Лекция 2.3. Деятельностный подход к образованию.

Игры и тренажеры.
Натуральные макеты и визуальные модели.
Развитие психологических и поведенческих структур.
Учебные языки и системы программирования.
Проблема второго языка.
Учебные миры, игры и тренажеры.
Естественный язык и «первая грамотность».
Общение с носителями знаний.
Феномен настоящего.
Учебные языки и системы программирования (Робик, Рапира, Кубик и др.).

Появились многочисленные и весьма разнообразные игры и тренажеры, натуральные макеты и визуальные модели, используемых для образовательных применений, начиная с дошкольного возраста и включая образовательную поддержку людей без возрастных границ.

Развернуты массовые проекты, нацеленные на развитие психологических и поведенческих структур личности, включая разработку адаптационных компьютерных игр по проблеме дефицита внимания и интереса к жизни, а также проблем здоровья.

Появились десятки учебных языков и систем программирования, ценнейшие из которых созданы в Новосибирске. Идей таких работ заслуживают продолжения на новой технологической основе.

Проблема второго языка смягчается под давлением средств автоматического перевода и обучения иностранным языкам. Такие средства – это не только автоматизация дистанционного обучения. Это и массово распространяющиеся средства общения через Интернет, и разнообразные мобильные устройства, и оперативный доступ и электронным библиотекам, музеям, фильмам и т.п.

Естественный язык становится практичным как форма представления знаний. Контакты с носителями знаний выходят за пределы текстового общения. Объемы аудио и видео-записей теперь не смущают инженеров и программистов.

Учебные миры, игры и тренажеры стали новой отраслью информационной индустрии, что позволяет рассчитывать на создание «идеальных» обстановок для обучения новым, сложным, экзотическим и неудобным для системы общего образования предметам. В качестве примера можно рассмотреть проблему обучения параллельному программированию через создание языка начального обучения параллельному программированию и организацию системы дистанционного обучения.

Мир программирования и его техническая основа претерпели значительные изменения за последние двадцать лет. Возрастает актуальность обучения параллельному программированию, что требует развития языково-информационной поддержки введения в программирование.

В середине 1970-х годов активное исследование методов параллельного программирования рассматривалось как ведущее направление преодоления кризиса технологии программирования. Рост интереса к параллельному программированию связан с переходом к массовому производству многоядерных архитектур. Роль параллелизма в

современном программировании подробна рассмотрена в работах Вл.В. Воеводина. Их можно найти на сайте www.parallel.ru

Проблемы определения языков параллельного программирования связаны с дистанцией между уровнем абстрактных понятий, в которых описываются решения сложных задач, и уровнем конкретных аппаратных средств и архитектурных принципов управления параллельными вычислениями (потактовая синхронизация, совмещение действий во VLIW-архитектурах, сигналы, семафоры критических участков, рандеву и т.п.)

С параллельными процессами студенты встречаются при работе на уровне ОС, при организации практики на базе суперкомпьютеров, при сетевой обработке данных, при компиляции учебных программ и т.п. Проблемы подготовки параллельных программ для всех столь разных работ обладают общностью, но есть и существенная специфика, требующая понимания разницы в критериях оценки программ и информационных систем для различных применений.

Сложившиеся в Новосибирске традиции обучения программированию концентрируются вокруг мировоззренческой ценности информатики, конструктивной роли программирования в процессах развития и применения информационных технологий, системного подхода к информатике обучения. Разработанные в 1970-е годы учебные языки Робик и Рапира работали как предварительные ступени или лестница для перехода к производственным языкам и системам программирования. При вполне убедительных успехах Новосибирской системы обучения программированию учебный потенциал ее раскрыт не полностью. В настоящее время возрастает актуальность обучения параллельному программированию, что требует доработки понятийной базы и языково-информационной поддержки системы обучения.

Благодаря свободно-распространяемому обеспечению и Интернет можно получить опыт организации процессов для разных ОС. Управление процессами на уровне ОС заключается в оперировании заданиями, сводимыми к передаче данных между устройствами и файлами. Параллельное программирование на уровне ОС выглядят как нечто среднее между программированием на макроассемблере и языках высокого уровня. Различие проявляется в понимании данных и команд:

- Роль данных выполняют файлы.
- Файлы могут участвовать одновременно в разных процессах.
- Выполнение команды рассматривается как событие.
- Событие может быть успешным или неудачным.
- Реакция на событие программируется как обработчик события.
- Программа процесса может быть нацелена на непрерывное обслуживание заданий.
- Возможна синхронизация процессов и порождение подчиненных процессов.
- Программа процесса создается как данное, которое может применяться равноправно с командами.

В результате разработка программ для организации взаимодействия процессов отличается от подготовки обычных последовательных программ на весьма глубоком уровне, что можно показать на модели интерпретирующего автомата для языка управления процессами. Обычно используется две модели - супервизор, контролирующий взаимодействие семейства процессов, или автомат, способный тиражировать себя при ветвлении процессов. Функционирование автомата сводится к бесконечному циклу анализа происходящих событий, появление которых влечет включение обработчиков, соответствующих событиям. Проблема остановки решается вне языка – на уровне базовых

средств или внешним образом через прерывания. Доступ к общим ресурсам регулируется с помощью очередей запросов на обслуживание на базе моно- и/или мульти-процессорных комплексов. Обслуживание носит асинхронный характер, формулируемый в терминах событий, что сближает технику программирования процессов с ООП. Основной критерий - возможность продолжить выполнение заданий без принципиальных потерь информации.

По мнению Т. Хоара "Параллельная композиция действий внешне не сложнее последовательного сочетания строк в языке программирования". Функциональное программирование на Лиспе и других языках благодаря необычности (мало смягчившейся за 40 с лишним лет) и более гибкой модели организации вычислений позволяет предоставить простые примеры для показа непривычных идей параллелизма, интуитивное понимание которых не требует напряжения. В основе – выделение списков, порядок вычисления элементов которых может быть произвольным.

Обычно для решения конкретной задачи в терминах событий вводится специальный объект - автомат, способный реагировать на заданное множество событий в зависимости от состояния автомата. Поведение объекта, который способен действовать, описывается в виде системы правил, внешне похожих на уравнения. Решением такой системы уравнений являются процессы, которые может выполнить автомат. Формулировка такого рода закономерностей позволяет разделить описание процесса на уровень спецификации и реализации. Сравнивая запись вариантов процесса с представлением грамматики языка, можно обратить внимание на их подобие. Множества текстов или языки используются как характеристика, показывающая разнообразие вариантов хода событий - потенциал модели процессов.

Реализация процесса - функция, определяющая ход процесса по начальному событию. Таким событием может быть в частности готовность данных. Функциональная модель представления процессов позволяет легко описывать взаимодействие процессов в виде функционалов, т.е. функций над процессами, представленными как функциональные переменные. При определении взаимодействий используется понятие "протокол". Протокол - это последовательность символов, обозначающих произошедшие события. Важное направление анализа протоколов - проверка соответствия объектов спецификации процесса.

Функциональный подход к параллельному программированию ценен возможностью унификации понятий, различие которых мало существенно с точки зрения их реализации. При необходимости можно одинаково работать с процессами, объектами, системами и их контекстом. Все это определенные структуры данных с заданной схемой взаимодействия и разными предписанными ролями в более общем процессе информационной обработки. При необходимости взаимодействие процессов может быть пошагово синхронизовано. Взаимодействия процессов естественно могут быть заданы как взаимосвязанные рекурсивные функции.

Языки параллельного программирования

Существование проблемы второго языка объясняет, что достаточно распространены методы представления параллельных программ, через добавление средств параллелизма в популярные языки программирования, такие как Fortran, Pascal, Ada и др. Существуют версии ряда стандартных языков императивного программирования, приспособленные к выражению взаимодействия последовательных процессов в предположении, что в каждый момент времени существует лишь один процесс. При таком подходе в программе выделяются критические интервалы, учет которых полезен при распараллеливании

программ. Многие традиционные языки программирования приспособлены к выражению параллелизма с помощью специальных расширений или библиотечных функций, обеспечивающих выделение участков с независимыми действиями, пригодными для распараллеливания компилятором.

Применение параллельных архитектур повышает производительность при решении задач, явно сводимых к обработке векторов. Но автоматическое распараллеливание последовательных программ ограничивает ускорение вычислений. Более успешным может быть выражение языковыми средствами параллелизма на уровне постановки задачи. В таком случае при оптимизирующей компиляции возможен аккуратный выбор эффективной схемы параллелизма.

Независимая разработка специализированных языков параллельного программирования и языков управления процессами дала ряд интересных идей по представлению и масштабированию параллельных вычислений, с которыми можно ознакомиться в материалах о языках APL, Sisal, БАРС, Occam, Поляр и др. Характерно внимание комбинаторике компонентов, адаптированных к полному пространству независимо программируемых решений.

Исторически первое предложение по организации языка высокого уровня для параллельных вычислений было сделано в 1962 году Айверсоном в виде языка APL. Был предложен интересный механизм реализации многомерных векторов, приспособленный к расширению и распараллеливанию обработки. Сложные данные представляются как пара из последовательности скаляров и паспорта, согласно которому эта последовательность структурируется. В результате любое определение функции над скалярами автоматически распространяется на произвольные структуры данных из однотипных скаляров.

И в настоящее время для большинства специализированных языков параллельного программирования типично, что сложные построения факторизуются с учетом особенностей структуры данных так, что выделяются несложные отображающие функции, "просачиваемые" по структуре данных с помощью функций более высокого порядка - функционалов. В результате можно независимо варьировать структуры данных, функционалы, отображающие функции, методы сборки полного результата отображения и набор отображаемых множеств. Целенаправленно выделяются конвейерные процессы, приспособленные к минимизации хранения промежуточных результатов.

Языки параллельного программирования занимают видное место среди функциональных языков. Довольно известен язык функционального программирования Sisal. Название языка расшифровывется как “Streams and Iterations in a Single Assignment Language”. Sisal представляет собой результат развития языка VAL, известного в середине 70-х годов. Среди целей разработки языка Sisal следует отметить наиболее характерные, связанные с функциональным стилем программирования.

- Создание универсального функционального языка.
- Разработка техники оптимизации для высокоэффективных параллельных программ.
- Достижение эффективности исполнения, сравнимой с императивными языками типа Fortran и C.
- Внедрение функционального стиля программирования для больших научных программ.

Эти цели создателей языка Sisal подтверждают, что функциональные языки способствуют разработке корректных параллельных программ. Одна из причин заключается в том, что

функциональные программы свободны от побочных эффектов и ошибок, зависящих от реального времени. Это существенно упрощает отладку. Результаты переносимы на разные архитектуры, операционные системы или инструментальные окружения. В отличие от императивных языков, функциональные языки уменьшают нагрузку на кодирование, в них проще анализировать информационные потоки и схемы управления.

Легко создать функциональную программу, которая является безусловно параллельной, если ее можно писать, освободившись от большинства сложностей, связанных с выражением частичных отношений порядка между отдельными операциями уровня аппаратуры. Пользователь Sisal-а получает возможность сконцентрироваться на конструировании алгоритмов и разработке программ в терминах крупноблочных и регулярно организованных построений, опираясь на естественный параллелизм уровня постановки задачи.

Как и свойственно языкам функционального программирования, язык Sisal математически правилен - функции отображают аргументы в результаты без побочных эффектов, и программа строится как выражение, вырабатывающее значение. Форма параллельного цикла включает в себя три части: генератор пространства итераций, тело цикла и формирователь результата. Такая форма, дополненная требованием однократности присваиваний в любом блоке, хорошо приспособлена к распараллеливанию при компиляции.

Sisal-программа представляет собой набор функций, допускающих частичное применение, т.е. вычисление при неполном наборе аргументов. В таком случае по исходному определению функции строятся его проекции, зависящие от остальных аргументов, что позволяет оперативно использовать эффекты смешанных вычислений и определять специальные оптимизации программ, связанные с разнообразием используемых конструкций и реализационных вариантов параллельных вычислений.

Основные идеи языков параллельного программирования APL и VAL, предшественника языка Sisal, были обогащены в языке БАРС в трех направлениях:

1)в качестве базовой структуры данных были выбраны комплексы, представляющие собой нечто вроде размеченных множеств с возможностью обозначить кратность вхождения элементов,
2)описание элементов памяти сопровождается предписанием дисциплины доступа к памяти,
3)средства управления асинхронными процессами включали механизм синхросетей, позволяющий согласовывать функционирование узлов из независимо представленных фрагментов.

Процедуры в таком языке приспособлены к варьированию дисциплины доступа к данным и схемы управления процессами обработки комплексов. Синхросети позволяют независимые описания процессов связывать в терминах разметки. Узлы с одинаковой разметкой срабатывают одновременно. Полное представление об асинхронных процессах, их эффективности и проблемах организации дают работы по сетям Петри.

Перспективно применение универсальных языков сверх высокого уровня, таких как Setl, Python, Cw и т.п., абстрагирование данных и процессов в которых приспособлено к гибкому и строгому структурированию, удобному для доказательных построений. В этом плане представляет интерес эксперимент по развитию теоретико-множественной семантики языка Setl, в котором весьма общее построение формул с кванторами над

множествами погружено в обычную схему последовательного управления процессами. Реализация языка Setl характеризуется богатым полиморфизмом. Для представления множеств используется около двадцати разных структур данных, выбор которых осуществляется системой программирования в зависимости от динамики операций над множествами. В результате программируемые функции не могут зависеть от реализационной структуры данных.

В практике управления процессами используется понимание команд как позиций независимого порождения процессов. Такое понимание естественно согласуется с идеями теории множеств о независимости элементов множеств. Не исключено, что принятая в языке Sisal схема управления процессами может успешно заменить фортрановский стиль, принятый в языке Setl.

Учебные языки программирования

Прежде чем описать проект учебного языка параллельного программирования, рассмотрим наиболее известные учебные языки программирования с целью проявления их специфики и типовых свойств.

Возникший как удобное диалоговое средство быстрого программирования небольших расчетов язык Basic, оказался столь удобным для массового продвижения непрофессионалов в программирование, что в середине 1980-ых годов оказался серьезным кандидатом как учебный язык и рассматривался как медиум для интеграции информационных ресурсов.

Разработанный специально для университетской системы обучения программированию язык Pascal в конце 1970-х неожиданно перерос в язык системного программирования. На его основе созданы образцы не только высокоэффективной компиляции программ, но и прецедент непревзойденного по качеству стиля самодокументирования систем программирования.

Универсальный язык детского программирования и самообучения Logo, стал базисом для международных проектов ЮНЕСКО по развитию и поощрению творческой информатики в разных странах в форме конструирования Logo-миров и программистских конкурсов.

Ряд интересных идей предложено в калифорнийском проекте Grow, вводящем школьников в мир программирования через разработку компьютерных игр. Так, например, трудное для восприятия младшими школьниками понятие условного оператора заменено понятием вероятностного действия.

Наиболее методически обусловлено введение программистских понятий в языке начального обучения программированию Робик. Тщательная проработка понятий в этом языке может служить базисом для любой профориентации учебного процесса по информатике, программированию и информационным технологиям. Ведущее понятие - «исполнитель», причем исполнителей может быть много и они могут обладать разными системами команд. Такое понимание допускает естественное развитие в направлении моделирования асинхронных процессов. При записи программ используются варианты лингвистических форм, по разному обеспечивающие удобочитаемость в зависимости от уровня обучения, что позволяет язык оперативно приспосабливать к исходному лексикону аудитории с тем, чтобы в процессе обучения нормализовать терминологию. Учебная демонстрация взаимодействующих процессов может быть показана на примере

исполнителей “Муравей” и “Машинист” языка начального обучения программированию Робик.

Язык учебно-производственного программирования Рапира был реализован с эргономичным интерфейсом работы с процедурами. В качестве основной структуры данных используются многоуровневые кортежи произвольной размерности. Нет чрезмерного напряжения вокруг проблемы типизации переменных. Языковые конструкции нацелены на гладкий переход к программированию на основных языках производственного программирования, характерных для конца 1970-ых годов.

Во многих странах в качестве учебных ЯП успешно используются Lisp, Scheme, Prolog, в которых четко выделены и красиво описаны учебные подъязыки, приспособленные к полноценному программированию интересных задач на базе ограниченного свода средств, что делает учебную практику увлекательным занятием. Языки REX и Forth часто относят к учебным.

С начала 1980-х появилось много интересных предложений в области учебных языков программирования, по уровню весьма близких языку Рапира, - A++, ECL, Elan, Karel.

Обычно учебные языки программирования обеспечивают быстрое получение первых успехов. Языки XXI века как правило поддерживают все четыре основные парадигмы программирования на ЯВУ – императивное, функциональное, логическое и объектно-ориентированное, что позволяет учащимся программировать решения задач с разным уровнем изученности в рамках единой языковой обстановки и получать навыки работы на всех фазах полного жизненного цикла программ.

Для многих языков высокого уровня (ЯВУ) характерно включение в семантику языка одной модели параллелизма. Так в Algol-68 включено управление процессами в терминах семафоров для критических участков. В языке Ada предложен механизм «рандеву». APL нацеливает на обобщение скалярных операций до обработки однородных векторов произвольной размерности. SISAL предоставляет ряд средств формирования пространства итераций для параллельного исполнения циклов. Производственные инструменты MPI и Open MP также связаны определенными моделями параллелизма, поддержанными на уровне аппаратуры, что чревато известной проблемой переноса программ.

Перспективно применение универсальных языков сверх высокого уровня, таких как Setl, Python, Cw и т.п., абстрагирование данных и процессов в которых приспособлено к гибкому и строгому структурированию, удобному для доказательных построений. В этом плане представляет интерес эксперимент по развитию теоретико-множественной семантики языка Setl, в котором весьма общее построение формул с кванторами над множествами погружено в обычную схему последовательного управления процессами. Реализация языка Setl характеризуется богатым полиморфизмом.

Многие затруднения с параллельным программированием имеют отчасти образовательный характер. Концепции взаимодействующих процессов следовало бы осваивать при начальном знакомстве с информационными технологиями и программированием.

Обучение программированию

При проектировании языка начального обучения параллельному программированию (ЯНОП) учтены факторы успеха наиболее известных учебных языков программирования,

таких как Basic, Pascal, Logo, Grow, с целью проявления их специфики и типовых свойств. Методически обусловлено введение программистских понятий в языке начального обучения программированию Робик. Тщательная проработка понятий в этом языке может служить базисом для любой профориентации учебного процесса по информатике, программированию и информационным технологиям. Ведущее понятие - «исполнитель», причем исполнителей может быть много, и они могут обладать разными системами команд. Такое понимание допускает естественное развитие в направлении моделирования асинхронных процессов.

Обычно учебные языки программирования обеспечивают быстрое получение первых успехов и обеспечивают переход от абстрактной алгоритмизации к практике отладки программ на производственных системах. Языки XXI века как правило поддерживают основные парадигмы программирования на ЯВУ – императивное, функциональное, логическое и объектно-ориентированное, что позволяет программировать решения задач с разным уровнем изученности в рамках единой языковой обстановки и получать навыки работы на всех фазах полного жизненного цикла программ.

Для языка обучения параллельному программированию следует предъявить следующие требования:

- Самоприменимость
- Полиморфизм и взаимосводимость типов данных
- Многовариантность лингвистической базы
- Визуализируемость объектов и процессов
- Концентричность или разложение по уровням обучения
- Настраиваемость на понятия производственных языков и систем

Самоприменимость позволяет описывать в терминах изучаемого языка его реализацию. Для языка параллельного программирования это дает возможность изучать реальные модели параллелизма уровня аппаратуры и зависимость эффективности параллельных вычислений от программных решений. Тем самым появляется возможность формировать правильную интуицию при выборе структур данных и схем управления процессами.

Полиморфизм и взаимосводимость типов данных нужна при формировании пошаговой схемы обучения, свободной от необходимости изучать все сразу. При организации сложных данных и процессов используются общие структуры или средства композиции, такие как списки, множества и варианты, обеспечивающие представление отношений «после», «одновременно» и «взаимоисключение». Свойства таких структур и средств их обработки можно изучать в академическом стиле, рассматривая отдельно реализационные особенности.

Многовариантность лингвистической базы - это требование к учебному языку программирования – обеспечить возможность отражения в его синтаксисе прогресса в обучении, овладении терминологией и приобретении навыков программирования в разных парадигмах. Для этого ЯНОП описывается многовариантной грамматикой, представляющей эквиваленты изучаемых конструкций. Похожий подход реализован в языках Cobol и Робик.

Визуализируемость объектов и процессов – важное требование к учебному языку программирования, связанное с параллелизмом. Нужна понятная визуализация взаимодействия процессов, включая демонстрацию зависимости от времени доступа к разным уровням памяти. Система визуализации для ЯНОП представляет собой иерархию

классов объектов, изображаемых и перемещаемых на раскрашенной клетчатой доске, символизирующей обстановку для решения модельных задач.

Концентричность обеспечивает разложение по уровням обучения. Реализация учебного языка параллельного программирования образует три уровня – элементарно-учебное ядро, технолого-прагматическое расширение и вводно-ознакомительная оболочка.

Задача вводно-ознакомительной оболочки – дать средства тексто-графической визуализации и моделирования примеров из жизни, а также создания простых игр, на которых можно показать и пронаблюдать модели аппаратных решений и связанные с их применением явления на уровне автоматов, описанных в книге Хоара. Решение строится на основе идей и конструкций учебных языков Grow и Робик с учетом современных технологий разработки новых информационных систем.

Постановка учебной задачи формулируется в терминах команд исполнителю. Решение может быть получено как оперированием, так и программированием. При отладке решений можно задавать условное время показа отдельных действий. Кроме обычных команд по изменению обстановки, объектов, статуса исполнителя и меню допустимых команд, в ЯНОП имеются специальные команды для смены исполнителей и обстановок, а также для разрешения взаимодействий между исполнителями.

Учебные программы в ЯНОП строятся из действий, выполнение которых может быть обусловлено ожиданием времени (пауза) или другого сигнала, предикатом или вероятностью срабатывания. Последнее означает, что при исполнении ведется учет частоты выполнения вероятностных действий как в системах для разработки компьютерных игр. Можно объявить планируемую длительность выполнения действий.

Настраиваемость на понятия производственных языков и систем. Представление циклов на ЯНОП учитывает решения языка Sisal, а именно, выделяются участки для формирования пространства распараллеливаемых итераций и для свертки полученных параллельно значений в общий результат.

Ядро реализации ЯНОП составляют базовые средства уровня ОС – условия готовности и завершения процесса, очереди и серийная обработка, типичные для ОС UNIX и языка Java. Такие средства рассматриваются как основа для реализации схем управления, встречающихся в ЯВУ (критические участки и семафоры, рандеву, защищенные команды, конвейеры), навыки программирования и применения которых входят в задачу элементарно-учебного уровня.

Мультизначения, приоритеты процессов, просачивание скалярных операций на структуры данных, выстраивание пространства итераций, фильтры данных, сетевая синхронизация действий, оптимизирующие преобразования программ и распараллеливающие трансформации процессов – все это формируется как расширения ядра языка, выполняемые с целью достижения понимания реализационной семантики параллельного программирования для современных средств, что и является задачей технолого-прагматического уровня.

Парадигмы параллельного программирования можно характеризовать сочетанием низкоуровневых средств управления процессами обработки событий (семафоры, рандеву) и высокоуровневых методов представления иерархии данных (вектора, структуры и размеченные объединения) с просачиваемыми по такой иерархии действиями в стиле объектно-ориентированного программирования, дополненным механизмами

сверхвысокого уровня, допускающими или обеспечивающими формирование сложных комплексов из независимо программируемых компонент.

В последние годы созрело понимание целесообразности раннего обучения параллельному программированию. На новосибирских Летних школах юных программистов традиционно проводятся лекции, знакомящие школьников с проблемами параллелизма. При проведении экспериментов по моделированию параллельных процессов учащимися 6-х классов не было замечено затруднений в понимании задачи. Концептуальный подход к обучению программированию позволяет быстро вникнуть в существо основных понятий. Акцент на приобретение практических навыков работы на весьма различных системах обеспечивал профилактику привыкания к первому языку, системе, машине, создавал предпосылки успешного самообучения и освоения новых средств.

В работах ИСИ СО РАН им. А.П. Ершова по исследованию и классификации компьютерных языков, нацеленных на реализацию идей А.П. Ершова и С.С. Лаврова о создании лексикона программирования, заметное место занимает исследование парадигм программирования и особо – парадигм параллельного программирования, обладающих большим разнообразием вычислительных моделей и подходов к аппаратной поддержке высокопроизводительных вычислений. Для систематичной подготовки специалистов в области параллельного программирования необходима инструментально-языковая поддержка учебной практики, опережающей навыки последовательного программирования, преодоление которых чревато значительными трудозатратами. Пусть параллельное программирование станет столь же легким и приятным как обычное, системное, функциональное, логическое, объектно-ориентированное и т.д. и т.п.