МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ В.Ф. УТКИНА»

КАФЕДРА

«ВЫЧИСЛИТЕЛЬНАЯ И ПРИКЛАДНАЯ МАТЕМАТИКА»

МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ Б1.В.ДВ.06 «РАЗРАБОТКА СИСТЕМНЫХ УТИЛИТ»

Направление подготовки – 09.03.04 «Программная инженерия»

Направленность (профиль) подготовки
«Программная инженерия»

ОПОП академического бакалавриата

Квалификация выпускника – бакалавр

Формы обучения – очная

Рязань 2022 г

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Рекомендуется следующим образом организовать время, необходимое для изучения дисциплины:

Для освоения лекционного материала следует: изучить конспект лекции в тот же день, после лекции: 10 – 15 минут, повторно прочитать конспект лекции за день перед следующей лекцией: 10 – 15 минут. Также следует изучить теоретический лекционный материал по рекомендуемому учебнику/ учебному пособию: 1 час в неделю.

Следует максимально использовать лекционное время для изучения дисциплины, понимания лекционного материала и написания конспекта лекций. В процессе лекционного занятия студент должен уметь выделять важные моменты и основные положения. При написании конспекта лекций следует придерживаться следующих правил и рекомендаций.

1. При ведении конспекта рекомендуется структурировать материал по разделам, главам, темам. Вести нумерацию формул. Выделять по каждой теме постановку задачи, основные положения, выводы. Кратко записывать те пояснения лектора, которые показались особенно важными. Это позволит при подготовке к сдаче зачёта и экзамена не запутаться в структуре лекционного материала.

2. Лекционный материал следует записывать в конспект лишь после того, как излагаемый лектором тезис будет вами дослушан до конца и понят.

3. При конспектировании следует отмечать непонятные, на данном этапе, положения, доказательства и пр.

4. Рекомендуется по каждой теме выразить свое мнение, комментарий, вывод.

Подготовка к практическим занятиям:

Практические занятия по дисциплине существенно дополняют лекции. В процессе анализа теоретических положений и решения практических задач студенты расширяют и углубляют свои знания, полученные из лекционного курса и учебников, приобретают умение применять общие закономерности к конкретным случаям. В процессе решения задач развивается логическое мышление и вырабатываются навыки вычислений, работы со справочной литературой. Практические занятия способствуют закреплению знаний и практических навыков, формированию конструктивного стиля мышления, расширению кругозора.

При подготовке к практическому занятию необходимо внимательно ознакомиться с соответствующим теоретическим материалом по конспекту

лекций и рекомендуемому учебнику, затем изучить конспект или материалы предыдущего практического занятия и выполнить заданное расчетное задание: 1 – 2 часа в неделю.

Следует максимально использовать аудиторное время практических занятий. В процессе занятия студент должен активно участвовать в дискуссиях, обсуждениях и решениях практических задач и вести конспект практических занятий отдельно от конспекта лекций.

Дополнительно в часы самостоятельной работы студенты могут повторно решить задачи, с которыми они плохо освоились во время аудиторных занятий, и обязательно те задачи, которые не получились дома при предыдущей подготовке к практическим занятиям.

Подготовка к лабораторным работам.

Перед началом проведения лабораторной работы необходимо ознакомится с методическими указаниями к данной лабораторной работе, внимательно ознакомиться с заданием и желательно заранее выполнить подготовку проекта в используемой инструментальной среде, чтобы время лабораторного занятия использовать для исправления ошибок, модификации проекта и защиты данной работы.

Выполнение каждой из запланированных работ заканчивается предоставлением отчета. Требования к форме и содержанию отчета приведены в методических указаниях к лабораторным работам или определяются преподавателем на первом занятии. Отчет по лабораторной работе студент должен начать оформлять еще на этапе подготовки к ее выполнению. Допускаясь к лабораторной работе, каждый студент должен представить преподавателю «заготовку» отчета, содержащую: оформленный титульный лист или название и номер работы при ведении общего конспекта, цель работы, задание, проект решения, полученные результаты, выводы.

Изучение методических указаний к лабораторной работе – 2 часа перед выполнением лабораторной работы и в ходе разработки проекта и 2 часа для оформления отчета, отладки проекта и подготовки к сдаче работы.

После выполнения лабораторной работы необходимо согласовать полученные результаты с преподавателем. Важным этапом является защита лабораторной работы. В процессе защиты студент отвечает на вопросы преподавателя, касающиеся теоретического материала, относящегося к данной работе, и проекта, реализующего его задание, комментирует полученные в ходе работы результаты. При подготовке к защите лабораторной работы рекомендуется ознакомиться со списком вопросов по изучаемой теме и попытаться самостоятельно на них ответить, используя конспект лекций и рекомендуемую

литературу. Кроме чтения учебной литературы рекомендуется активно использовать информационные ресурсы сети Интернет по изучаемой теме.

Подготовка к сдаче экзамена или зачета:

Экзамен/зачет – форма промежуточной проверки знаний, умений, навыков, степени освоения дисциплины. Главная задача экзамена/зачета состоит в том, чтобы у студента по окончанию изучения данной дисциплины сформировались определенное представление об общем содержании дисциплины, определенные теоретические знания и практические навыки, определенный кругозор. Готовясь к экзамену/зачету, студент приводит в систему знания, полученные на лекциях, на практических и лабораторных занятиях, разбирается в том, что осталось непонятным, и тогда изучаемая им дисциплина может быть воспринята в полном объеме с присущей ей строгостью и логичностью, ее практической направленностью.

Экзамены/зачеты дают возможность преподавателю определить теоретические знания студента и его практические навыки при решении определенных прикладных задач. Оцениваются: понимание и степень усвоения теоретического материала; степень знакомства с основной и дополнительно литературой, а также с современными публикациями; умение применить теорию к практике, решать определенные практические задачи данной предметной области, правильно проводить расчеты и т. д.; знакомство с историей данной науки; логика, структура и стиль ответа, умение защищать выдвигаемые положения.

Значение экзаменов/зачетов не ограничивается проверкой знаний, являясь естественным завершением обучения студента по данной дисциплине, они способствуют обобщению и закреплению знаний и умений, приведению их в стройную систему, а также устранению возникших в процессе обучения пробелов.

Подготовка к экзамену/зачету – это тщательное изучение и систематизация учебного материала, осмысление и запоминание теоретических положений, формулировок, формул, установление и осмысление внутрипредметных связей между различными темами и разделами дисциплины, закрепление теоретических знаний путем решения определенных задач.

Перед экзаменом назначается консультация, ее цель – дать ответы на вопросы, возникшие в ходе самостоятельной подготовки студента, студент имеет возможность получить ответ на все неясные ему вопросы, кроме того, преподаватель будет отвечать на вопросы других студентов, что будет способствовать повторению и закреплению знаний всех присутствующих. Преподаватель на консультации, как правило, обращает внимание на те разделы, по которым на предыдущих экзаменах ответы были

неудовлетворительными, а также фиксирует внимание на наиболее трудных разделах курса.

На непосредственную подготовку к экзамену обычно дается 3 – 5 дней. Этого времени достаточно для углубления, расширения и систематизации знаний, полученных в ходе обучения, на устранение пробелов в знании отдельных вопросов, для определения объема ответов на каждый из вопросов рабочей программы дисциплины.

Планируйте подготовку к зачету/экзамену, учитывая сразу несколько факторов: неоднородность в сложности учебного материала и степени его проработки в ходе обучения, свои индивидуальные способности. Рекомендуется делать перерывы в занятиях через каждые 50-60 минут на 10 минут. После 3-4 часов занятий следует сделать часовой перерыв. Чрезмерное утомление приведет к снижению тонуса интеллектуальной деятельности. Целесообразно разделять весь рабочий день на три рабочих периода – с утра до обеда, с обеда до ужина и с ужина до сна. Каждый рабочий период дня должен заканчиваться отдыхом не менее 1 часа. Работая в сессионном режиме, студент имеет возможность увеличить время занятий с 10 (как требовалось в семестре) до 12 часов в сутки.

Подготовку к экзаменам или зачетам следует начинать с общего планирования своей деятельности. С определения объема материала, подлежащего проработке, необходимо внимательно сверить свои конспекты с программой дисциплины, чтобы убедиться, все ли разделы отражены в лекциях, отсутствующие темы изучить по учебнику. Второй этап предусматривает системное изучение материала по данному предмету с обязательной записью всех выкладок, выводов, формул. На третьем этапе – этапе закрепления – полезно чередовать углубленное повторение особенно сложных вопросов с беглым повторением всего материала.

Рекомендации по работе с литературой:

Теоретический материал курса становится более понятным, когда дополнительно к прослушиванию лекции и изучению конспекта изучаются и книги по данному предмету. Литературу по дисциплине рекомендуется читать как в бумажном, так и в электронном виде (если отсутствует бумажный аналог). Полезно использовать несколько учебников и пособий по дисциплине. Рекомендуется после изучения очередного параграфа ответить на несколько вопросов по данной теме. Кроме того, полезно мысленно задать себе следующие вопросы (и попробовать ответить на них): «о чем этот параграф?», «какие новые понятия введены, каков их смысл?», «зачем мне это нужно по специальности?».

Рекомендуется самостоятельно изучать материал, который еще не прочитан на лекции и не применялся на лабораторном или практическом занятии, тогда занятия будут гораздо понятнее. В течение недели рекомендуется выбрать время (1 час) для работы с литературой.

Лабораторная работа №1.
Создание системной утилиты, управляемой параметрами запуска.

Цель работы:

Создание консольной утилиты для генерации арифметических выражений и их трансляции в словесное представление.

Задание:

Необходимо сделать консольную утилиту, которая имеет два режима работы: генерация и трансляция арифметических выражений.
1) Режим генерации. Утилита принимает в качестве параметра аргумент «G», имя файла, количество строк, минимальное количество операндов, максимальное количество операндов. Количество операндов в каждом выражении задаётся случайным образом, исходя из двух последних аргументов вызова (т.е. в нашем примере это от 4 до 8 операндов).
2) Режим трансляции. Утилита принимает в качестве параметра аргумент «T», имя транслируемого файла, имя выходного файла. После этого утилита считывает входной файл output.txt и транслирует его в output_trans.txt заменяя цифры и операции на их словесное представление.
Утилита должна правильно обрабатывать ошибочный ввод, например, передача неправильного количества параметров, неправильные значение параметров, передача для трансляции несуществующего файла и т.д. и выдавать описание ошибки в случае её возникновения.

Лабораторная работа №2.
Создание системной утилиты, для шифрования файлов.

Цель работы:

Создание консольной утилиты для шифрования файлов с помощью алгоритма SPEED.

Задание:

Необходимо создать консольную утилиту, которая на вход принимает бинарный файл и код шифрования, а на выходе выдаёт шифрованный или дешифрованный файл (в зависимости от переданного аргумента).

Шифрование по алгоритму SPEED состоит из двух этапов. Процедура расширения ключа и непосредственно шифрование. Для дешифрования выполняется сначала процедура расширения ключа, а затем операции, обратные процедуре шифрования.

Так как алгоритм SPEED имеет переменные параметры, то для спецификации алгоритма с конкретными параметрами принято указывать (W,L,R), где:

- W — это размер блока шифруемых данных, который может принимать значения: 64, 128 и 256 бит соответственно.
- L — это размер ключа, который принимает значение L диапазоне от 48 до 256 бит, которое кратно 16.
- R — это количество раундов преобразований. Количество преобразований при этом должно быть не менее 32 и кратно 4.

Ключ шифрования/дешифрования K для SPEED представляет собой двоичную строку из L бит, где L является целым числом от 48 до 256 и делится на 16. Функция планирования ключей состоит в том, чтобы «расширять» ключ K на R фрагментов циклового ключа, необходимых для R раундов обработки.

SPEED, применяя ключ K длиной L бит, преобразует открытый текст M из W бит в зашифрованный C той же длины.

Криптографический ключ K, представляющий собой строку из L бит, сначала расширяется с помощью функции планирования ключей на четыре ключа K1, K2, K3 и K4. Каждый из этих ключей состоит из R цикловых ключей, где указано количество раундов в каждом проходе.

Открытый текст M представляется как 8 строк, W/8 бит каждый. Эти 8 строк обрабатываются в фазах P1, P2, P3 и P4 последовательно. Каждая фаза называется проходом и включает в себя ключи K1, K2, K3, K4 соответственно. На выходе мы получим зашифрованный текст C из открытого M. Все 4 фазы внутреннего прохода P1, P2, P3, P4 работают одинаково, хотя в каждом проходе используется отдельный подключ K1, K2, K3, K4, а также разные нелинейные функции для побитовых логических операций.

Лабораторная работа №3.
Создание системной утилиты, для архивации файлов.

Цель работы:

Создание консольной утилиты для архивации файлов с помощью алгоритма LZ77.

Задание:

Необходимо создать консольную утилиту, которая на вход принимает бинарный файл, а на выходе выдаёт архивированный или разархивированный файл (в зависимости от переданного аргумента).

Для понимания данного алгоритма LZ77 необходимо разобраться с двумя его составляющими: принципом скользящего окна и механизмом кодирования совпадений.

Метод кодирования, согласно принципу скользящего окна, учитывает уже ранее встречавшуюся информацию, то есть информацию, которая уже известна для кодировщика и декодировщика (второе и последующие вхождения некоторой строки символов в сообщении заменяются ссылками на её первое вхождение).

Благодаря этому принципу алгоритм иногда называется методами сжатия с использованием скользящего окна. Скользящее окно можно представить в виде буфера (или более сложной динамической структуры данных), который организован так, чтобы запоминать «сказанную» ранее информацию и предоставлять к ней доступ. Таким образом, сам процесс сжимающего кодирования согласно LZ77 напоминает написание программы, команды которой позволяют обращаться к элементам «скользящего окна», и вместо значений сжимаемой последовательности вставлять ссылки на эти значения в «скользящем окне». Размер скользящего окна может динамически изменяться и составлять 2, 4 или 32 килобайта. Следует также отметить, что размер окна кодировщика может быть меньше или равен размеру окна декодировщика, но не наоборот.

Перед тем, как перейти к рассмотрению механизма кодирования, уточним понятие совпадения (от англ. match). Рассмотрим последовательность из N элементов. Если все элементы последовательности уникальны, то такая последовательность не будет содержать ни одного повторяющегося элемента, или, иначе говоря, в последовательности не найдется хотя бы двух равных друг другу или совпадающих элементов.

В стандартном алгоритме LZ77 совпадения кодируются парой:

- длина совпадения (match length)
- смещение (offset) или дистанция (distance)

В продолжение уже приведенной аналогии с программированием отметим, что в большинстве статей, посвященных алгоритму LZ77, кодируемая пара трактуется именно как команда копирования символов из скользящего окна с определенной позиции, или дословно как: «Вернуться в буфере символов на значение смещения и скопировать значение длины символов, начиная с текущей позиции».

Хотя для приверженцев императивного программирования такая интерпретация может показаться интуитивно понятной, она мало говорит о сущности алгоритма LZ77 как метода сжатия. Особенность данного алгоритма сжатия заключается в том, что использование кодируемой пары длина-смещение является не только приемлемым, но и эффективным в тех случаях, когда значение длины превышает значение смещения.

Пример с командой копирования не совсем очевиден: «Вернуться на 1 символ назад в буфере и скопировать 7 символов, начиная с текущей позиции». Каким образом можно скопировать 7 символов из буфера, когда в настоящий момент в буфере находится только 1 символ? Однако следующая интерпретация кодирующей пары может прояснить ситуацию: каждые 7 последующих символов совпадают (эквивалентны) с 1 символом перед ними.

Это означает, что каждый символ можно однозначно определить, переместившись назад в буфере — даже если данный символ ещё отсутствует в буфере на момент декодирования текущей пары длина-смещение. Такая кодируемая пара будет представлять собой многократное (определяемое значением смещения) повторение последовательности (определяемой значением длины) символов.

Лабораторная работа №4.
Создание простейшего WDM драйвера.

Цель работы:

Создание простейшего WDM драйвера, который будет записывать в бинарный файл след управляющий команд пользователя.

Задание:

Необходимо создать простейший WDM драйвер, который будет записывать в бинарный файл след управляющий команд пользователя: все

коды нажатых клавиш клавиатуры, координаты движения мыши и нажатия ее клавиш.

Драйвер — это набор функций, которые вызываются операционной системой при наступлении некоторых событий, приходящих от устройства или пользовательского режима. Существует достаточно много типов драйверов, ниже перечисленны некоторые из них:

Драйверы классов — это драйверы, котрые пишет Microsoft. Это общие драйвера для определенного класса (неужели!) устройств.

Минидрайверы — драйверы, которые используют драйвер класса для управления устройством.

Функциональные драйверы — это драйверы, которые работают самостоятельно и определяет все что связано с устройством.

Фильтрующие драйверы — драйверы, которые используются для мониторинга или изменения логики другого драйвера путем изменения данных, которые идут к нему.

Необязательно определять все возможные функции в своем драйвере, но он обязательно должен содержать DriverEntry и AddDevice. IRP — это структура, которая используется драйверами для обмена данными.

Для того чтобы считывать управляющие команды с устройств пользователя, будет использоваться фильтрующий драйвер. Существует два типа фильтрующих драйверов:

- верхние фильтрующие драйверы;
- нижние фильтрующие драйверы.

То, к какому типу относится драйвер, зависит от того где этот драйвер находится в стеке драйверов устройств. Если драйвер находится выше функционального драйвера, то его называют верхним фильтрующим драйвером, если ниже, то, нижним фильтрующим драйвером.

Через верхние фильтрующие драйверы проходят все запросы, а это значит, что они могут изменять и/или фильтровать информацию, идущую к функциональному драйверу, ну и далее, возможно, к устройству.

Пример использования верхних фильтрующих драйверов: фильтр-хук драйвер, который устанавливает свою хук-функцию для системного драйвера IpFilterDirver, для отслеживания и фильтрации траффика. Такие драйверы используются в брандмауэрах.

Через нижние фильтрующие драйверы проходит меньше запросов потому что большинство запросов выполняет и завершает функциональный драйвер.

Для запуска драйвера будет использоваться утилита KmdManager. Для просмотра отладочной информации будет использоваться утилита DbgView.

Объект pLowerDO это объект устройства, который находится ниже в стеке. Он нужен для того чтобы знать кому дальше отправлять IRP-пакеты.

theDriverObject – объект драйвера, содержит указатели на все необходимые операционной системе функции, которые должны быть инициализированы.

ustrRegistryPath – имя раздела в реестре, где хранится информация о данном драйвере.

Функция DispatchRead будет обрабатывать запросы на чтение. Она будет вызываться, когда нажата или отпущена клавиша клавиатуры.

Функция DriverUnload вызывается, когда драйвер уже не нужен и его можно выгрузить из памяти, или когда пользователь сам выгружает драйвер. В данной функции должна производиться «зачистка», т.е. освобождаться ресурсы, которые использовались драйвером, завершаться все незавершенные запросы и т.д.
Функция DispatchThru это функция-заглушка. Все что она делает это передача IRP-пакета следующему драйверу (драйверу который находится под нашим в стеке, т.е. pLowerDO из DEVICE_EXTENSION).

Перед тем как передать запрос следующему драйверу необходимо настроить указатель стека для драйвера. IoCopyCurrentIrpStackLocationToNext копирует участок памяти, который принадлежит текущему драйверу, в область памяти следующего драйвера.

Практическое занятие по теме № 1

Вопросы:

1. Архитектура компьютера.
2. Архитектура фон Неймана.
3. Язык программирования. Языки низкого уровня. Языки высокого уровня.
4. Поколения языков программирования.
5. Парадигмы языков программирования. Императивные языки.
6. Парадигмы языков программирования. Функциональные языки.

7. Парадигмы языков программирования. Декларативные языки.
8. Парадигмы языков программирования. Объектно-ориентированные языки.
9. Трансляторы. Проблема трансляции языка программирования.
10. Виртуальная трехадресная машина. Переход к мнемокоду. Примеры.
11. Требования, предъявляемые к транслятору.
12. Компиляция и интерпретация. Примеры.
13. Компилятор. Целевая программа. Схема.
14. Интерпретатор. Схема.
15. Система обработки языка. Схема. Препроцессор. Компилятор.

Практическое занятие по теме № 2

Вопросы:

1. Операционные системы.
2. Драйверы устройств.
3. Ассемблер. Компоновщик. Загрузчик.
4. Ядро операционной системы.
5. Фазы компиляции. Схема.
6. Приведения (coercion). Пример.
7. Генерация промежуточного кода. Пример.
8. Оптимизация кода. Пример.
9. Генерация кода. Пример.
10. Таблица символов. Пример. Проходы компилятора.
11. Инфиксная и постфиксная форма выражения. Назначение. Примеры.
12. Модель начальной стадии компилятора. Схема. Лексический анализатор. Токены.
13. Фаза генерации промежуточного кода. Виды промежуточного кода.

Практическое занятие по теме № 3

Вопросы:

1. Абстрактное синтаксическое дерево. Пример.
2. Контекстно-свободная грамматика. Пример.
3. Определение грамматик. Компоненты грамматик. Терминалы и нетерминалы.
4. Лексемы. Токены. Терминалы.
5. Пример составления грамматики для выражений, состоящих из цифр и знаков плюс и минус.
6. Выведение (Порождение). Пример для грамматики для выражений, состоящих из цифр и знаков плюс и минус.

7. Пример составления грамматики для функций и их параметров.
8. Деревья разбора. Пример. Свойства.
9. Деревья разбора. Пример. Узлы. Метки. Корень. Родитель. Дочерний
10. Деревья разбора. Пример. Родственный узел. Лист. Внутренний узел. Потомок. Предок.
11. Пример построения дерева для выражения, состоящих из цифр и знаков плюс и минус.
12. Неоднозначности грамматики. Пример.
13. Ассоциативность операторов. Пример грамматики и дерева разбора.
14. Приоритет операторов. Пример построения грамматики.
15. Синтаксически управляемая трансляция. Пример.

Практическое занятие по теме № 4

Вопросы:

1. Синтаксически управляемая трансляция. Атрибуты.
2. Синтаксически управляемая трансляция. Схемы трансляции.
3. Постфиксная запись. Правило трансляции. Пример трансляции.
4. Постфиксная запись. Арность. Правило вычисления. Пример вычисления.
5. Синтезированные атрибуты. Синтаксически управляемое определение. Вычисление атрибутов.
6. Аннотированное дерево разбора. Пример.
7. Синтаксически управляемые определения для трансляции инфиксных выражений в постфиксные.
8. Обходы дерева. Обход в глубину. Схема.
9. Обходы дерева. Прямой и обратный порядок.
10. Схемы трансляции. Семантические действия. Пример продукции и дерева разбора с семантическим действием.
11. Семантические действия. Пример действий трансляции выражения в постфиксную запись, дерево разбора.
12. Разбор. Классы методов разбора.
13. Нисходящий анализ. Построение дерева разбора.
14. Предиктивный анализ. Использование пустых продукций.
15. Разработка предиктивного анализатора. Левая и правая рекурсия.

Библиографический список

1. Гинзбург С. Математическая теория контекстно-свободных языков. – М.: Мир, 1977.
2. Гладкий А. В. Формальные грамматики и языки. – М.: Наука, 1973.
3. Грис Д. Конструирование компиляторов для цифровых вычислительных машин. – М.: Мир, 1975.
4. Гросс М., Лантен А. Теория формальных грамматик. – М.: Мир, 1971.
5. Кнут Д. О переводе (трансляции) языков слева направо. / В сб. "Языки и автоматы". – М.: Мир.
6. Кнут Д. Семантика контекстно-свободных языков./ В сб. "Семантика языков программирования". – М.: Мир, 1980.
7. Лебедев В. Н. Введение в системы программирования. – М.: Статистика, 1975.
8. Льюис Ф., Розенкранц Д., Стирнз Р. Теоретические основы построения трансляторов. – М.: Мир, 1979.
9. Льюис Ф., Розенкранц Д., Стирнз Р. Атрибутные трансляции. / В сб. "Семантика языков программирования". – М.: Мир, 1980.
10. Оалева Э. А., Самойленко В. П., Семенова О. Н. Формальные методы описания перевода. Учеб. пособие. – СПб.: СПбГЭТУ, 2000.
11. Оалева Э. А., Самойленко В. П. Формальные грамматики и распознающие автоматы. Учеб. пособие. – Л.: ЛЭТИ, 1991.
12. Опалева Э. А., Самойленко В. П., Семенова О. Н. Разработка языковых процессоров. Методические указания к курсовой работе. – СПб.: СПбГЭТУ, 2002.
13. Пратт., Зелковиц М. Языки программирования: реализация и разработка. – СПб.: Питер, 2001.
14. Рейуорд-Смит В. Дж. Теория формальных языков. Вводный курс. – М.: Радио и связь, 1988.
15. Себеста Р. У. Основные концепции языков программирования. / 5-е из; Пер. с англ. – М.: Издательский дом "Вильямс, 2001.

16. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов языков и вычислений. / 2-е изд. Пер. с англ. – М.: Издательский дом Вильямс, 2002.

17. Пентус А.Е. Математическая теория формальных языков. - М.: Изд-во "Интернет университет информационных технологий - ИНТУИТ.ру", 2006. - 248 с.: ил.

18. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. 2-е издание. - М.: Издательский дом "Вильямс", 2002. - 528 с.

19. Хантер Р. Основные концепции компиляторов. - М.: Издательский дом "Вильямс", 1984. - 256 с.

20. Соколов А.П. Системы программирования: теория, методы, алгоритмы. Учебное пособие. - М.: Финансы и статистика, 2004. - 320 с.

21. Свердлов С. Языки программирования и методы трансляции: Учебное пособие 27. Пратт Т., Зелковиц М. Языки программирования: разработка и реализация. 4-е изд. - СПб.: Питер, 2002. - 688 с.: ил.

22. Керниган Б., Ритчи Д. Язык программирования Си. — 2-е изд. — М.: Вильямс, 2007. — С. 304. — ISBN 0-13-110362-8.

23. Гукин Д. Язык программирования Си. — М.: Диалектика, 2006. — С. 352. — ISBN 0-7645-7068-4.

24. Подбельский В. В., Фомин С. С. Курс программирования на языке Си: учебник. — М.: ДМК Пресс, 2012. — 318 с. — ISBN 978-5-94074-449-8.

25. Прата С. Язык программирования С: Лекции и упражнения. — М.: Вильямс, 2006. — С. 960. — ISBN 5-8459-0986-4.

26. Прата С. Язык программирования С (С11). Лекции и упражнения, 6-е издание. — М.: Вильямс, 2015. — 928 с. — ISBN 978-5-8459-1950-2.

27. Шилдт Г. С: полное руководство, классическое издание. — М.: Вильямс, 2010. — С. 704. — ISBN 978-5-8459-1709-6.

Составил: к.т.н., доцент кафедры «Вычислительная и прикладная математика» Антипов О.В.

Заведующий кафедрой вычислительной и прикладной математики, д-р техн. наук, профессор Овечкин Г.В.