А.П.ЕРШОВ И СТАНОВЛЕНИЕ НОВОСИБИРСКОЙ ШКОЛЫ ПРОГРАММИРОВАНИЯ

Поттосин[1] И.В.
г. Новосибирск, Россия

Становление ведущих научных школ по информатике у нас в стране практически всегда было связано с именами ярких личностей. Если у истоков киевской школы стоял В.М.Глушков, а у истоков ИПМовской школы - А.А.Ляпунов и М.Р.Шура-Бура, то у истоков новосибирской школы стоял их более молодой соратник А.П.Ершов, ученик А.А.Ляпунова.

Сибирское отделение АН СССР было создано в 1957 году, а год спустя, осенью 1958 года было создано ядро будущей новосибирской школы программирования - отдел программирования Института математики СО АН СССР, фактическим, а потом и формальным руководителем которого стал А.П.Ершов. Несмотря на свои молодые годы А.П.Ершов к этому времени уже стал известным специалистом в области трансляции и одним из лидеров этого направления в стране - он был руководителем двух проектов программирующих программ, ПП БЭСМ и ППС, автором первой монографии по трансляции. Набираемый им коллектив был тоже молодежным, что, впрочем, не удивительно: подавляющее большинство всех программистских коллективов было молодо, новая наука бурно втягивала в себя молодежь. Первыми сотрудниками отдела программирования стали выпускники или недавние выпускники университетов Москвы, Ленинграда, Томска, Риги, Саратова.

Первым большим проектом, с которого начиналась новосибирская школа, был трансляторный проект "Альфа". В отличии от предыдущих проектов программирующих программ проект Альфа был хорошо, практически по современному специфицирован: было подготовлено формальное описание входного языка [1,2], была создана развернутая функциональная спецификация системы [3] .

Отправной точкой проекта была публикация начальной версии нового языка программирования, суммировавшего накопившийся программистский опыт и создаваемого международной рабочей группой - так называемого Алгола 58. Группа, руководимая А.П.Ершовым, стала готовить на основе Алгола 58 новый проект языка - параллельно с работавшей международной группой. Во многом направления развития языка оказались совпадающими, но в новосибирском проекте появился ряд существенно новых механизмов, поэтому в конце концов язык был сформулирован как правильное расширение окончательной версии международного языка - Алгола 60. В Альфа-языке впервые были разработана средства, характерные для последующих за Алголом 60 языков. Было определено столь важное для вычислительных алгоритмов понятие многомерных значений, определены операции над ними, в том числе,

[1] Институт систем информатики им. А.П.Ершова СО РАН и Новосибирский университет.

их конструирование. Были введены свойственные современным языкам концепции, такие, как разнообразие видов циклов, задание начальных значений и т.п. В формальном определении языка впервые была сделана попытка выйти за пределы контекстно-свободных грамматик.

Сама работа над транслятором была в большой мере и исследовательской, и пионерской. Именно в этой работе начались складываться принципы современной оптимизирующей трансляции. Дело в том, что сам Алгол 60 был определенным вызовом для существовавших методов трансляции. Его существенно рекурсивная структура, мощность механизма процедур, возможная их вложенность и потенциальная рекурсивность, общность циклов и индексных выражений - все это требовало заметной модификации и развития техники трансляции, а самое главное - ставило вопрос о возможности получения эффективного кода. По последнему поводу высказывалось ряд сомнений, и Альфа-транслятор стал действительно первым конструктивным доказательством того, что для языков, сопоставимых по мощности с Алголом, можно построить транслятор, дающий сравнимый с ручным программированием код.

Достижению поставленной цели послужил богатый набор оптимизаций, реализованный в Альфа-трансляторе [4,5]. Была предложена смешанная стратегия программирования (то, что позднее на Западе было названо "casing") таких конструкций, как процедуры, циклы, индексные выражения, когда на основании анализа контекста выбирался наиболее эффективный из допустимых способов генерации конструкций. Особенно изощренно программировались процедуры и подстановка параметров для них: выбор осуществлялся среди 11 способов. В результате алголовские процедуры при всей мощности механизма программировались оптимальным образом (что стимулировало программистов активно использовать это средство). Существовавшая ранее оптимизация экономии выражений была существенно развита - полностью учитывались свойства коммутативности и ассоциативности; если это преобразование не стало глобальным, то оно уже было квазилокальным. Впервые была реализована глобальная чистка циклов. Тщательной оптимизации подвергались операции над многомерными значениями - то, что не было в стандартном Алголе, но активно использовалось вычислителями. Впервые была реализована глобальная экономия памяти, опирающаяся на теоретические работы А.П.Ершова и С.С.Лаврова.

Альфа-транслятор стал первым в мире транслятором с Алгола с большими оптимизирующими возможностями. Похожий английский проект Хоукинса и Хакстебла, который разрабатывался в это время, так и не был до конца завершен из-за сложности задачи. Конечно, успеху способствовало не только механическое соединение многих оптимизаций, но и существенное развитие существовавшей методологии оптимизации программ. Была выдвинута идея внутреннего языка, к программе на котором и применяются оптимизации, которая, хотя и не совсем в чистом виде, была реализована. Был реализован практически глобальный анализ контекста, хотя и разрозненно для различных оптимизаций, учет которого существенно увеличил мощность оптимизаций - зародыш современного потокового анализа. Стремление

практически всегда получать эффективный код привело к отказу от некоторых средств Алгола, важнейшим из которых была рекурсивность процедур.

Альфа-транслятор активно использовался в большом числе организаций страны. Хотя его интерфейсы с пользователем и простота эксплуатации желали много лучшего, но высокая эффективность получаемого кода обеспечила хороший интерес пользователей к этой системе. Поэтому когда на смену М-20 пришла существенно более высокопроизводительная машина БЭСМ-6, организации, связанные с большими задачами, стимулировали создание версии Альфа-транслятора для БЭСМ-6. Самым быстрым решением была простая переделка Альфа-транслятора, при которой та его часть, которая отвечала за генерацию кода, была заменена генератором кода для БЭСМ-6. Возник транслятор Алгибр (“Альфа гибридный”), который был одним из первых отечественных кросс-трансляторов. Такое решение имело ряд недостатков, но зато оно было реализовано быстро, и транслятор Алгибр стал первым транслятором для новой машины БЭСМ-6. Решение существенно облегчалось тем, что в Альфе-трансляторе уже существовал внутренний язык, хотя и не до конца единый для всех оптимизирующих преобразований.

Основным затруднением при Алгибр-трансляции была передача оттранслированного кода с М-20 на БЭСМ-6. В ВЦ СО АН СССР был даже разработан специальный канал передачи информации между инструментальной и объектной ЭВМ, однако это не могло быть серийным решением при массовой трансляции. Возникла необходимость "родного" транслятора для БЭСМ-6.

У разработчиков Альфа-транслятора возникала и внутренняя потребность модифицировать и улучшить как схему трансляции, так и пользовательские свойства транслятора, в первую очередь, его интерфейс с пользователем. Альфа-транслятор, несмотря на его достаточно широкую используемость, носил черты эксперимента и поиска - хотя идейные решения были найдены верно, но реализационные решения были характерны скорее для экспериментального, чем производственного транслятора (большое число просмотров программы - 24, некоторое дублирование анализа контекста, не всегда оправданное усложнение оптимизирующих преобразований). Не нужно забывать, что этот транслятор был первым оптимизирующим транслятором с Алгола. Дополнительный опыт - транслятор Алгибр, незавершенный проект Альфа-транслятора для ЭВМ Урал-14 (проект ТАУ) - давал основу для новых решений.

Новый транслятор - транслятор Альфа-6 [6] - был улучшенной версией оптимизирующего транслятора с Алгола. Ряд предыдущих решений был приведен к более чистому и эффективному виду. Более четко был выделен внутренний язык и этап потокового анализа программ на внутреннем языке. Общая схема трансляции уже могла рассматриваться как типовая схема оптимизирующей трансляции (для одноязыкового транслятора), она насчитывала всего 10 просмотров. Лучше была решена проблема взаимного влияния различных оптимизирующих преобразований. Идентификация и визуализация ошибок пользователя была более совершенной, что облегчало эксплуатацию. В результате система Альфа-6 стала достаточно широко используемой системой у пользователей БЭСМ-6.

В середине 60-х годов начало складываться новое направление системного программирования - операционные системы. Причины их возникновения связаны и с усложнением архитектур ЭВМ и с совершенствованием методов взаимодействия человек - машина. В архитектурах появились параллельно работающие устройства и возникла задача управления их совместной работой. Назрела необходимость автоматической смены задач, диалоговой связи с оператором ЭВМ и, что еще важнее, с пользователем. Появилось понятие режима разделения времени, обеспечивающее многозадачность. Как зарубежные, так и отечественные программисты ринулись в эту новую область, закладывая основы методологии современных операционных систем.

А.П.Ершов с его прекрасным чувством нового и перспективного стал инициатором и руководителем проекта “автоматической информационной станции” АИСТ-0 [7,8] . Этот проект сочетал в себе как создание многопроцессорной архитектуры, так и развитого программного обеспечения, использующего те возможности, которые предоставляла эта архитектура.

АИСТ-0 стал одной из первых отечественных многопроцессорных систем. В системе существовал управляющий процессор и несколько рабочих. В качестве процессоров были взяты серийно производимые отечественные ЭВМ - управляющим процессором был взят Минск-2, а рабочим процессором М-220. Внешняя и оперативная памяти рабочих процессоров были обобществлены и подключались через коммутатор. Для фиксации событий и для программируемых прерываний существовала система прерываний. Архитектурными и программными средствами была сохранена полная совместимость с существующим программным обеспечением для М-220.

Программное обеспечение АИСТа-0 было богатым и разнообразным, его создатели хотели использовать все предоставившиеся возможности. Существовало несколько специализированных операционных систем пакетного режима, которые могли образовывать несколько потоков задач, исполняемых по различным дисциплинам обслуживания, учитывающими важность, срочность и прочие пользовательские требования. Были созданы многопользовательские системы диалоговой отладки и редактирования, системы диалогового программирования как для стандартных языков (Алгол 60), так и для специальных диалоговых языков (известный в то время ДЖОСС). Как пример применения диалога в прикладных программах были разработаны система аналитических преобразований и интеллектуальный пакет для линейной алгебры. Было создано несколько информационно-поисковых систем. По-видимому, впервые в стране было разработано несколько диалоговых игровых программ.

Создание столь разнообразных систем облегчалось разработанной структурой программного обеспечения [9]. Как и другими разработчиками операционных систем разделения времени и реального времени решалась задача выделения ядра операционной системы. Такое ядро (в АИСТ-0 оно называлось диспетчер) обеспечивало создание и управление процессами, передачу сообщений между процессами, первичную обработку сигналов и прерываний, реализовывало некоторую универсальную дисциплину управления

многопроцессностью, на базе которой за счет задания параметров можно было реализовать удобную для данного вида работы дисциплину обслуживания. Следующий слой программного обеспечения составляли так называемые системные программы - специализированные на вид обслуживания операционные системы. Диспетчер работал на управляющей машине, все остальное программное обеспечение - на рабочих процессорах. Такая структура облегчала создание множества режимов и видов обслуживания, каждый из которых реализовался своей системной программной с использование базовых понятий и средств диспетчера. Конечно, можно было попытаться найти общность в некоторых классах режимов и видов обслуживания, но отсутствие реального опыта разнообразной работы делал эти попытки умозрительными, и от них сознательно отказались, сосредоточив все общие модели в диспетчере, что, по-видимому, удалось сделать довольно удачно.

Достаточно успешная реализация АИСТа-0 вызвала к жизни проект серийного производства многопроцессорных комплексов, который прорабатывался Казанским заводом ЭВМ, однако общий поворот в идеологии отечественного производства ЭВМ не дал этому проекту осуществиться. Часть идей АИСТа-0, но далеко не полностью, была использована при создании ЭВМ М-222.

Заглядывая в более поздние времена, надо сказать, что опыт, накопленный при архитектурной и программной разработке АИСТа-0, сказался и в последующих новосибирских разработках - макете потокового суперкомпьютера МАРС, первой отечественной 32-битной рабочей станции Кронос, специализированной рабочей станции для редакционно-издательской деятельности МРАМОР. Следует отметить дух создания новосибирских архитектур - они проводились под сильным влиянием системных программистов. Так, архитектура МАРС разрабатывалась под влиянием языка асинхронного потокового программирования Барс, а в архитектуре Кроноса были заложены средства схемной реализации конструкций языка программирования высокого уровня, прежде всего - Модулы-2.

В качестве непосредственного продолжения проекта АИСТ-0 был начат проект создания системы коллективного пользования для новой тогда ЭВМ БЭСМ-6, руководителем разработки программного обеспечения в котором был Г.И.Кожухин. Проект этот пришлось прекратить ввиду организационных трудностей, связанных с требуемой проектом модификации БЭСМ-6, однако он повлиял на следующий реализованный проект ВЦКП (Вычислительного Центра Коллективного Пользования), прообраза гетерогенных локальных вычислительных сетей.

Работы по АИСТ-0 не исключали и продолжение исследований и в уже традиционном для новосибирской школы направлении языков программирования и трансляции. Параллельно с работами по АИСТ-0 новосибирцами были созданы одни из первых языков системного программирования - Эпсилон и Сигма. Естественным было взаимодействие между этими работами и проектом АИСТ-0 - так, все системное программное обеспечение АИСТа-0 было написано на Эпсилоне. Языкотворчество осталось хорошей традицией новосибирской школы - позднее новосибирцами был

реализован ряд интересных и оригинальных языковых проектов: язык системного программирования второго поколения Ярмо, язык программирования баз данных Бояз, язык параллельного потокового программирования Барс, язык описания и проектирования архитектуры Поляр.

Накопленный в оптимизирующей трансляции опыт дал возможность перейти к следующему новосибирскому трансляторному проекту - многоязыковой транслирующей системе БЕТА. Цели проекта были весьма амбициозные, недаром название проекта расшифровывалось некоторыми коллегами по профессии как Большая Ершовская Трансляторная Авантюра. Предполагалось, что будет создана отрытая транслирующая система с высоким уровнем глобальной оптимизации программ, охватывающая практически весь тогдашний класс императивных языков высокого уровня - от ФОРТРАНа до Алгола 68 и ориентированная на получение программ для большинства существующих архитектур. Речь шла о создании мощной базовой системы трансляции, одноязыковые трансляторы в которой выглядели как частный случай, подсистема. Забегая вперед, нужно сказать, что в принципиальном отношении задача была решена.

Для создания подобной системы определяющим было решение трех проблем - разработка общего внутреннего языка как семантического базиса широкого класса входных языков, выработка универсальной схемы языково-независимой оптимизации программ, разработка технологии включения новых входных языков в единую транслирующую систему. Все эти проблемы носили весьма серьезный характер, поэтому проект планировался как комплекс длительных методологических и экспериментальных исследований. Он выполнялся более 10 лет, задание на систему было опубликовано в 1971 г. [10], а итоговая публикация появилась только в 1982 г. [11].

Начальный этап проекта в значительной мере определялся исследованиями по внутреннему языку. Он мыслился как семантический базис, унифицирующий языковые средства входных языков, как основа оптимизации программ и как общее исходное представление программ для генерации кода. В начальных проектах внутреннего языка стремились найти ортогональные средства выражения языковых конструкций, разложив их на небольшое число семантически элементарных средств, выразить внутренний параллелизм etc. Эти проекты были интересны в теоретическом отношении, но, к сожалению, оказались непрактичны. Окончательная версия внутреннего языка была создана как объединение абстракций общих понятий и конструкций широкого класса входных языков с включением не полностью интерпретируемых языково-зависимых конструкций. Версия эта была построена на анализе большого числа существовавших тогда языков программирования - Алгола 60, ПЛ/1, Симулы-67, Алгола 68, Паскаля (ФОРТРАН игнорировался из-за наличия в нем рудиментарных конструкций, Кобол же по существу никогда не был в поле зрения отечественных программистов), на выделении общностей этих языков - в системе типов, в управляющих конструкциях, на построении абстракций, как отражающих эти общности, так и дающих простор для включения специфических средств данного языка. Вся эта работа была частью общемировых исследований по внутренним языкам и помимо своей

конструктивной роли для БЕТА-проекта обогатила понимание общей содержательной семантики существующих языков программирования.

Оптимизирующая трансляция в системе БЕТА реализуется специальным оптимизирующим процессором, осуществляющим глобальные оптимизирующие преобразования программы на внутреннем языке. Оптимизирующее преобразование понимается как пара - условие применимости преобразования и собственно схема трансформации в терминах внутреннего языка. Условия применимости большинства преобразований существенно используют управляющие и информационные зависимости между операторами программы, поэтому в оптимизирующем процессоре выделяется начальный этап потокового анализа. В результате его работы над операторами внутреннего языка надстраивается гамачно-цикловая иерархическая структура, а каждый оператор и компонента такой структуры снабжается списками аргументов и результатов. Как условия применимости, так и трансформации определяются в терминах введенной иерархии, кроме того иерархия используется для факторизации глобальных оптимизаций.

Собственно трансформирующая часть оптимизируюшего процессора построена как открытый набор преобразований. Был выбран некоторый порядок их исполнения, дающий наиболее максимальный эффект их совместного применения с учетом их взаимозависимости. Ряд оптимизаций был существенно обобщен - чистка циклов реализовывалась не только за счет выноса перед циклом фрагментов выражений, но и за счет выноса как перед, так и после цикла операторов целиком, а также целых гамаков; максимально осуществлялось удаление несущественных операторов, обобщавшее так называемую ликвидацию “мертвого кода”.

Таким образом, в системе БЕТА были развиты в сторону кристаллизации те понятия оптимизирующей трансляции, которые были найдены в Альфа-проектах: внутренний язык стал действительно единым, потоковый анализ был выделен как отдельный фрагмент, да и само выполнение языково-независимых оптимизаций было выделено в отдельную компоненту транслирующей системы, которая могла привлекаться по желанию пользователей.

Технологической целью проекта была максимальная унификация трансляции с различных языков. Общая схема трансляции включала двухпроходную схему получения абстрактной программы на внутреннем языке (декомпозиции), обнимающую лексический, синтаксический и контекстный анализ (язык Ада внес сюда некоторые коррективы), многопроходную работу оптимизирующего процессора, включаемую лишь по специальному запросу, и двухпроходную схему генерации кода. Оптимизирующий процессор не зависит ни от входного, ни от выходного языка - языково-зависимые конструкции внутреннего языка снабжались нужными для оптимизации атрибутами и были в этом смысле частично интерпретируемы. Вначале предполагалось, что как получение абстрактной программы, так и генерация кода будут строиться специальными метапроцессорами по описанию входного и выходного языков соответственно, однако в окончательной версии победил модульный подход. Была сделана библиотека модулей декомпозиции, фактически вводящая обобщенные операции, в терминах которых строились лексический,

синтаксический и контекстный анализ для конкретного языка. Точно также была создана библиотека генерации кода, включавшая модули общих для генерации действий - обхода программы на внутреннем языке, назначения регистров (с учетом конкретных параметров архитектуры) и пр. Как показал опыт, общая часть декомпозиции и генерации кода (оптимизация была универсальной) составляла в зависимости от языка от 40% (Ада) до 60% (Паскаль).

В целом этот проект был интересным экспериментом в трансляции программ, по своему размаху превосходящим другие опыты многоязыковой трансляции, существовавшие в мире. Были реализованы Симула 67, Паскаль, Ада и Модула-2, причем два последних языка, не участвовавшие в выработке схемы трансляции и внутреннего языка, достаточно хорошо вписались в систему, что свидетельствовало о надежности принятых в проекте решений. Выходными языками были столь разные языки БЭСМ-6 и СМ-4.

Область трансляции стала традиционной для новосибирцев. Было реализовано множество трансляторов для различных языков - от Сетла до Оберона, традиции многоязыковой трансляции были продолжены в биязыковой системе программирования XDS.

Наряду с исследованиями в различных направлениях системного программирования - и это очень важно отметить - в новосибирской школе велись исследования и по теоретическому программированию, основы которых были заложены А.П.Ершовым. Сознательная его позиция, заключавшаяся в том, что серьезные эксперименты по созданию систем должны подкрепляться разработкой теоретических моделей и стимулировать эту разработку, была весьма плодотворной.

Опыт в трансляторных проектах вызвал естественный интерес к теории схем программ - как последовательных, так и параллельных. Первыми работами новосибирской школы в этом направлении были работы А.П.Ершова по операторным алгоритмам, прообразом стандартных схем, и по аксиоматике схем Янова, когда предложенная Яновым аксиоматика преобразований была сведена к существенно меньшему числу аксиом. Работы по схемам программ концентрировались вокруг проблем эквивалентности и преобразований схем Янова. А.П.Ершовым и его учениками (В.Э.Иткиным, В.К.Сабельфельдом, Э.Л.Горель) был получен ряд интересных результатов[12]: был найден общий критерий локальности правил преобразований схем Янова, доказана логическая независимость правил преобразования, построена полная система преобразований с отношением тождества, доказана распознаваемость логико-термальной эквивалентности и пр.

Новосибирская школа одной из первых в мире занялась проблемами теории параллельного программирования. В.Е.Котовым и А.С.Нариньяни была предложена одна из первых моделей параллельных программ - недетерминированные асинхронные автоматы, и для этой модели получен ряд важных результатов, связанных с условиями отсутствия конфликтов параллелизма, с преобразованием последовательных программ в параллельные и преобразованиями, увеличивающими внутренний параллелизм - это определяющие проблемы теории параллелизма.

Для коллектива, активно работающего в области трансляции, естественным было обратить внимание на теорию оптимизации программ и на применение этой теории для создания оптимизирующих трансляторов. А.П.Ершовым (совместно с С.С.Лавровым) были заложены основы современной теории экономии памяти [13]. Результаты, полученные здесь, были использованы при построении оптимизирующей части Альфа-транслятора. Был предложен ряд моделей программ, ориентированных на обоснование других оптимизирующих преобразований [14]. Модель линейных схем позволяла обосновывать преобразования, связанные с удалением и перестановкой операторов, что было использовано в системе БЕТА. Модель линейных программ позволила построить оптимальный алгоритм преобразования линейных участков. Была предложена и более общая модель крупноблочных схем, объединяющая возможность линейных схем и преобразований, связанных с памятью.

Так закладывалось в новосибирской школе направление исследований по теоретическому программированию. Впоследствии эти работы вылились в исследования по специализации программ - А.П.Ершовым был предложен принцип смешанных вычислений, который дал толчок потоку исследований в разных коллективах в мире, была предложена идея конкретизации программ, - по дальнейшим исследованиям в теории оптимизации программ и в моделях параллельных программ, по спецификации и верификации параллельных и распределенных программ.

А.П.Ершов стимулировал возникновение и еще одного направления работ новосибирской школы - искусственного интеллекта. Г.И.Кожухиным еще в 1959 г. была построена, по-видимому, одна из первых программ по искусственному интеллекту - она на основании знаний путем проб и ошибок строила алгоритм нахождения корней полинома. Дальнейшие исследования по искусственному интеллекту возглавлялись А.С.Нариньяни, учеником А.П.Ершова. Вначале исследования концентрировались вокруг проблем понимания естественного языка и недетерминированного управления шагающим роботом. Проблемы, возникавшие здесь, привели к созданию современных направлений исследований - недоопределенности как основе программирования в ограничениях и методологии создания инструментов построения интеллектуальных систем.

Эта триада - системное программирование, теоретическое программирование, искусственный интеллект - с их взаимным влиянием и остается полем деятельности новосибирской школы. Основы всей этой деятельности возникли при могучем влиянии такого выдающегося отечественного ученого, как академик А.П.Ершов.

1. А.П.Ершов, Г.И.Кожухин, Ю.М.Волошин. Входной язык системы автоматического программирования: предварительное сообщение/М., ВЦ АН СССР, 1961, 176 с.
2. A.P.Ershov,G.I.Kozhuhin, Yu.M.Voloshin. Input language for automatic programming systems/Acad. Press, London-New York, 1963, 70 p.
3. А.П.Ершов. Основные принципы построения программирующей программы Института математики Сибирского отделения Академии наук СССР //Сиб.мат. журнал, т.2, № 6, 1961, - С.835-852.

4. АЛЬФА-система автоматизации программирования (под. ред. А.П.Ершова)/ Наука, Сиб.отд. - Новосибирск, 1967, -308 с.
5. The Alpha automatic programming System (Ed. by A.P.Ershov)/ Acad. Press, London-New York, 1971, 247 p.
6. И.Н.Аникеева, С.Ф.Богданова, А.О.Буда, Т.С.Васючкова и др. Система автоматизации программирования АЛЬФА-6//Системное программирование, часть 1, ВЦ СО АН СССР, 1973, -С.12-23.
7. A.P.Ershov, G.I.Kozhuhin, G.P.Makarov, M.I.Nechepurenko, I.V.Pottosin. An experimental automatic information station AIST-0//Proc. of AFIPS Conf., v.30, Academic Press, Washington-London, 1966, p.577-582.
8. Ю.Л.Вишневский, А.П.Ершов, Г.И.Кожухин, Г.П.Макаров и др. Экспериментальная система коллективного пользования АИСТ-0//Тр. 2-ой Всесоюзной конф. по программированию, заседание Н, ВЦ СО АН СССР, Новосибирск, 1970, -С.3-14.
9. И.В.Поттосин. Структура операционных систем коллективного пользования// Некоторые проблемы вычислительной и прикладной математики, Наука, Сиб.отд., Новосибирск, 1975.
10. А.П.Ершов.Универсальный программирующий процессор//Проблемы прикладной математики и механики. -М., Наука, 1971, -С.103-116.
11. А.П.Ершов, В.Н.Касьянов, С.Б.Покровский, И.В.Поттосин, Г.Г.Степанов. Методика разработки многоязыковых трансляторов на примере системы БЕТА//Математическая теория и практика систем программного обеспечения, ВЦ СО АН СССР, Новосибирск, 1982, -С.64-80.
12. А.П.Ершов. Современное состояние теории схем программ//Проблемы кибернетики, вып. 27, М., Наука, 1973, -С.87-110.
13. A.P.Ershov. Axiomatics for memory allocation//Acta Informatica, v.6, № 1, 1976, -P.61-75.
14. И.В.Поттосин. О математических моделях программ, ориентированных на оптимизацию программ// Proc. YII Nation. School “Mathematical Methods in Informatics”, Varna, 1981.