АКАДЕМИЯ НАУК СССР
СИБИРСКОЕ ОТДЕЛЕНИЕ

СОВЕТ МАТЕМАТИЧЕСКОЙ СЕКЦИИ ОБЪЕДИНЕННОГО СОВЕТА ПО ФИЗИКО-МАТЕМАТИЧЕСКИМ И ТЕХНИЧЕСКИМ НАУКАМ

---

На правах рукописи

А.П.ЕРШОВ

# НЕКОТОРЫЕ ВОПРОСЫ ТЕОРИИ ПРОГРАММИРОВАНИЯ И КОНСТРУИРОВАНИЯ ТРАНСЛЯТОРОВ

Автореферат диссертации, представленной на соискание ученой степени доктора физико-математических наук

НОВОСИБИРСК
1966

47-12

Работа выполнена
в Вычислительном центре
СО АН СССР

47-13

Диссертация состоит из введения и трех глав. В первой части введения дается краткий обзор развития и состояния теории и автоматизации программирования. Этот обзор не является библиографическим и содержит ссылки только на сравнительно небольшое число работ, наиболее, по мнению автора, повлиявших на развитие программирования. Во второй части введения дается систематический обзор основных результатов диссертации. Текст автореферата совпадает с текстом введения.

Ссылки арабскими цифрами во введении относятся к списку литературы, приведенному в конце диссертации. Ссылки римскими цифрами в квадратных скобках во введении и аннотациях к главам относятся к списку работ автора, включенных в диссертацию.

## § I. Очерк развития идей

В связи с тем, что работы автора, положенные в основу диссертации, относятся почти к двум третям всего периода развития программирования и в значительной степени были обусловлены общим ходом развития идей и методов, полезно, хотя бы очень схематично, рассмотреть основные вехи развития теории и автоматизации программирования. Автор не претендует на полноту или абсолютную объективность этого очерка, излагая события так, как они ему представлялись.

Классическая работа Дж. фон Неймана и Г.Голдстайка [1] не только впервые познакомила читателей с предметом программирования. Помимо массы конкретных сведений в работе был сформулирован ряд фундаментальных идей общего характера, из них главная – это идея расчленимости алгоритма на отдельные вычислительные акты с указанием логических связей между ними и, в связи с этим, понятие блок-схемы. Кроме того, хотя об этом и не говорилось явно, весь стиль изложения подчеркивал систематичность и поэтапность работы программирования при составлении программы. Первым шагом к автоматизации или, более точно, к рационализации труда программиста следует признать известную работу М.Уилкса, Д.Уилера и С.Гилла [2], в которой впервые были изложены принципы программирования на основе библиотеки стандартных программ.

Идеи фон Неймана и Голдстайна были углублены и развиты А.А.Ляпуновым [3]. В этой работе, заложившей основы развития советской школы программирования, главными оказались следующие положения:

I. Трактовка программирования как двуединого процесса формализации алгоритма и его перевода на язык машины.

2. Идея алгебраизации изобразительных средств, используемых в записи алгоритма.

3. Принцип расчленения алгоритма на операторы, классифицируемые по своим логическим и функциональным свойствам.

4. Понятие степени энтропийности процесса программирования и выделения в нем направленных действий и операций, требующих перебора.

5. Трактовка вычислительного процесса как последовательности преобразований состояния памяти.

Примерно к этому же времени относится работа Г.Рутисхаузера [4], в которой впервые с достаточной логической полнотой ставится проблема автоматизации составления программ методом трансляции и указываются некоторые, правда, довольно ограниченные средства к ее решению.

Фактическим началом развития автоматического программирования методом трансляции следует считать "программирующую программу ПП-2" [5], разработанную в СССР С.С.Камыниным и Э.З.Любимским с сотрудниками и "систему программирования ФОРТРАН" [6], созданную в США группой сотрудников корпорации ИБМ во главе с Дж.Бэкусом. Именно эти работы послужили отправной точкой и стимулировали бурное развитие трансляторов, которые стали сейчас наиболее эффективным средством автоматизации программирования.

Наряду с появлением этих новых систем автоматизации программирования были созданы первые шаги в теории преобразований схем программ как теоретической основы формальных методов улучшения качества программ при их автоматической трансляции. Главной работой этого направления явилась работа Ю.И.Янова [7], в которой впервые была очерчена и рассмотрена вся совокупность проблем, возникающих при формальных преобразованиях алгоритмов.

С самого начала наряду с методом трансляции родились и развивались другие подходы к автоматизации программирования. Составляющие программы для машины ИБМ 650 [8] и других машин ИБМ надолго определили стиль разработки систем "символического кодирования", направленных на мнемонизацию программирования в машинном языке и автоматизацию объединения библиотечных

программ с главной программой. В СССР на основе опыта работы с машиной М-2 [9] были заложены основы "малой автоматизации" программирования. В 1955 г. Л.В.Канторовичем и его сотрудниками были сформулированы принципы "блочного программирования"[10], давшего теории и практике программирования в СССР ряд важных идей:

1. Выделение информационных связей между компонентами вычислительного процесса в самостоятельное понятие.

2. Рассмотрение сложных информационных массивов как элементарных единиц информации.

3. Понятие списочной структуры информации.

4. Соединение интерпретации и трансляции в единый процесс реализации вычислительного плана.

Таким образом к 1956 г. завершился первый период развития программирования, который можно назвать периодом становления идей. Эти идеи получили свое воплощение и развитие в потоке работ, выполненных во втором пятилетии 50-х годов, завершающем "доалголовский" период программирования.

За рубежом ФОРТРАН наряду с системами символического кодирования стал доминирующим средством автоматизации программирования. В СССР был создан ряд программирующих программ, однако их внедрение было затруднено отсутствием средств для ввода и вывода буквенно-цифровой информации.[11,12] Методы малой автоматизации получили дальнейшее развитие в работах Е.А.Жоголева [13], В.В.Мартынюка [14], Ю.И.Морозова [65] и, в особенности, М.Р.Шура-Буры [15], чья интерпретирующая система легла в основу большинства систем автоматического использования библиотечных подпрограмм. На основе блочного программирования был создан ряд специализированных процессоров для аналитических выкладок и действий над полиномами [16,17]. Работа В.С.Королюка [18], в которой было введено понятие косвенной адресации и ранга адреса, породила в дальнейшем целое направление "адресного программирования", развивавшегося Е.Л.Ющенко [19] и ее учениками.

К концу 50-х годов увеличивается разнообразие типов трансляторов. С именем Р.Брукера [20] связано появление "автокодов", занимающих по своим возможностям промежуточное положение между трансляторами и системами символического кодирования.

Работами А.Перлиса [21] и,в особенности, Ф.Бауэра и К.Замельзона [22] были заложены основы методики построения одно-и двухходовых трансляторов, получивших за рубежом большое распространение благодаря своему быстродействию. При конструировании трансляторов в заметной степени стали выделяться в самостоятельные проблемы вопросы внутренней организации трансляторов и, в особенности, вопросы оптимизации транслируемых программ. Решение экономических задач и обработка символьной информации стали столь же равноправными классами задач, подлежащих решению на машинах, как задачи численного анализа. Для них были разработаны первые языки ФАКТ [23], КОБОЛ [24], ИПЛ [25] и ЛИСП [26] и трансляторы, положившие начало созданию самостоятельных направлений автоматизации программирования.

Теоретические работы в области программирования, хотя и не были столь эффектными, однако вносили свой вклад в математический фундамент и в формирование мировоззрения программистов.

В работах Н.А.Криницкого [27], Р.И.Подловченко [28] и А.А.Летичевского [29] было продолжено изучение вопросов формальных преобразований программ. В работах Дж.Маккарти [30], В. Ван дер Пула [31] и автора [32,33] были исследованы связи программирования с классической теорией алгоритмов,а также найдены системы "алгоритмических полных" операций, обеспечивающих реализацию любого алгоритма. Большое методологическое влияние на развитие теории оказала небольшая работа Л.А.Калужнина [34] о граф-схемном представлении алгоритмов, которая привлекла внимание к математическому рассмотрению программ с фиксированными логическими связями. Это рассмотрение привело к созданию "теории операторных схем", начало которой было положено работами В.В.Мартынюка [35], С.С.Лаврова [36] и Ю.И.Смирнова [64].

Независимо от того, как относиться к языку АЛГОЛ 60 [37], следует признать, что с его появлением наступил новый этап развития теории и автоматизации программирования и возникли новые точки роста идей и методов. Прежде всего, с появлением АЛГОЛа программирование стало по-настоящему международной наукой не только за счет использования АЛГОЛа в качестве языка публикаций, но и благодаря повышению общезначимости резуль-

татов, связанных с обсуждением или реализацией АЛГОЛа. Кроме того, богатство изобразительных средств, сравнительная трудность реализации, общий логический уровень языка, в особенности описание его синтаксиса, резко усилили роль теоретических исследований в конструировании трансляторов.

Первый цикл работ, появившихся в связи с АЛГОЛом, касался вопросов реализации механизмов языка. Был предложен ряд универсальных алгоритмов общей обработки АЛГОЛовских программ - Э.Дейстра [38] и программирования процедур - П.Ингерман [39], М.Р.Шура-Бура и В.И.Собельман [40]. Ряд вопросов реализации был описан в связи с конкретными трансляторами. Большую роль в распростанении АЛГОЛа в СССР сыграло создание трансляторов ТА-I [41] и ТА-2 [42], в котором был реализован табличный метод построения машинных команд и разработан метод моделирования страничной структуры при динамическом распределении памяти. В. Ван дер Пулом [43], Б.Рэнделлом и Л.Расселом [44] был разработан метод полуинтерпретационной трансляции с АЛГОЛа для малых и средних машин. П.Науром [45] был описан метод динамической сегментации программ.

Предложенный Дж.Бэкусом [46] метод формального описания синтаксиса АЛГОЛа с помощью металингвистических формул, заложил основу, с одной стороны, исследований, связавших анализ языков программирования с теорией формальных грамматик Хомского [47,48], и, с другой стороны, создания так называемых синтаксически управляемых трансляторов [49,50,51], в которых описание языка наряду с текстом программы в этом языке становится входом некоторого "мета-транслятора", производящего синтаксический анализ и перевод текста программы на основании описания языка. Проблема конструирования синтаксически управляемых трансляторов, в сочетании с обнаружением ряда неточностей в АЛГОЛе стимулировали появление работ [52,53], в которых предлагались различные способы формализации семантики языков программирования.

47-19



В настоящее время программирование вступило в фазу зрелости, став самостоятельной отраслью вычислительного дела с развитой методикой и обширной литературой, перевалившей за тысячу публикаций. Все чаще конструирование трансляторов и других систем программирования уже может ставиться как инженерная задача, когда центр тяжести переносится на проблему выбора и балансировки средств, предоставляемых теорией программирования, с целью удовлетворения заданным наперед эксплоатационным характеристикам. Наиболее всеобъемлющим результатом этого периода развития программирования является создание интегрированных операционных систем, объединяющих все средства программного обеспечения и автоматизирующих процесс прохождения задач через вычислительную систему в том числе и в режиме мультипрограммирования.

В то же время наступление этой фазы свидетельствует, что идеи 50-х годов, служившие до последнего времени движущей силой в развитии программирования, начинают исчерпывать себя. Это, однако, не означает, что программирование полностью становится технической наукой. Жизнь постоянно ставит перед нами новые задачи. Создание универсальных языков систем, способных с единой точки зрения синтезировать или вычленять конкретные языки общения с машиной, программирование параллельных вычислений, оптимизация програм на основе глобального анализа алгоритма, разговорное программирование, организация взаимодействия человека и машины, программирование в естественных языках – все эти новые проблемы ждут своих пионеров, которые возродят в программировании дух первооткрывательства, характерный для первых десяти лет истории этой новой сферы человеческой деятельности.

## § 2. Обзор результатов диссертации

Работы автора, включенные в диссертацию, относятся к теории преобразований програмы и к проблеме автоматизации программирования задач численного анализа методом трансляции с символических входных языков.

Главным результатом диссертации является обоснование и практическая реализация возможности конструирования уже для машин средней мощности трансляторов с входных языков с богатыми изобразительными средствами, дающих при приемлемой скорости трансляции рабочие программы высокого качества, сответствующего, в среднем, качеству массовой продукции квалифицированного программиста.

Конкретным выражением этого результата является система автоматизации программирования АЛЬФА [УП – IX] , разработанная в Вычислительном центре СО АН СССР в I96I–I964 годах и используемая сейчас во многих вычислительных центрах СССР. Этому конечному результату предшествовала серия работ [I–УI, X и XI], выполненых в течение I955 – I964 г. и в той или иной мере способствовавших созданию системы АЛЬФА. Некоторые работы, включенные в диссертацию [I–Ш], не имеют прямой связи с системой АЛЬФА и представляют самостоятельное значение, относясь к разделу диссертации, связанному с общими вопросами теории программирования. В целом осуждение работ удобно проводить, распределив их по трем разделам: теоретические основы трансляции, конструирование трансляторов и АЛЬФА-транслятор.

### 2.I. Теоретические основы трансляции

Процесс программирования можно рассматривать как процесс перевода (трансляцию) записи алгоритма в одном языке (входной

язык) в запись этого же алгоритма на другом языке (язык машины). Выражение "этот же" означает, что между алгоритмами во входном языке и машинными программами существует отношение эквивалентности, которое должно выполняться между исходным алгоритмом и его машинной программой. Принципиальной особенностью программирования является его неоднозначность: для любого алгоритма во входном языке существует целое множество машинных програм, эквивалентных этому алгоритму и, следовательно, между собой. На этом множестве реализаций алгоритма естественным образом определяются некоторые меры качества реализации, связанные, главным образом, со временем и объемом памяти, требуемыми для выполнения данного алгоритма. В связи с этим задача трансляции ставится как задача нахождения такой машинной реализации алгоритма, которая обладала бы наилучшими мерами качества по сравнению с другими эквивалентными реализациями. При таком подходе проблема трансляции расчленяется на проблему нахождения порождающего процесса, который бы мог фактически порождать множество эквивалентных реализаций, и проблему выбора оптимальной реализации из множества эквивалентных реализаций, создаваемых порождающим процессом. Будем условно называть первые проблемы алгебраическими проблемами и вторые проблемы - комбинаторными проблемами трансляции.

### 2.I.I. Алгебраические проблемы трансляции.

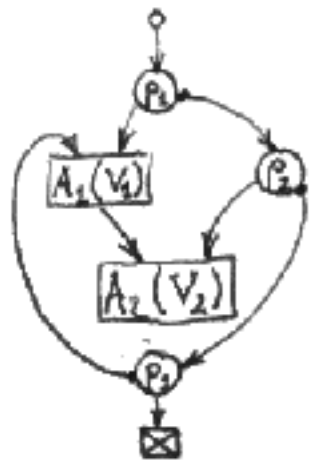
2.I.I.I. Одним из подходов к нахождению процесса, порождающего множество эквивалентных реализаций, является построение, при заданном определении эквивалентности, исчисления, или системы преобразований, полных в том смысле, что свободное применение их к исходному алгоритму позволяет трансформировать его в любой другой алгоритм, эквивалентный первому. В этом направлении автором была исследована известная работа Ю.И. Янова[7], в которой было описано исчисление, состоящее из I7 схем аксиом и трех правил вывода, которое давало полную систему эквивалентных преобразований для так называемых логических схем алгоритмов.

Автор попытался рассмотреть ( [I] , глава I.I.) теорию Янова в терминах граф-схем, введенных в рассмотрение Л.А.Калужниным[34]. При таком подходе основные определения теории схем алгоритмов выглядят следующим образом.

Операторной схемой Янова называется ориентированный граф с вершинами, имеющими не более чем по две выходящих из них стрелки

(в случае двух стрелок одна из них помечается), с одной (выходной) вершиной, не имеющей выходной стрелки, и с одной вершиной, в которую ведет извне специальная входная стрелка. Выходной вершине сопоставляется выходной оператор, вершине с одной выходной стрелкой сопоставляются различные операторы $A_1,\dots A_k$, а вершинам с двумя стрелками сопоставляются произвольные логические функции $n$ логических переменных $p_1, \dots, p_n$.

Пример




Каждому оператору $A_i$ ($i = 1,\dots,k$) сопоставляется некоторое подмножество $V_i$ переменных из числа $p_1,\dots,p_n$. Это сопоставление имеет тот смысл, что в результате выполнения $A_i$ переменные из $V_i$ могут получить произвольные значения. Это сопоставление по Ю.И.Янову называется распределением сдвигов.

Каждой операторной схеме $\mathcal{J}$ сопоставляется множество конфигураций. Каждая конфигурация имеет вид спаренной последовательности операторов схемы и наборов $\Delta$ значений логических переменных $p_1,\dots p_n$, получаемой при рассмотрении некоторого выполнения схемы $\mathcal{J}$ по следующему правилу. Берется произвольный набор $\Delta_{i_1}$. С этим набором начинаем двигаться по операторной схеме, начиная со входной стрелки. При попадании на оператор $A_{j_1}$ движение с набором $\Delta_{i_1}$ прерывается. При попадании на логическую фуницию $\alpha(\Delta)$ движемся по помеченной стрелке, если $\alpha(\Delta_{i_1}) = 1$, и по непомеченной в противном случае. После прохождения оператора $A_{j_1}$ набор $\Delta_{i_1}$ может трансформироваться в любой другой набор $\Delta_{i_2}$, но отличающийся от $\Delta_{i_1}$ значениями не более чем тех переменных из $p_1,\dots, p_n$, которые входят во множество $V_{j_1}$. Движение по схеме с набором $\Delta_{i_2}$ производится точно так же. Построение конфигурации, имеющей вид

$$\begin{matrix} \Delta_{i_1} & \Delta_{i_2} & \dots & \Delta_{i_k} & \dots \\ A_{j_1} & A_{j_2} & \dots & A_{j_k} & \dots \end{matrix}$$

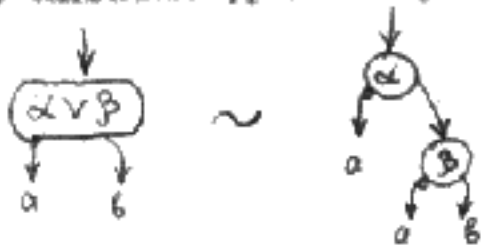
обрывается тогда, когда либо при движении встречается выход-

ной оператор, либо движение с некоторым набором $\Delta$ начинает происходить по замкнутому циклу, образованному одними логическими операторами. Две схемы Янова считаются эквивалентными, если они имеют одно и то же множество конфигураций.

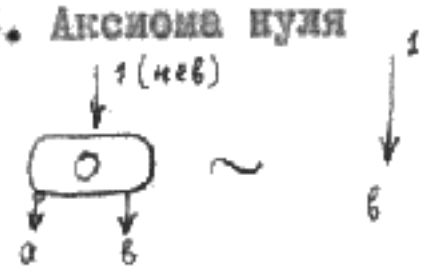
Это переформулирование основных определений, сохраняющее равнообъемность старых и новых понятий, позволяет существенно упростить аксиоматику, заменив 17 схем аксиом Янова следующими 6-ю аксиомами, имеющими вид соотношений между парами фрагментов операторной схемы (понятие фрагмента отличается от понятия подграфа тем, что в фрагменте указываются входные и выходные стрелки, соединяющие его с остальной частью схемы) в том смысле, что один фрагмент из пары можно заменить на другой с сохранением эквивалентности. Эти аксиомы суть:

1. Аксиома отрицания

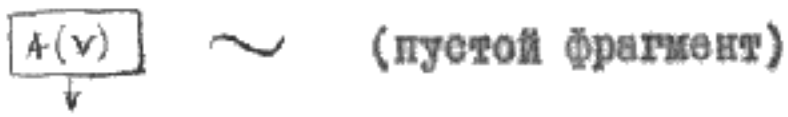

2. Аксиома дизъюнкции



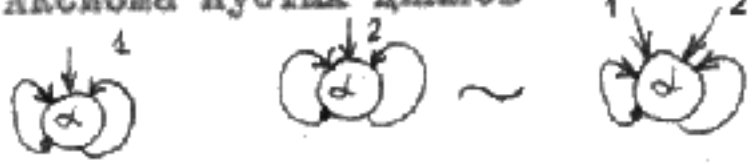

3. Аксиома нуля



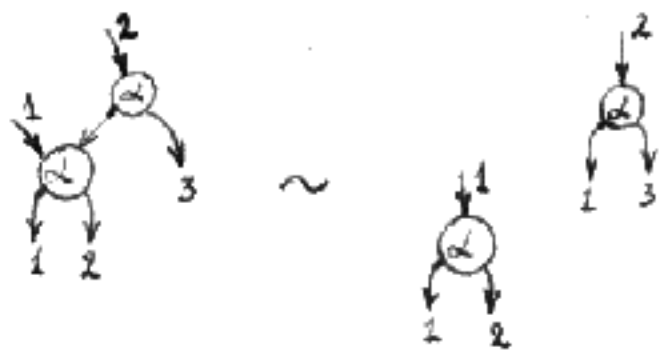

4. Аксиома неработающего оператора




5. Аксиома пустых циклов



## 6. Аксиома переноса плюс-стрелки



Первые два правила вывода, введенные Ю.И.Яновым, переносятся без изменений в аксиоматику операторных схем Янова:

I. Правило замены тождественным логическим оператором

$$\frac{\alpha \equiv \beta}{F(\alpha) \sim F(\beta)}$$

(Здесь $\equiv$ знак логической тождественности, $F(\alpha)$ – фрагмент схемы с выделенным вхождением логического оператора $\alpha$ ).

II. Правило замены эквивалентным фрагментом

$$\frac{F_1 \sim F_2,\quad F_3(F_1) \sim F_4}{F_3(F_2) \sim F_4}$$

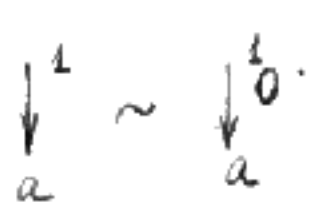
Вторым аспектом, изученным автором в теории операторных схем Янова, была аксиоматизация понятий логической подчиненности и распределения сдвигов. Дело в том, что в работе [7] 3-е правило вывода имело вид


Понятие логической подчиненности определяется содержательно: логический оператор $\alpha$ логически подчинен функции $\beta$ , если при построении любой конфигурации для схемы, содержащей логический оператор $\alpha$ , функция $\alpha$ вычисляется только на таких наборах $\Delta$ , на которых $\beta(\Delta)=1$. Функция $A_\alpha$ называется полным условием работы логического оператора $\alpha$ , если для любой функции $\beta$ , логически подчиняющей $\alpha$ , выполняется тождественно импликация $A_\alpha \to \beta$ .

Применение 3-го правила вывода требует знания полных условий работы логических операторов. В работе[7] эти полные условия находятся с помощью некоторого процесса неэквивалентных трансформаций схемы средствами, лежащими вне аксиоматики и описываемыми содержательно.

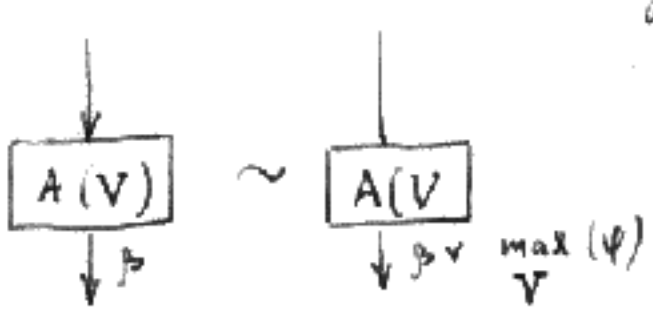
Автор заменил правило логической подчиненности 4-мя аксиомами и одним правилом вывода, которые аксиоматизируют процесс построения полных логических условий и основанного на них преобразования логических операторов. Аксиомы задают некоторый процесс разметки ребер операторной схемы логическими функциями, а правило вывода позволяет при выполнении условия "стационарности" разметки ребер, осуществить преобразование схемы. Условие стационарности формулируется в виде тождественного выполнения некоторых логических соотношений между функциями, метящими входы и выходы операторов.

Аксиомы разметки:

7. 


8. 


9. 

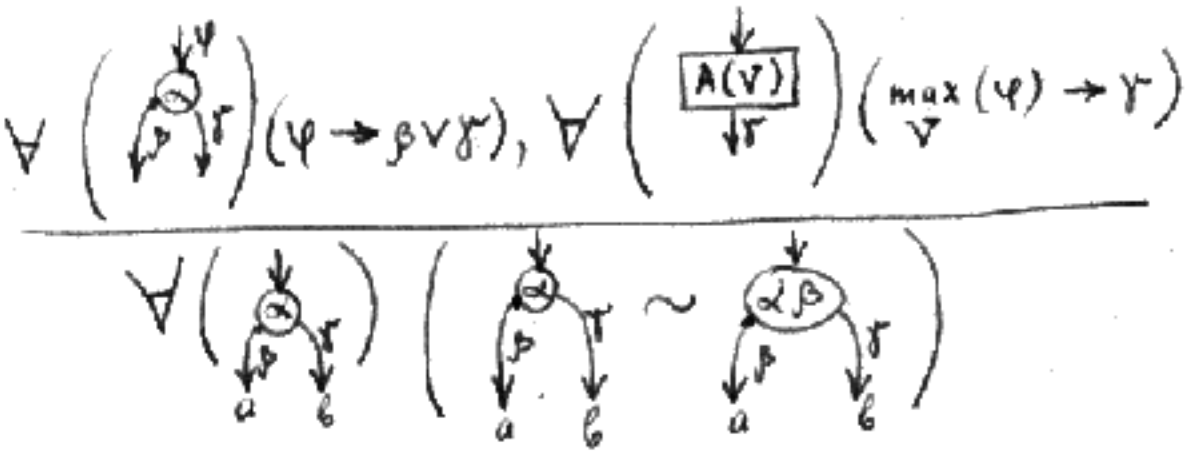

10. 


11.

47-26



III. Правило вывода:



Дополнительным последствием аксиоматизации понятий логической подчиненности и распределения сдвигов является то, что алгоритм распознавания эквивалентности операторных схем Янова становится тривиальным следствием из доказательства полноты системы аксиом и правил вывода.

2.I.I.2. Построение полной системы преобразований возможно только для весьма узкого определения эквивалентности. Например, в теории операторных схем Янова преобразуется только структура логических условий, обеспечивающих сохранение одного и того же порядка выполнения счетных операторов. Сами же операторы никаким преобразованиям в этой теории не подлежат. В то же время на практике программист, находясь в рамках более широкого определения эквивалентности, но не ставя вопрос о полной системе преобразований, желает осуществить более глубокое преобразование программы, но более ограниченным набором средств. В этом случае задача ставится по-другому: даны некоторые средства преобразований программ и требуется найти допустимые условия, обеспечивающие сохранение эквивалентности. С этих позиций автором была исследована задача о распределении памяти для хранения величин при составлении программ ( П, глава I.2).

В этой работе программы рассматривались в виде операторных схем, т.е. в виде произвольных ориентированных графов, вершинами которых являются операторы. Стрелки между операторами указывают на возможный порядок их выполнения. Каждый оператор имеет некоторое количество аргументов и результатов. Каждому аргументу или результату сопоставляется обозначение

величины (ячейки памяти), либо доставляющей оператору воспринимаемое им значение аргумента, либо берущей от оператора вырабатываемый им результат. Число различных величин в схеме называется ее весом. Средства преобразований схемы - это всевозможные переобозначения величин схемы. Главная задача - задача об экономии памяти в схеме, т.е. получения схемы, эквивалентной данной, но имеющей наименьший вес. Определение допустимых преобразований схемы базируется на понятии маршрута величины, введенного С.С.Лавровым [36], который впервые рассмотрел задачу об экономии памяти для одного частного вида операторных схем. Маршрутом величины $x$ называется такой путь в операторной схеме, что начальный оператор пути вырабатывает $x$, конечный оператор воспринимает $x$ и ни один из внутренних операторов пути не вырабатывает $x$. Зоной хранения некоторого оператора в схеме называется множество маршрутов величины, сопоставленной этому результату, имеющих своим началом данный оператор. Преобразование схемы $S$ в схему $S'$ называется допустимым, если зоны хранения любого результата любого оператора в схеме $S'$ являются расширениями соответствующих зон хранения результатов операторов в схеме $S$. По-существу определение допустимости является некоторым достаточным условием того, что в любой содержательной интерпретации схема $S'$ будет работать так же, как и схема $S$.

Сформулированное определение допустимости неконструктивно, поскольку его буквальная проверка требует анализа зон хранения, которые могут быть бесконечными множествами. Поэтому необходимо построение эффективного критерия допустимости преобразований. Этот критерий был найден в такой форме:

Для того, чтобы в операторной схеме все вхождения величины $x$ и величины $y$ можно было бы переобозначить одной и той же величиной, необходимо и достаточно, чтобы множество операторов схемы вида

$$H(x) \cap (H(y) \cup B(y)) \cup H(y) \cap (H(x) \cup B(x))$$

было пусто. Здесь $H(x)$ - множество операторов, вырабатывающих $x$, $B(x)$ - множество операторов, каждый из которых является внутренним оператором некоторого маршрута величины $x$. Множество $H(x)$ по схеме строится тривиально, для $B(x)$ были найдены достаточно простые алгоритмы его построения.

Таким образом для получения множества допустимых преобразований необходимо для всех величин схемы $x_1, \ldots, x_n$ построить матрицу бинарных отношений несовместимости: величины $x_i$ и $x_j$ являются несовместимыми, если для них вышеуказанное множество операторов непусто. Если трактовать величины $x_1, \ldots, x_n$ как вершины некоторого графа (графа несовместимости), а несовместимость величин $x_i$ и $x_j$ как наличие ребра, соединяющего вершины $x_i$ и $x_j$, то задача нахождения допустимых преобразований схемы становится хорошо известной комбинаторной задачей раскраски вершин графа; минимальная раскраска графа несовместимости даст оптимальную экономию памяти для исходной операторной схемы.

2.I.2. Комбинаторные проблемы трансляции.

2.I.2.I. В связи с задачей об экономии памяти, в простейшем случае сводящейся к задаче раскраски вершин графа, Г.И.Кожухиным и автором ([III], глава I.3) был исследован вопрос об оценке хроматического числа и о способах нахождения минимальных раскрасок графов. Были найдены неулучшаемые верхняя и нижняя оценки хроматического числа $\chi(n, p)$ любого связного графа с $n$ вершинами и $p$ ребрами, имеющие следующий вид

$$-\left[-\frac{n}{\left[\frac{n^2-2p}{n}\right]}\left(1-\frac{\left\{\frac{n^2-2p}{n}\right\}}{1+\left[\frac{n^2-2p}{n}\right]}\right)\right] \le \chi(n,p) \le \left[\frac{3+\sqrt{9+8(p-n)}}{2}\right].$$

Г.И.Кожухиным была доказана теорема, согласно которой для любой вершины $P$ в графе $\mathcal{G}$, не смежной со всеми остальными вершинами, на расстоянии двух ребер от $P$ судествует вершина $Q$, такая, что для графа $\mathcal{G}$ существует минимальная раскраска, при которой $P$ и $Q$ окрашены одной краской. На основании этой теоремы автором был предложен ряд беспереборных алгоритмов раскраски графов, описанных в совместной работе Л.Л.Змиевской, Р.Д.Мишкович, Л.К.Трохан и автора [IX](глава 3.3)

47-29



В основе этих алгоритмов лежит монотонное преобразование раскрашиваемого графа, состоящее в последовательном "склеивании" вершин графа, окрашиваемых одной и той же краской. Различие между алгоритмами лежит в различии стратегий выбора среди вершин, находящихся от вершины Р на расстоянии двух ребер, претендента на склеивание. Статистические эксперименты показали достаточную практичность этих алгоритмов.

2.I.2.2. Как известно из комбинаторики, выбор реализации путем беспереборных монотонных преобразований дает некоторые тупиковые реализации, как правило, не являющееся абсолютно наилучшими. Тем интереснее случаи, к сожалению, довольно редкие, когда может быть найдена такая цепочка монотонных преобразований, которая всегда даст оптимальный результат. В работе [VI] (глава 2.3.) автор исследовал вопрос об оптимальном программировании арифметических выражений с точки зрения минимизации числа рабочих ячеек, требуемых для программирования выражения. В этой работе выражение рассматривается как не более чем бинарное ориентированное дерево, корнем которого является заключительная арифметическая операция в выражении, граничными вершинами - остальные операции. Стрелка, идущая от операции А к операции В, означает, что результат операции В используется в качестве аргумента в операции А. Таким образом, стрелка в дереве может символизировать рабочую ячейку, в которой хранится промежуточный результат операции В. Стрелки в дереве задают отношение частичного порядка между вершинами. В связи с линейностью программ задачу программирования выражения можно трактовать как задачу упорядочивания вершин соответствующего дерева с сохранением исходной частичной упорядоченности. Если в некотором упорядоченном расположении вершин дерева провести сечение между двумя вершинами, то число пересекаемых при этом стрелок даст число рабочих ячеек, требуемых для сохранения промежуточных результатов, вычисленных "ранее" сделанного сечения и используемых "позже". Максимум по всем сечениям даст число рабочих ячеек, требуемых для программы вычисления выражения в соответсвии с заданным упорядочиванием. Возникает

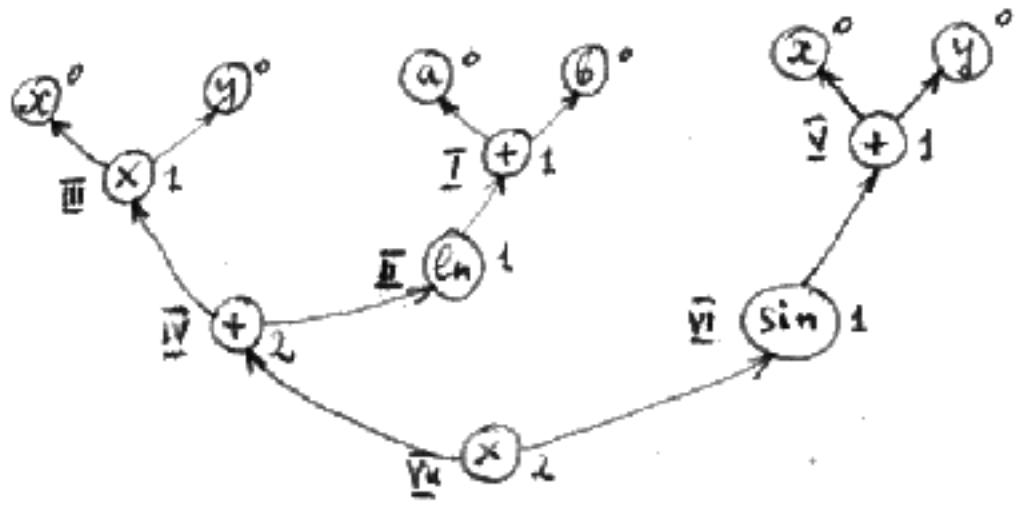
задача: найти такое упорядочивание вершин дерева, для которого максимум пересекаемых стрелок будет минимальным среди всевозможных упорядочиваний. Оказывается, что такая задача решается путем построения на дереве некоторой вершинной функции $f(P)$, строящейся по следующим рекуррентным соотношениям:

1) $f(P) = 0$, если P - граничная вершина дерева;

2) $f(P) = f(Q)$, если P имеет одну выходную стрелку, ведущую в $Q$;

3) $$f(P) = \begin{cases} f(Q) + 1, & \text{если } f(Q) = f(R) \\ \max(f(Q), f(R)), & \text{если } f(Q) \neq f(R) \end{cases}$$

для случая, когда P имеет две выходных стрелки, ведущих в $Q$ и . После построения этой функции порядка наивыгоднейшее упорядочивание вершин делается на основе следующего индуктивного определения: пусть определено положение вершины P, из которой выходят стрелки в вершины $R$ и $Q$. Тогда если $f(R) \geq f(Q)$, то все вершины, достижимые из $R$, должны предшествовать всем вершинам, достижимым из $Q$. Ниже показан пример дерева для выражения $(x \cdot y + \ln(a+b)) \times \sin(x+y)$. Арабскими цифрами показаны значения функции порядка и римскими - номера вершин в наивыгоднейшем упорядочивании.




В работе А.Я.Куртукова[57] показано, что эта методика распространяется на существенно более широкий класс выражений, нежели выше рассмотренные.

## 2.2. Конструирование трансляторов

Конструирование трансляторов является типичной задачей синтеза сложных управляющих систем. Главная трудность, которую необходимо преодолевать при построении транслятора, – это сочетание широты функционирования транслятора с возможно большей простотой его структуры, которая является залогом быстродействия и технологичности изготовления транслятора. Следует отметить, что сейчас еще нельзя говорить о существовании законченной и всеобъемлющей теории конструирования трансляторов. Обсуждаемые ниже результаты представляют лишь некоторую совокупность идей и методов, объединенных, правда, определенным подходом к задаче автоматизации программирования и достаточно хорошо апробированных на практике.

2.2.I. Входные языки трансляторов. Разработка входных языков трансляторов преследует две цели. С одной стороны необходимо построить языковую систему, обладающую точным синтаксисом и семантикой, имеющую по возможности, минимальный набор элементов языка и допускающую выполнение различных преобразований. С другой стороны входной язык должен обладать гибкостью и разнообразием изобразительных средств, позволяющих быстро и удобно записывать алгоритмы задач, подлежащие трансляции. Эти две цели во многом противоречат одна другой, особенно при конструировании так называемых универсальных языков. Было, однако, обнаружено, что эти противоречия в значительной мере можно снять, организовав должным образом процесс трансляции (см. ниже). Поэтому при построении входных языков автором было отдано предпочтение расширению изобразительных средств на основе исследования алгоритмов численного анализа и выделения в них устойчивых и часто встречающихся конструкций, с тем чтобы путем соответствующей формализации включить эти конструкции в состав входных языков.

Принципиальный шаг в этом направлении был сделан А.А.Ляпуновым[3], который впервые дал классификацию основных типов операторов при обосновании известного "операторного метода программирования". На основе этой классификации был разработан первый в СССР входной язык для программирующей программы ПП-2[5]

47-32

2.2.I.I. При разработке программирующей программы для БЭСМ автором был предложен дополнительный тип оператора, учитывающий циклический характер большинства алгоритмов ( [IУ] , глава 2.I.), и даны способы его автоматического программирования [X] . Этот тип операторов, получивших название операторов цикла, а настоящее время применяется в большинстве входных языков.

2.2.I.2. С начала 60-х годов основой для большинства вновь разрабатываемых трансляторов стал язык программирования АЛГОЛ 60[38], принятый в качестве стандартной основы для входных языков трансляторов. Этот язык, обладая развитой структурой, имеет сравнительно бедный запас элементарных вычислительных действий. В то же время анализ алгоритмов и изучение методики крупноблочного программирования[10,58] показали желательность наличия в развитом входном языке векторно-матричной символики. В связи с этим автором была предложена некоторая система обозначений и операций, позволяющая включать во входные языки для трансляторов действия над многомерными величинами ( [У] , глава 2.2.). Эта система была уточнена, расширена и "вписана" в АЛГОЛ 60 в совместной работе Г.И.Кожухина, Ю.М.Волошина и автора [XI] . Кроме введения многомерных величин в последней работе был описан еще ряд более мелких, но практически удобных, расширений АЛГОЛа 60: действия над комплексными числами, цепочки неравенств, описания начальных значений, функции-выражения, верхние индексы и т.д.

2.2.2. Быстродействие алгоритмов трансляции. Алгоритмы трансляции являются типичными алгоритмами посимвольной переработки информации и ее локальной перекодировки. Повышение быстродействия этих основных операций является существенным средством повышения скорости работы трансляторов.

2.2.2.I. В связи с задачей поиска информации автором в 1955 году был предложен способ "адресной кодировки информации" ([IУ] , глава 2.I.), при которой некоторая единица языка (например, идентификатор переменной) кодируется адресом, указывающим место хранения информации об этой единице языка. Этот способ, учитывающий принцип адресации в современных запоминающих устройствах, позволяет производить беспросмотровый поиск информации и в настоящее время широко применяется в трансляторах и других программах переработки информации.

2.2.2.2. В связи с задачей о перекодировке информации автором на одном конкретном примере было предложено решение этой задачи, позволяющее исключить квадратичные просмотры информации при выполнении перекодировки ( [УI] , глава 2.3.).

47-33



В общем случае предложенный метод получил название перекодировки с помощью "функций расстановки" и схематически состоит в следующем. Пусть $S$ – последовательность, образованная (с повторениями) элементами информации $a_1, \ldots, a_k$, подлежащими перекодировке и представляющими собой произвольную выборку из некоторого генерального множества A большой мощности ($k << \bar{A}$). Пусть $1, \ldots, n$ – запас номеров (кодов), в которые должны быть перекодированы элементы $a_1, \ldots, a_k$ ($k \le n$). Будем также трактовать номера $1, \ldots, n$ как адреса некоторого вначале пустого множества ячеек памяти $R$ (расстановочного поля), в которых могут храниться элементы последовательности $S$. Прямым подходом к перекодировке является такой, при котором элементы $S$ расставляются в ячейках $1, \ldots, n$ подряд и каждый очередной элемент сличается поочередно со всеми, уже хранящимися в памяти. При этом в качестве кода $a_i$ берется адрес этого элемента в $R$. Такая организация перекодировки даст время работы, пропорциональное $k^2$.

Предположим теперь, что на множестве A ($A = \{a\}$) определена "функция расстановки" $f(a)$, такая, что $1 \le f(a) \le n$. Перекодировка очередного элемента $a_i$ организуется теперь следующим образом. Пусть $F(a_i) = m$. Пытаемся поставить элемент $a_i$ в ячейку $m$. Если она пустая, то $a_i$ встретился в $S$ впервые и получает в качестве кода число $m$ и направляется в $m$. Если $m$ оказалось занятой таким же элементом $a_i$, то это значит, что $a_i$ встречается в $S$ повторно и просто заменяется на $m$. Если $m$ занята элементом, отличным от $a_i$, то, просматривая ячейки $m+1$, $m+2$ и т.д. (счет ведется по $\mathrm{mod}\, n$), находим первую ячейку M, либо пустую, либо занятую элементом $a_i$. В обоих случаях $a_i$ кодируется числом M, при этом в первом случае $a_i$ направляется в M. Тривиальная перекодировка с квадратичным поиском в расстановочном поле получается при $f(a) \equiv 1$. Оказывается, что если подобрать функцию расстановки так, чтобы на случайных выборках из A она давала бы значения, равномерно распределенные на $[1, n]$ и если при этом взять $n = 1{,}5k$, то в этом случае математическое ожидание времени работы перекодировки будет пропорционально $k$. Для различных конкретных

случаев перекодировки была найдена методика построения функции расстановки, базирующаяся на предельных законах распределения сумм ( по некоторому модулю) независимых случайных величин.

2.2.3. Организация трансляторов. Как уже указывалось, богатство изобразительных средств входных языков и их способность к формальным преобразованиям входят друг с другом в противоречие. Это противоречие фактически проявляется в том, что обилие синтаксических конструкций с богатой синонимией, во-первых, делают практически невозможной построение систем эквивалентных преобразований и, во-вторых, загромождают алгоритмы трансляции, которые, рассчитывая на общий случай, приводят к неэффективному программированию в большинстве конкретных ситуаций.

2.2.3.I. Для преодоления первого противоречия автором был предложен метод многофазной трансляции, состоящий в том, что между входным и машинным языком вводится еще один промежуточный уровень записи транслируемых алгоритмов, получивший название "внутреннего языка" ([У], глава 2.2). Поскольку внутренний язык не предназначен для записи алгоритмов человеком, его структура может быть полностью подчинена требованиям минимальности изобразительных средств и удобства выполнения анализа и различных формальных преобразований. При наличии внутреннего языка процесс трансляции производится следующим образом: сначала алгоритмы трансляции с богатого входного языка осуществляют перевод алгоритма на внутренний язык. После этого вступают в действие алгоритмы оптимизации, реализующие те или иные формальные преобразования, направленные на улучшение программы. После оптимизации действуют алгоритмы трансляции на машинный язык.

2.2.3.2. В связи со вторым противоречием автором была предложена идея "смешанной стратегии" трансляции на основе анализа транслируемой задачи ([УШ], глава 3.2). Суть ее состоит в том, что для программирования той или иной крупной единицы входного языка транслятор должен иметь не только общий универсальный алгоритм, но также располагать некоторым набором более простых алгоритмов, рассчитанных на те или иные более примитивные, но зато сравнительно часто встречающиеся синтаксические конструкции данной единицы языка. Выбор того или иного способа трансля-

ции делается на основе предварительного анализа транслируемого алгоритма.

2.2.3.3. Ряд дополнительных требований к организации трансляторов выдвигает обеспечение их быстродействия, оставаясь в рамках машин среднего класса со сравнительно небольшим объемом оперативной памяти. Автором был разработан ряд приемов ([УШ], глава 3.2), позволяющих более экономично использовать ограниченный объем оперативной памяти для размещения информации, возникающей в процессе трансляции.

В связи с созданием АЛЬФА-транслятора был разработан метод полудинамического распределения оперативной памяти для блоков транслятора, состоящий в том, что каждый блок перед своей работой производит распределение памяти, оставляя в ней только те таблицы, которые будут им использоваться, и отводя место для вновь организуемых таблиц (всюду,где это в принципе возможно) впритык, без запаса за счет предварительного просчета необходимых количественных характеристик транслируемой программы (например, число процедур, количество и глубина вложенности операторов цикла и т.п.). Таким образом каждый блок в процессе работы использует преимущества фиксированного и экономного распределения памяти, но от задачи к задаче распределение памяти меняется.

Для вновь образуемых информационных массивов неопределенного размера или с единицами информации различной длины автором был предложен механизм "общего списка", при котором все порции информации, относящиеся к разнородным массивам, выписываются на одно поле - общий список - вперемежку, а связь между порциями указывается с помощью ссылок. Механизм общего списка можно рассматривать как приспособление общеизвестного метода реализации в памяти "списочных структур" [25,26] к нуждам трансляторов и к условиям ограниченного объема памяти.

## 2.3. АЛЬФА- транслятор

Как уже указывалось, АЛЬФА-транслятор следует рассматривать как пример действующей системы программирования, реализующей большую часть идей и методов, излагавшихся в предыдущих

разделах обзора. Система АЛЬФА ([УП], глава 3.I) является коллективной разработкой, выполненной автором совместно с Г.И.Бабецким, М.М.Бежановой, Ю.М.Волошиным, Б.А.Загацким, Л.Л.Змиевской, Г.И.Кожухиным, С.К.Кожухиной, Р.Д.Мишкович, Ю.И.Михалевичем, И.В.Поттосиным и Л.К.Трохан.

АЛЬФА-транслятор, являющийся основной частью системы, транслирует программы, записанные в АЛЬФА-языке, в машинные программы для машины М-20. АЛЬФА-язык является конкретным представлением, разработанным Г.И.Кожухиным, И.В.Поттосиным и автором [59] на основе Входного языка для систем программирования [XI]. Третьей составной частью системы АЛЬФА является система отладки (АЛЬФА-отладчик), разработанная М.М.Бежановой и Ю.И.Михалевичем [60].

АЛЬФА-транслятор находится в производственной эксплоатации с 1964 г., демонстрируя удовлетворительные эксплоатационные характеристики. Средняя скорость трансляции - порядка 150 команд рабочей программы на минуту работы транслятора. Сравнение показало высокое качество трансляции, приближающейся к качеству ручного программирования: при сравнении с 17 программами, составленными квалифицированными программистами, среднее удлинение программы составило порядка 15%, а среднее увеличение времени работы - около 12% по сравнению с ручными программами.

В АЛЬФА-трансляторе был реализован принцип многофазной трансляции. На уровне внутреннего языка, разработанного автором ([УП], глава 3.2), осуществляется ряд оптимизирующих преобразований, в том числе экономия совпадающих выражений, чистка циклов и приведение индексных выражений к каноническому виду. Алгоритмы этих преобразований были разработаны М.М.Бежановой и И.В.Поттосиным [61,62]. На основе смешанной стратегии программирования были реализованы анализ и способы трансляции процедур - Б.А.Загацкий [63] и циклов - М.М.Бежанова и И.В.Поттосин [62]. Быстродействие транслятора обеспечивалось признаково-адресной кодировкой идентификаторов и символов языка, си[illegible]ческим применением функции расстановки, полудинамическим распределением памяти для блоков транслятора и механизмом "общего списка" ([УП], глава 3.2). В АЛЬФА-трансляторе Л.Л.Змиевской, Р.Д.Мишкович и Л.К.Трохан совместно с автором ([IX], глава 3.3) были реализованы алгоритмы экономии памяти на основе теории, разра-

ботанной в [II] и [III](главы I.2 и I.3).

## § 3. Заключение

В программировании, как и в любой бурно развивающейся отрасли науки, вовлекающей в свою сферу большое количество исследователей, часто одинаковые или аналогичные результаты возникают независимо и почти одновременно сразу в нескольких местах. Автор считает необходимым отметить ряд работ, выполненных независимо и содержащих результаты, аналогичные некоторым результатам диссертации.

Циклы в качестве отдельного вида операторов были независимо и, вероятно, практически одновременно введены во входной язык системы ФОРТРАН. Первая открытая публикация [6] появилась в 1957 г. через год после работы [IУ], однако впоследствии стало известно о существовании отчетов, относящихся к 1956 г.

Функция расстановки и адресная кодировка были использованы независимо и раньше появления работ [IУ] и [УI] в ряде составляющих программ для машин ИБМ. Автору впоследствии стало известно об этом из работ [8] и [54], относящихся к 1957 г.

Ряд предложений о включении многомерных массивов и операций над ними в АЛГОЛ 60 был независимо и несколько позже разработан Р.Хокни [55], чья публикация появилась примерно через год после работы [У].

Некоторые концепции, аналогичные многофазной трансляции и смешанной стратегии программирования, были независимо изложены в проекте многоходового транслятора Э.Хоукинса и Д.Хакстебла, описанной в работе [56], появившейся в конце 1963 г. К сожалению отсутствуют публикации, свидетельствующие о реализации этого интересного проекта.

47-38



В заключение обзора автор еще раз обращает внимание на то, что работы, на которых базируется диссертация, относятся к сравнительно длительному периоду, в ходе которого автору приходилось сотрудничать с большим числом лиц, работавших либо совместно с автором, либо под его руководством. Большая часть этих товарищей уже упомянута в обзоре или тексте диссертации. Однако появившаяся необходимость в связи с написанием обзора окинуть единым взглядом историю выполнения этой работы побуждает автора еще раз выразить свою признательность всем, кто своим трудом, советом или критикой способствовал успеху дела.

47-39

27

## СПИСОК ДИССЕРТАЦИОННЫХ РАБОТ

I. А.П.ЕРШОВ. Операторные алгоритмы Ш. (Об операторных схемах Янова). Сб.:"Проблемы кибернетики" (в печати).

II. А.П.ЕРШОВ. Сведение задачи распределения памяти при составлении программ к задаче раскраски вершин графов. Доклады АН СССР, Том I42, № 6 (I962).

Ш. А.П.ЕРШОВ, Г.И.КОЖУХИН. Об оценках хроматического числа связных графов. Доклады АН СССР, Том I42, № 2 (I962).

IУ. А.П.ЕРШОВ. Программирующая программа для БЭСМ АН СССР. Сб. Конференция "Пути развития советского математического машиностроения и приборостроения". Часть Ш. Москва I2-I7 марта, I956.

У. А.П.ЕРШОВ. Основные принципы построения программирующей программы Института математики Сибирского отделения Академии наук СССР. Сибирский математический журнал. Том 2, № 6 (I96I

УI. А.П.ЕРШОВ. О программировании арифметических операторов. Доклады АН СССР, Том II8, № 3 (I958).

УП. Г.И.БАБЕНКИЙ, М.М.БЕЖАНОВА, Ю.М.ВОЛОШИН, А.П.ЕРШОВ, Л.Л.ЗМИЕВСКАЯ, Б.А.ЗАГАЦКИЙ, Г.И.КОЖУХИН, С.К.КОЖУХИНА, Р.Д.МИШКОВИЧ, Ю.И.МИХАЛЕВИЧ, И.В.ПОТТОСИН, Л.К.ТРОХАН. Система автоматизации программирования АЛЬФА. Журнал выч. матем. и матем. физики. Том 5, № 2 (I965).

УШ. А.П.ЕРШОВ. Организация АЛЬФА-транслятора. Сб. "АЛЬФА-система автоматизации программирования", Новосибирск, I965.

IX. А.П.ЕРШОВ, Л.Л.ЗМИЕВСКАЯ, Р.Д.МИШКОВИЧ, Л.К.ТРОХАН. Экономия и распределение памяти в АЛЬФА-трансляторе. Сб. "АЛЬФА-система автоматизации программирования", Новосибирск, I965.

47 - 40



X. А.П.ЕРШОВ. Программирующая программа для быстродействующей электронной счетной машины, Москва, 1958.

XI. А.П.ЕРШОВ, Г.И.КОЖУХИН, Ю.М.ВОЛОШИН. Входной язык для систем автоматического программирования. Новосибирск, 1964.

47-41

# Л И Т Е Р А Т У Р А


I. H.H. Goldstein, J.von Neumann. Planning and coding problems for an electronic computing instrument. J.von Neumann, Collected works, vV, Pergamon Press, Oxford, I963

2. М.Уилкс, Д.Уилер, С.Гилл. Составление программ для электронных счетных машин. ИЛ, М., I953.

3. А.А.Ляпунов. О логических схемах программ. Проб. кибернетики, вып. I, ФМГиз, М., I958.

4. H.Rutishauser. Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen. Mittag N 3 aus dem Inst. Für angew. Mathem. der ETH Zürich. Verlag Birkhäuser, Basel, I952.

5. С.С.Камынин, Э.З.Любимский. Автоматизация программирования. Конф. "Пути развития советского математ. машиностр. и приборостр." ч.Ш, М., I956.

6. J.W. Backus, R.J.Beeber, S.Best et al. The FORTRAN automatic coding system. Proc. WJCC, I957.

7. Ю.И.Янов. О логических схемах алгоритмов. Пробл. кибернетики, вып. I, ФМГиз, М., I958.

8. СОАП 2. Сб. "Автоматизация программирования". ФМГиз, М., I96I.

9. Н.П.Трифонов, М.Р.Шура-Бура. Опыт эксплоатации машин М-2 в вычислительном центре МГУ. Конф. "Пути развития советского математ. машиностр. и приборостр." ч.Ш, М., I956.

I0. Л.В.Канторович, Л.Т.Петрова, М.А.Яковлева. Об одной системе программирования. Там же, где 9.

II. Т.М.Великанова, А.П.Ершов, К.В.Курочкин, Ю.А.Олейник, В.Д.Поддерюгин. Программирующая программа для машин. Сб. "Труды Бакинской конф. по вычислит. матем. и применению средств вычислит. техники." АН АзССР, Баку, I96I.

I2. М.Р.Шура-Бура, Н.П.Трифонов (ред.). Система автоматизации программирования. ФМГиз, М., I96I.

I3. Е.А.Коголев. Система программирования с использованием библиотеки подпрограмм. Сб. "Система автоматизации программирования". ФМГиз, М., I96I.

I4. В.В.Мартынюк. Программа автоматического присвоения адресов (ПАПА) для М-20. Сб. "Библиотека стандартных программ". ЦБТИ, М.,I96I.

47-42


I5. М.Р.Шура-Бура. Система интерпретации ИС-2. Там же, где I4.

I6. Л.Т.Петрова. О проведении аналитических выкладок на машинах с программным управлением. Изв. ВУЗов, Математика, № 5, I958.

I7. Т.Н.Смирнова. Полиномиальный пробег на "Стрелу". Тезисы докл. 2-й научной конф. по вычислит. матем. и вычислит. техн. ВЦ АН УССР, Киев, I960.

I8. В.С.Королюк. О понятии адресного алгоритма. Пробл. кибернетики, вып.4, ФМГиз, М., I960.

I9. Е.Л.Ющенко. Адресное программирование. ГИТЛ УССР, Киев, I963.

20. R.A. Brooker. The Autocode programs developed for the Manchester university computer. Comp. Journ., I, N I, I958.

2I. А.Дж.Перлис, Дж.В.Смит, Г.Р.Ван-Зорен. ИТ. Сб. "Автоматизация программирования", ФМГиз, М., I96I.

22. F.L.Bauer, K. Samelson. Sequentielle Formelubersetzung. Electron. Rechenanlagen, I, N 4, I959.

23. R.F.Clippinger. FACT - a business compiler. Annual Review in Autom. Progr., v.2, Pergamon Press, Oxford, I96I.

24. J.E. Sammet. A detailed description of COBOL. Ibid.

25. J.C.Shaw, A.Newell, H.A.Simon, T.O.Ellis. A command structure for complex information processing. Proc. WJCC, I958.

26. J.McCarthy. LISP: a programming system for symbolic manipulation. Proc. I4th Nat. meeting ACM, I959.

27. Н.А.Криницкий. Формальные преобразования спрямляемых схем программ. Диссертация, МГУ, I959.

28. Р.И.Подловченко. О преобразованиях схем программ и их применении в программировании. Проб. кибернетики, вып. 7, М.I962.

29. В.С.Королюк, А.Л.Летичевский. Об одном классе адресных алгоритмов. ДАН УССР, № 2, I959. (укр.)

30. J.McCarthy. Recursive functions of symbolic expressions and their computation by machine. ACM Comm., 3 , N 4, I960.

47-43

31. W.L.van der Poel. The essential types of operations in an automatic computer. Nachrichtentechn. Fachber., 4, 1956.

32. А.П.Ершов. Операторные алгоритмы. I (Основные понятия). Проб. кибернетики, вып. 3, ФМГиз, М., 1960.

33. А.П.Ершов. Операторные алгоритмы. II (Описание основных конструкций программирования). Проб. кибернетики, вып. 8, ФМГиз, М., 1962.

34. Л.А.Калужнин. Об алгоритмизации математических задач. Проб. кибернетики, вып. 2., ФМГиз, М., 1959.

35. В.В.Мартынюк. Выделение цепей в схеме алгоритма. Ж.вычисл. матем. и матем. физики, 1, № 1, 1961.

36. С.С.Лавров. Об экономии памяти в замкнутых операторных схемах. Ж. вычисл. матем. и матем. физики. 1 , № 5, 1961.

37. J.W. Backus, F.L.Bauer, J.Green et al. Revised report on the algorithmic language ALGOL 60. International federation for information processing. 1962.

38. E.W.Dijkstra. Recursive programming. Numerische Mathematik, 2, N 5, 1960.

39. P.Z.Ingerman. Thunks. Comm. of the A.C.M., 4, n 1, 1960.

40. В.И.Собельман, М.Р.Шура-Бура. Реализация рекурсивных процедур в языке АЛГОЛ 60. Ж. вычисл. матем. и матем. физики 2, № 2, 1962.

41. В.Н.Попов, В.А.Степанов, А.Г.Стишева, Н.А.Травникова. Программирующая программа. Журн. вычислит. матем. и матем. физики, 4, № 1, 1964.

42. М.Р.Шура-Бура, Э.З.Любимский. Транслятор АЛГОЛ 60. там же, где 41.

43. W.L.van der Poel, G.van der Mey, P.A. Witmans, G.G.M. Mulders. Process for an Algol translator. Dr. Neher laboratorium, Report 164 MA, July 1962.

44. B.Randell, L.Russel. ALGOL 60 implementation. Academic Press, N.Y., 1964.

45. P.Naur. The design of the GIER ALGOL compiler
Ann. Review in Autom. Progr. v.4, Pergamon Press, Oxford, I963.

46. J.Backus. The syntax and semantics of the proposed international algebraic language of the Zurich ACM - GAMM conference. ICIP Paris, June I959.

47. R.W.Floyd. On syntactic analysis and operator precedence. Computer Associates, Report CA-62-2, August I962.

48. K.Čulik. Formal structure of ALGOL and simplification of its description. Coll. "Symbolic languages in data processing". Gordon and Breach, N.Y.,I962.

49. P.Ingerman. A translation technigue for languages whose syntax is expressible in extended Backus normal form. Ibid. as 48.

50. E.T.Irons. The structure and use of the syntax directed compiler. Ann. revier in autom. progr., v. 3, Pergamon Press, Oxford, I963.

5I. R.A.Brooker, D.Morris. A general translation program for phrase structure languages. Journ. of the A.C.M., 9, NI, I962.

52. J.McCarthy. A formal description of algorithmic languages. Working Conference "Formal language description languages", Wiena, I964.

53. A.van Wijngaarden. Orthogonal design and description of a formal language. Mathematisches Centrum, MR 76, Amsterdam, October I965.

54. W.W. Peterson. Addressing for random-access storage. IBM Journ res. and developm., I, N 2, I957.

55. R.Hockney.An extension to ALGOL 60 for industrial use. Comp. Journ. 4, N 4, I962.

56. E.N.Hawkins, D.H.R. Huxtable. A multi-pass translator scheme for ALGOL 60. Ann. Rewiew in autom. progr. v.3, Pergamon Press, Oxford, I963.

57. А.Я.Куртуков. Об оптимальном расположении графов. (в печати).

58. В.А.Булавский. О символике записи вычислительных планов при автоматизации программирования. Изв. ВУЗов, Математика. 5 № 5, I958.

47-45

59. А.П.Ершов, Г.И.Кожухин, И.В.Поттосин. Обзор особенностей АЛЬФА-языка. Сб. "АЛЬФА- система автоматизации программирования". Сиб. отд. АН СССР, Новосибирск, 1965.
60. М.М.Бежанова, Ю.И.Михалевич. АЛЬФА-отладчик. Там же, где 59.
61. И.В.Поттосин. Экономия выражений в АЛЬФА-трансляторе. Там же, где 59.
62. М.М.Бежанова, И.В.Поттосин. Программирование циклов и индексных выражений в АЛЬФА-трансляторе. Там же, где 59.
63. Б.А.Загацкий. Программирование процедур в АЛЬФА-трансляторе. Там же, где 59.
64. Ю.И.Смирнов. О построении матрицы запретов операторной схемы с обращениями к библиотеке. Журн. вычисл. матем. и матем. физ. 4, № 1, 1964.
65. Ю.И.Морозов. О стандартных программах; программирующая система компилирующего типа (ПСК-1). Пробл. кибернетики, вып. 5, ФМГиз., М., 1961.