{ then go to } A;$$

$$b[i] := a[i];$$

$$i := i + 1;$$

$$\text{go to } k; \quad A : ;»,$$

в языке АЛГОЛ-60 может быть записана в форме одного предложения:

$$«\text{for } i := 1, i + 1 \text{ while } i \leqslant n \text{ do } b[i] := a[i];».$$

Такой «укрупненный» язык можно назвать *языком среднего уровня*. Типичным языком среднего уровня является язык АЛГОЛ-60.

Следует подчеркнуть, что чем выше уровень алгоритмического языка, тем ниже требования к специальным знаниям по вычислительной технике, предъявляемые к составителю алгоритмического текста.

Алгоритмические языки среднего уровня, играя очень важную роль при алгоритмизации решения различных задач, не всегда оказываются удобными. Возникает необходимость еще более повысить уровень алгоритмического языка до уровня, который можно назвать субчеловеческим — по форме он близок к естественным человеческим языкам и отличается от последних лишь строгой формализацией. Овладеть языком субчеловеческого уровня особенно просто. Записи на этом языке могут иметь, например, такой вид:

«По данным геофизических исследований скважин 217, 315, 472 и 538 Зурбаганской площади выделить продуктивные пласты и определить их пористость и нефтенасыщенность».

Описанное выше разделение на уровни произведено в основном по одному признаку — какой длины «цепи событий» соответствуют предложениям данного алгоритмического языка. Однако, процесс перехода от «машинных» языков к «почти-человеческим» связан и с другими моментами. Так, весьма важен переход от конкретных адресов, которые фигурируют в машинных кодах, к условным адресам (переход от *уровня конкретных адресов к уровню условных адресов*) и далее, от условных адресов к адресам, задаваемым в косвенной форме (общеалгоритмический уровень*). С другой стороны, *укрупнение* элементов алгоритмического языка связано с приближением его к естественным языкам: *сжатие* языка за счет укрупнения его элементов делает допустимым некоторое *разбавление* его *избыточной информацией*, характерное для естественных языков. Внесение такой *избыточной информации* в случае языков более низких уровней делает тексты весьма громоздкими, а следовательно, и менее удобными для использования. Весьма существенные нарекания в этом отношении может вызвать язык КОБОЛ.

Порождаемый базовой схемой алгоритмический язык будет предельно гибким, если он будет включать все указанные уровни—от машинного до субчеловеческого.

Разработку алгоритмического языка естественно начать с наиболее низких уровней, постепенно вырабатывая по мере решения поставленных задач наиболее эффективные изобрази

---

* Выделение уровней алгоритмического языка в зависимости от характера использования адресов было произведено Е. Л. Ющенко [4] при разработке адресного программирования.

тельные средства, поднимаясь до языков более высоких уровней. В этом направлении уже проделана большая работа как по линии разработки изобразительных средств, так и по созданию соответствующих трансляторов.

Выделение в алгоритмическом языке *специализаций* связано с тем, что для задач, относящихся к различным областям знаний, часто требуются различные изобразительные средства для описания алгоритмов, учитывающие их специфику, сложившиеся традиции (условные обозначения, терминологию), и т. д. Специализация особенно рельефно проявляется на более высоких уровнях языка. Оформление конкретной специализации может быть осуществлено:

путем установления для каждой из них определенной группы стандартных процедур, то есть процедур, не требующих описания в алгоритмическом тексте;

путем закрепления стандартных идентификаторов;

путем включения специальных символов, не используемых в других специализациях алгоритмического языка;

путем допущения специальных синтаксических конструкций и т. д.

Выделение *стилей* языка обусловлено ограничениями, накладываемыми на язык в связи с особенностями применяемых трансляторов.

Выделение различных *диалектов* алгоритмического языка связано с использованием изобразительных средств, отражающих возможности не всех машин рассматриваемого класса, а только их части или же даже только одной машины.

Появление *вариантов* алгоритмического языка обязано использованию различных форм записи предложений в алгоритмическом языке. Выбор той или иной формы записи во многом зависит от случайных причин (традиции, кажущиеся существенными для той или иной математической школы, вопросы приоритета, особенности вводных устройств, накладывающие ограничения на применяемый алфавит, и пр.).

Так, например, следующие записи на алгоритмических языках

«$a + b \Rightarrow a$;

$P\{a > c\}M$»

(адресный язык)

«$a := a + b$;

if $a > c$ **then go to** M»

(АЛГОЛ-60)

«COMPUTE A = A + B;

IF A IS GREATER THEN C GO TO M—ROUTINE»

(КОБОЛ)

отличаются только формой. Переход от одной формы записи к другой весьма прост и может быть записан алгоритмически в виде несложной группы операторов, в простейшем случае — в виде одного оператора (формулы) замены. Учитывая случайный, несущественный характер выбора той или иной формы языка, можно алгоритмические языки, различающиеся лишь формой записи, считать изоморфными.

Примером варианта алгоритмического языка может служить учебный вариант, который использует вместо кратких обозначений пространные, более легкие для запоминания и более удобные при составлении алгоритмов на первом (учебном) этапе. В дальнейшем, по мере повышения квалификации программиста, такие длинные обозначения становятся неудобными, так как делают запись алгоритма существенно более громоздкой, и их целесообразно заменить предельно краткими, «сжимая» тем самым текстовую информацию.

Так, например, громоздкая запись «**begin integer** i, n, m, p, q, d, f; **integer array** $j[1:n]$; **real** A, B; **real array** $a[1:m]$, $c[d:f]$; **Boolean** *phy*; **for** $i :=$ **if** $A > B$ **then** p **else** q **step if** *phy* **then** 1 **else** 2 **until if** $A > B$ **then** $n + m$ **else** $n - 1$, $i + 1$ **while** $phy \wedge A = B$ **do**

begin $c[i] := a[j[i]]$;

if $A < C[i]$ **then go to** B **end end**»

может быть записана более просто: «$\{\boldsymbol{Ц}\ i, n, m, p, q, d, f, \boldsymbol{M}\ j\ [1:n]$; $\boldsymbol{g}\ A, B, \boldsymbol{M}\ a[1:m], c[d:f]$; $\boldsymbol{Л}\ phy$; $\boldsymbol{Ц}\{i := \boldsymbol{P}\ A > B : p : q : \boldsymbol{P}\ phy : 1 : 2 : \boldsymbol{P}\ A > B : n + m : n - 1,\ i + 1 : phy \wedge A = B\{c[i] := a[j[i]]; \boldsymbol{P}\ A < C[i] : \boldsymbol{к}\ B\}\}$».

Переход от первого варианта ко второму может быть осуществлён с помощью следующей формулы (оператора) замены: $\boldsymbol{З}\{\boldsymbol{Ц} \to$ **integer**, $\boldsymbol{g} \to$ **real**, $\boldsymbol{M} \to$ **array**, $\boldsymbol{Л} \to$ **Boolean**, $\boldsymbol{P} \to$ **if**, $: \to \to$**then**, $: \to$ **else**, $: \to$**step**, $: \to$**until**, $: \to$ **while**, $\boldsymbol{Ц}\{\to$ **for**,$\} \to$ **do**, $\{\to$ **begin**,$\} \to$ **end**, $\boldsymbol{к} \to$ **go to**$\}$.

Замена ряда разделителей (**step**, **until**, **then**, **else**, **while**) одним (:) связана с тем, что эти разделители являются в определенном смысле тождественными (замена их одним и тем же символом не нарушает однозначности расшифровки текста).

Запись:

«дан геоф скв 217, 315, 472, 538 Зурб вид пр опр пор неф» будет являться *сжатым* вариантом

приведенной выше (стр. 5) записи алгоритмического предложения на *субчеловеческом* уровне (которая, следовательно, соответствует учебному варианту алгоритмического языка). Переход от учебного варианта к основному осуществляется выбрасыванием слов и фрагментов слов, не несущих алгоритмической информации.

Алгоритмический текст может подвергаться различной обработке на ЭЦВМ. В зависимости от характера этой обработки можно выделить несколько классов *программ-процессоров**.

1. *Оптимизаторы*, осуществляющие нахождение наиболее оптимальной (например, в смысле затрачиваемого машинного времени) реализации данной записи на языке машинного уровня. Оптимизаторы могут входить в качестве составных частей транслятора.

2. *Программирующие программы* (*трансляторы*), осуществляющие перевод алгоритмического текста в программу для конкретной машины. Могут быть предложены трансляторы различных типов.

Простые трансляторы непосредственно переводят алгоритмический текст в машинный код. *Многоступенчатые* трансляторы преобразуют алгоритмический текст с одного уровня на другой (с более высокого на более низкий), вплоть до машинного языка (машинного кода). Для базовой схемы языков важны саморедактирующиеся трансляторы, способные на основе характеристики данной реализации алгоритмического языка (например, данной специализации) «редактировать» *пратранслятор*, компонуя из фрагментов последнего *рабочий транслятор*, пригодный для обработки алгоритмов, составленных в соответствии с данной специализацией языка. Саморедактирующийся транслятор является частным случаем *самоорганизующегося транслятора*, способного совершенствовать свою структуру на основе анализа обрабатываемых алгоритмических текстов.

Одной из главных особенностей системы алгоритмических языков с единой базовой структурой должно являться использование сложных трансляторов (многоступенчатых, саморедактирующихся, самоорганизующихся и т. д.).

Следует заметить, что процесс преобразования алгоритмического текста в машинный код не обязательно должен быть автоматизирован от начала до конца, по крайней мере на данном этапе применения ЭЦВМ. В ряде случаев алгоритмизацию удобно производить на таком высоком уровне, что непосредственное преобразование потребует очень сложных трансляторов, разработка и применение которых может оказаться делом экономически неоправданным. Тогда возможен перевод алгоритмического текста вручную на некоторый более низкий уровень, допускающий автоматическую обработку*. Целесообразность такого перевода обусловлена, во-первых, тем, что он может быть осуществлен специалистами более низкой квалификации, чем это необходимо для составления алгоритма; во-вторых, тем, что программирование на более высоком уровне освобождает сотрудника от знания деталей транслирования (алгоритмический текст оказывается связующим звеном с математиками и другими специалистами в области вычислительной техники).

Особой разновидностью трансляторов являются *мультипликаторы*, преобразующие ряд алгоритмических текстов в программу для многоканальной ЭЦВМ (ЭЦВМ с мультипрограммированием).

3. *Специализаторы*, способные на основе обработки ряда алгоритмических текстов, являющихся представительными для некоторого класса алгоритмов, рекомендовать определенную реализацию алгоритмического языка (совокупность ограничений, стандартных идентификаторов и т. д.), наиболее оптимальную как в выборе изобразительных средств языка, так и с точки зрения перевода в машинный код.

4. *Анализаторы* — программы, в задачу которых входит на основе обработки представительной группы алгоритмических текстов найти параметры (быстродействие, объем и характер памяти, количество каналов и т. д.) ЭЦВМ, наиболее эффективной для реализации алгоритмов данного класса. Параметры, выданные специализатором, могут быть использованы либо для выбора марки машины, либо (если нет ЭЦВМ, удовлетворяющих заданным параметрам) для составления соответствующего технического задания.

5. *Процессоры высших рангов* — программы, обрабатывающие или порождающие другие процессоры.

Среди предложенных алгоритмических языков, по-видимому, наиболее удовлетворяющим перечисленным выше требованиям к базовой

---

* В литературе (например, [5]) слово «процессор» употребляется как синоним слова «транслятор». Здесь предлагается вкладывать в этот термин более широкий смысл, понимая под ним любого рода программы, обрабатывающие алгоритмические тексты.

* Эта идея положена в основу построения ряда трансляторов, разработанных в ИК АН УССР.

АКАДЕМИЯ НАУК
УКРАИНСКОЙ ССР

19 2 65

КИБЕРНЕТИКА

схеме языков является адресный язык* [4]. Этот язык основан на использовании отношений адресности и успешно применяется не только для описания алгоритмов вычислительного характера, но и для алгоритмов обработки текстовой информации (программирующих программ, Марковских алгоритмов и т. д).

Можно указать следующие пути в направлении разработки базовой схемы алгоритмических языков:

1. Анализ общих черт различных алгоритмических языков.

2. Выяснение единиц алгоритмического языка, тех «кирпичей, из которых складываются алгоритмические тексты, и разработка их методов синтаксического описания.

3. Анализ изобразительных средств алгоритмических языков, анализ «алгоритмических фигур», встречающихся в алгоритмах различных классов, и нахождение для них наиболее удачных синтаксических конструкций.

4. Разработка наиболее универсальных способов записи различных алгоритмов и их фрагментов.

5. Анализ путей «сжатия» алгоритмической информации.

6. Выяснение способов приближения алгоритмического языка по форме к «человеческим» языкам и по символике и терминологии к специальным языкам (разработка «субчеловеческих» уровней алгоритмического языка).

7. Выяснение закономерностей выделения тех или иных частных реализаций алгоритмического языка.

8. Совершенствование способов описания синтаксиса и семантики алгоритмического языка.

9. Выяснение вопросов выбора наиболее оптимального алфавита для алгоритмического языка (и его частных реализаций).

10. Разработка трансляторов, в том числе сложных (многоступенчатых, самосовершенствующихся), и анализ структуры алгоритмического текста с целью обеспечения наибольшей эффективности работы транслятора.

11. Разработка других видов процессоров и совершенствование алгоритмического языка в направлении повышения эффективности их работы.

Поступила в редакцию
4. I 1965

---

* В связи с ограниченностью возможностей языка АЛГОЛ в литературе горячо обсуждается вопрос создания специальных языков для описания трансляторов (см., например, [5]). При этом предложения ряда авторов (например, [6]) сводятся по существу к дополнению средств языка АЛГОЛ отношениями адресности.

Можно привести примеры недостаточности изобразительных средств в языке АЛГОЛ, обуславливающие трудности (порою непреодолимые) при описании многих алгоритмов, что делает затруднительным использование АЛГОЛ в качестве базовой схемы языков: трудности в описании алгоритмов обработки массивов с нефиксированной размерностью и подмассивов; сложность описания многих алгоритмов, связанных с использованием древообразных форматов [7] (в американской литературе такие форматы получили не совсем удачное название списков — list); строго фиксированное число символов — функторов; отсутствие средств для описания алгоритмов строчной информации; ограниченные возможности для описания статической информации и т. д.

ЛИТЕРАТУРА

1. Курс программирования для ГАММА-60, ИЛ, 1962.
2. Кибернетический сборник, № 5, 1962.
3. А. Черч, Введение в математическую логику, т. 1, ИЛ, 1960.
4. Е. Л. Ющенко, Адресное программирование, К., 1964.
5. Proceedings of the Symposium organized and edited by the International Computation Centre Rome, March, 1962.
6. A. A. Grau, A Translator — oriented Symbolic Programming Language, Journal of the Ass. for Computing machinery, v. 9, № 4, 1962.
7. А. Е. Кулинкович, О поиске, учитывающем энтропию словаря, Сб. «Кибернетика и техника вычислений», К., 1964.