МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВПО ВГУ)**

Факультет романо-германской филологии

Кафедра теории и методики преподавания иностранных языков

Лингвистика

**Курсовая работа**

на тему: «Основы программирования. История и современное состояние»

Автор курсовой работы: Опачанова Дарья Олеговна

Руководитель работы: Донина Ольга Валерьевна, к.ф.н.

Работа сдана «22»октября 2018г.

Оценка

Воронеж 2018

Оглавление

[Введение 3](#_Toc527938617)

[1. Структурное и модульное программирование 4](#_Toc527938618)

[2. Объектно-ориентированное программирование 6](#_Toc527938619)

[3. Компонентное программирование 7](#_Toc527938620)

[4. Применение структурных и объектно-ориентированных методов программирования 10](#_Toc527938621)

[5. Новые направления в программировании 12](#_Toc527938622)

[6. Узконаправленные языки 13](#_Toc527938623)

[7. Создание языка C 14](#_Toc527938624)

[8. Пролог и Ада 14](#_Toc527938625)

[9. Классификация 14](#_Toc527938626)

[10. Направления развития 16](#_Toc527938627)

[Заключение 20](#_Toc527938628)

[СПИСОК ЛИТЕРАТУРЫ 21](#_Toc527938629)

# Введение

Прогресс компьютерных технологий определил процесс появления новых разнообразных знаковых систем для записи алгоритмов – языков программирования. Смысл появления такого языка – упрощение программного кода. С каждым днём наш мир становиться более мобильным и информационным. Всё больше и больше компьютеры вступают в нашу повседневную жизнь и чтобы облегчить наше общение с ними создаётся новое ПО с помощью различных языков программирования. Целью данной курсовой работы является изучение истории возникновения программирования и основных принципов и подходов при создании языка программирования.

Изначально программирование имело крайне примитивный вид и практически не имело отличий от упорядоченного бинарного кода с формализованным подходом. По сути, при зарождении сферы отличий языка программирования от компьютерного кода было немного. Очевидных и естественных удобств для программиста не существовало, он обязан был обладать знаниями числовых кодов для каждой команды машины. Даже распределение памяти для выполнения команд ложилось на специалиста.

Для упрощения обращения с ЭВМ люди стали активно разрабатывать языки, одним из первых стал **Ассемблер**. Для отображения переменных стали использоваться символьные наименования. Вместо числовых операций человеку достаточно знать мнемонические имена, их запоминание в разы облегчалось. Уже на этом этапе языки программирования стали более приближёнными к понятному для человека языку.

К первооткрывателям среди языков программирования относится Фортран – это сокращённое сочетание 2 слов: Formula и Translation. Создан уже в середине 50-х. До сих пор язык используется благодаря лёгкости и простоте написания, а также развитой системе библиотек для Фортран. Чаще используется для научных и инженерных подсчётов, а также активно применяется в физичке и остальных науках, связанных с математикой.

# Структурное и модульное программирование

В 1965 г. итальянцы Бом и Джакопини предложили использовать в качестве базовых алгоритмических элементов следование, ветвление и цикл. Почти в то же время к аналогичным выводам пришел голландский ученый Э. Дийкстра, заложивший основы структурного программирования. В 1970-х гг. эта методология оформилась, и корпорация IBM сообщила о применении в разработке программного обеспечения «Усовершенствованных методов программирования», одним из компонентов которых являлась технология нисходящего структурного программирования, основу которого составляет следующее:

* сложная задача разбивается на простые, функционально управляемые задачи, каждая задача имеет один вход и один выход; управляющий поток программы состоит из совокупности элементарных функциональных подзадач;
* управляющие структуры просты, т. е. логическая задача должна состоять из минимальной, функционально полной совокупности достаточно простых управляющих структур;
* программа разрабатывается поэтапно, на каждом этапе решается ограниченное число точно поставленных задач.

Четко сформулированные основы нисходящей разработки, структурного кодирования и сквозного контроля позволяли перейти к промышленным методам разработки программного обеспечения.

Развитие получило модульное программирование, основа которого заключается в следующем:

\* функциональная декомпозиция (разбиение) задачи на самостоятельные подзадачи -- модули, связанные только входными и выходными данными;

\* модуль представляет собой «черный ящик», позволяющий разрабатывать части программ одного проекта на разных языках программирования, а затем с помощью компоновочных средств объединять их в единый загрузочный модуль;

\* должно быть ясное понимание назначения всех модулей задачи и их оптимального сочетания;

\* с помощью комментариев должно описываться назначение всех переменных модуля.

В период 1970--1980-х гг. развитие теоретических исследований оформило программирование как самостоятельную научную дисциплину, занимающуюся методами разработки программного обеспечения (ПО).

В истории развития промышленного программирования большую роль сыграл программист и бизнесмен Билл Гейтс (Gates William Henry, p. в 1955 г.). Гейтс и его школьный товарищ Пол Аллен основали компанию по анализу уличного движения «Трэф-О-Дейта» и использовали для обработки данных компьютеры с микропроцессором 8008 -- первым из знаменитого ряда микропроцессоров компании «Intel». Будучи студентом Гарвардского университета, в 1975 г. он совместно с Алленом написал для компьютера Altair (фирмы M1TS) интерпретатор – программу переводчик с языка программирования на язык машинных кодов.

Профессиональное программирование вышло на уровень технологии.

Методы разработки ПО синтезируют:

\* методы инженерных расчетов для оценки затрат и выбора решений;

\* математические методы для составления алгоритмов;

\* методы управления для определения требований к системе, учета ситуаций, организации работ и прогнозирования.

# Объектно-ориентированное программирование

На смену структурному программированию в начале 1990-х гг. пришло объектно-ориентированное программирование -- ООП. Его можно рассматривать как модульное программирование нового уровня, когда вместо во многом случайного, механического объединения процедур и данных главным становится их смысловая связь. Объект рассматривается как логическая единица, которая содержит данные и правила (методы) их обработки. Объектно-ориентированный язык создает «программное окружение» в виде множества независимых объектов, каждый из которых отличается своими свойствами и способами взаимодействия с другими объектами. Программист задает совокупность операций, описывая структуру обмена сообщениями между объектами. Как правило, он «не заглядывает» внутрь объектов, но при необходимости может изменять элементы внутри объектов или формировать новые.

ООП основано на трех важнейших принципах (инкапсуляция, наследование, полиморфизм), придающих объектам новые свойства.

1. Инкапсуляция -- объединение в единое целое данных и алгоритмов их обработки. Данные здесь -- поля объекта, а алгоритмы -- объектные методы.
2. Наследование -- свойство объектов порождать своих потомков. Объект-потомок автоматически наследует все поля и методы, может дополнять объекты новыми полями, заменять и дополнять методы.
3. Полиморфизм -- свойство родственных объектов решать схожие по смыслу проблемы разными способами.

Идея использования программных объектов исследовалась в течение ряда лет разными учеными. Одним из первых языков этого типа считают Simula-67.А в 1972 г. появился язык Smoltalk, разработанный Аланом Кеем, утвердивший статус ООП.

На современном этапе развиваются инструментальные среды и системы визуального программирования для создания программ на языках высокого уровня: (Turbo Pascal, Delphi, Visual Basic, C++Builder и др.).

# Компонентное программирование

Развитие основных принципов объектно-ориентированного программирования получило с появлением компонентного программирования (КП). КП -- динамический процесс без жестких правил, выполняющийся в основном для распределенной разработки (программирования) распределенных систем. Суть КП в том, что независимые проектировщики, программисты разрабатывают независимые компоненты (отдельные части) единой системы, распределенные по множеству узлов большой сети. Эти части могут принадлежать разным собственникам и управляться организационно независимыми администраторами.

В КП компонент рассматривается как хранилище (в виде DLL-или ЕХЕ файлов) для одного или нескольких классов. Классы распространяются в бинарном виде, а не в виде исходного кода. Предоставление доступа к методам класса осуществляется через строго определенные интерфейсы по протоколу. Это снимает проблему несовместимости компиляторов, обеспечивая без перекомпиляции смену версий классов в разных приложениях. Интерфейсы задают содержание сервиса и являются посредником между клиентом и сервером.

Фирма Microsoft создала технологии для распределенной разработки распределенных систем, такие как COM (Component Object Model), COM+, .NET. Разработаны и другие технологии: CORBA (консорциума OMG), JAVA (компании Sun Microsystem) и др.

Идея переложить на ЭВМ функции составителей алгоритмов и программистов дала новые возможности развитию сферы искусственного интеллекта, которая должна была создавать методы автоматического решения интеллектуальных задач. Формализация знаний, которые есть у профессионалов в разных областях, накопление их в базах знаний, реализованных на ЭВМ, стали основанием для создания экспертных систем. На основе баз знаний работают и ЭВМ V поколения, и интеллектуальные роботы, и экспертные системы. Эти системы могут не только найти решение той или иной задачи, но и объяснить, как оно получено. Появилась возможность манипулировать знаниями, иметь знания о знаниях -- метазнания. Знания, хранящиеся в системе, стали объектом ее собственных исследований.

Независимость языков высокого уровня от ЭВМ вовлекла в сферу алгоритмизации задач специалистов различных отраслей знаний, позволила использовать многочисленные стандартные типовые программы, а программистам -- устранять дублирование в написании программ для различных типов ЭВМ и значительно повысить производительность труда.

В конце 1980-х гг. в Японии и США появились проекты ЭВМ V поколения, реализованные в конце 1990-х гг. Прогресс в программировании связан с прогрессом в архитектуре вычислительных систем, отходом от фон-неймановской концепции, с достижениями в области искусственного интеллекта. Революционные изменения в элементной базе ЭВМ связываются с исследованиями по биоэлектронике.

На современном этапе программирование включает комплекс вопросов, связанных с написанием спецификаций (условий задач), проектированием, кодированием, тестированием и функционированием программ для ЭВМ. Современное ПО для ЭВМ имеет сложную структуру и включает, как правило, ОС, трансляторы с различных языков, текстовые программы контроля и диагностики, набор обслуживающих программ. Например, японские ученые для проектирования систем ПО разрабатывают идею «кольцевой структуры» шести уровней:

1. (внутренний) программы для аппаратуры;
2. ядро ОС;
3. программы сопряжения;
4. часть ОС, ориентированная на пользователя;
5. системы программирования;
6. (внешний) программы пользователя.

Согласно этим проектам научных исследований планируется упростить процесс создания программных средств путем автоматизации синтеза по спецификациям исходных требований на естественных языках. В последнее время в Японии удалось создать робота-переводчика, переводящего английскую речь на японский язык и наоборот, осуществляя это голосом человека. Во всех развитых странах работают над комплексами программ для создания роботов. Для многих сфер человеческой деятельности.

# Применение структурных и объектно-ориентированных методов программирования

Широкое применение структурных и объектно-ориентированных методов программирования с использованием графических моделей объединялось отсутствием инструментальных средств. Это породило потребность в программно-технологических средствах специального класса -- CASE (Computer Aided Software Engineering), реализующих технологию создания и сопровождения ПО различных систем. Предпосылки для появления CASE-технологий возникли к концу 1980-х гг. Первоначально термин «CASE» применялся только к вопросам автоматизации разработки ПО, теперь программная инженерия имеет более широкое значение для разработки систем в целом. В CASE-технологии входит разработка и внедрение языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описания системных требований.

В начале XXв. с созданием пишущей механической машинки появилась возможность общедоступного создания печатного текста, хотя внесение изменений в такой текст (исправление ошибок) было достаточно трудоемкой работой. Затем появились электрические пишущие машинки. С появлением персональных компьютеров подготовка печатного текста стала гораздо совершеннее. В последние два десятилетия прошлого века уже разрабатывается множество комплексов программ для обработки текстов, которые сначала получили название текстовых редакторов, а по мере расширения их функциональных возможностей -- текстовых процессоров.

В начале этого столетия текстовые процессоры стали более совершенными. Наряду с более простыми (например Professional Write и др.) появились такие мощные, как MS WinWord, WordPerfect WordStar 2000 и др. Из отечественных широкое распространение получил текстовый процессор Лексикон.

С начала 1980-х гг. для подготовки и обработки числовой информации стали использоваться табличные процессоры. В 1979 г. Д. Брикклин предложил первую программу для работы с электронными таблицами VisiCalc. В 1981 г. была разработана система SuperCalc фирмы «Computer Associates», в 1982 г. -- Multiplan фир-мы «Microsoft», далее -- пакет для IBM PC Lotusl-2-3 фирмы «Lotus Development», русифицированные пакеты АБАК, ДРАКОН и др. В 1985 г. появился табличный процессор Excel фирмы «Microsoft» первоначально для персонального компьютера Macintosh, а затем для совместимых с IBM PC. Этот процессор разрабатывался параллельно с ОС Windows, его версии вобрали в себя все черты графиче-ского интерфейса, вплоть до версий Excel 5.0 как приложения Windows 3.1, Excel 7.0 как приложения Windows 95 и т. д. В последние годы создано достаточно много систем подготовки табличных документов, т. е. электронных таблиц, табличных процессоров (например, Corel Quattro 6.0 фирмы «Corel Co», Lotus 5.0 фирмы «Lotus Development Co», Office Proftessional for Windows фирмы «Microsoft» и ДР-)- Но наиболее широко используют электронные таблицы Excel.

Разработано большое количество стандартных реляционных систем управления базами данных -- СУБД (например, MS Access, paradox и др.), на основе которых строят реляционные базы данных в различных предметных областях.

Для многих организаций (особенно управленческих) разработаны так называемые офисные пакеты, в которых на основе единой ОС функционируют приложения, включающие в себя системы для работы с различными видами информации. Например, созданы пакеты приложений к ОС Windows (MS Office, WordPerfect Office фирмы «Corel», StarOffice фирмы «SunMicrosystems» и др.), которые включают программные средства для выполнения функций обработки всех видов информации. Например, MS Office включает совершенствующиеся год от года (в зависимости от последней версии ОС Windows) средства обработки текста (MS Word), графики (Photo Draw) и презентаций (PowerPoint), таблиц (Excel), баз данных (Access), электронной почты (Outlook), работы во Всемирной паутине (FrontPage), создания звуковых клипов (MS Sound Recorder).

# Новые направления в программировании

Мощным толчком в развитии новых направлений в программировании послужило объединение компьютерных и телекоммуникационных технологий.

За рубежом в 1960-х гг. появились первые вычислительные сети, с которых началась техническая и технологическая революция, т. к. была предпринята попытка объединить технологию сбора, хранения, передачи и обработки информации на ЭВМ с техникой связи. В Европе в те годы были созданы международные сети EIN и Евро-нет, затем появились национальные сети. В 1972 г. в Вене была создана сеть МИПСА, к которой присоединились в 1979 г. 17 стран Европы, СССР, США, Канада и Япония. В 1980-х гг. в нашей стране была создана система телеобработки статистической информации, обслуживающая государственные и республиканские органы статистики.

# Узконаправленные языки

Из-за увеличения сфер использования ЭВМ появились и другие языки для отдельных разработок в новых сферах:

экономическое направление оставалось незанятым до появления Кобол;

Снобол – обрабатывает алгоритмы, связанные с текстами;

Лисп. Работает на основании алгоритмов для отработки символов. Активно используется для формирования искусственного интеллекта.

Уже в 1968 г. был впервые запущен конкурс, в котором главным местом являлось звание лучшего языка программирования для начала карьерного пути. Данные планировалось использовать для обучения специалистов. Победу одержал Алгол-68, но он остался малоизвестным, о популярности и речь не идёт.

Специально для участия в конкурсе был создан Паскаль, разработчиком являлся Никлаус Вирт. Язык весьма доступный, удобный и объединяет немало мощных инструментов для структурирования информации. Несмотря на изначальную разработку с целью обучения студентов, Паскаль получил широкое распространение и активно развивался. Даже сегодня он является одним из лучших и известнейших языков программирования.

Для обучения детей в школах был создан Лого, у истоков стоял Самуэль Пайперт. Достоинства – простота работы и обилие возможностей.

В школах стал преподаваться простой язык Бейсик, он легко взаимодействует с ЭВМ в качестве прямого диалога. Время никак не повлияло на эту сферу, до сих пор Бейсик является самым простым языком для начала изучения большинства распространённых направлений программирования.

# Создание языка C

Развитие возможностей вычислительного оборудования привело к необходимости написания ёмких программ для управления ЭВМ. Это место по праву занял язык Си, который стал активно использоваться в 70-х годах. Явным достоинством языка является его универсальность. Он превосходит Паскаль благодаря наличию вложенных возможностей сотрудничества с разными машинными командами и подходящими частями памяти.

Си используется повсеместно в качестве инструментального языка для написания операционных платформ, трансляционных устройств, баз данных и остальных прикладных, системных задач. Си не имеет чёткой направленности, он подходит для многих задач из-за эффективности, лёгкости переноса и экономного потребления ресурсов. Чаще всего Си по скорости обработки данных сопоставим с Ассемблером, производительность программ на обоих языках будет приблизительно равной. В небольшом языке заложена немалая мощность.

# Пролог и Ада

Внедрение функционального программирования неизбежно повлекло создание Пролога. Задачи языка сводились к анализу и взаимодействию с человеческими языками. Логика приложения формальна, она оптимально подходит для автоматического решения задач и теорем.

Только в 80-х годах был разработан язык Ада. Он расширяет классическое понимание и свойства языков того периода. Ада могла решать задачи в режиме реального времени и моделировать независимые решения.

# Классификация

Сегодня разработаны классификации языков по уровню работы, это распределение самое распространённое. Выделяют 3 основных уровня:

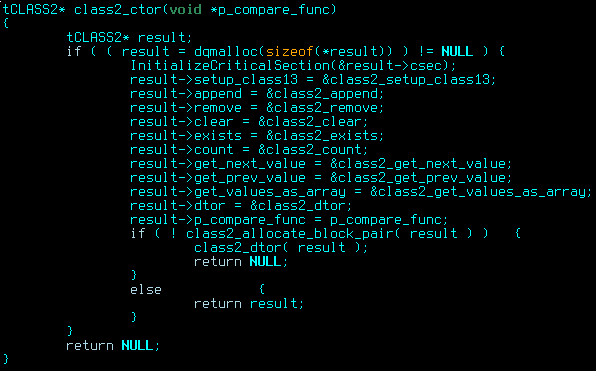
Табл.1 «Классификация языков»

|  |  |
| --- | --- |
| 1. Низкий | Сюда относятся различные машинные языки или разновидности с символическим кодирование типа Ассемблера и Автокода. За основу взяты операторы машинных команд, только разработаны с привязкой к мнемоническому коду. Операндами являются уже не точные адреса, а символьное обозначение имён. Все модели разработаны для отдельных разновидностей ПК, они являются машинно-зависимыми. В подобных языках отмечается сильная зависимость языка от внутренних особенностей системы; |
| 1. Высокий | Языки встречаются куда чаще, они более удобны в использовании. К ним причисляются: Алгол, С, Пролог, Паскаль, Бейсик, Фортран и другие. Перечисленные языки не имеют жёсткой зависимости от машины, ведь они основываются на возможностях системы операндов, которые подобны для классовых алгоритмов. Недостатками высокого уровня являются большая ресурсоёмкость и медленное исполнение; |
| 1. Сверхвысокий | Представителей языков крайне мало, только APL и Алгол-68. Их считают сверхвысокого уровня из-за разработки сверхмощных операторов. |

Согласно другой классификации языки делятся на:

• символьные – Пролог, Лисп и Снобол;

• вычислительные – Паскаль, С, Алгол, Бейсик, Фортран.



# Направления развития

Информатика в современном мире развивается в 3 ключевых направлениях:

Табл.2 «Направления развития»

|  |  |
| --- | --- |
| 1. Процедурное | появилось в период активнейшего развития компьютеров и других вычислительных устройств, с тем пор широко используется. В процедурных направлениях присутствуют выраженные описания действий, необходимых к выполнению. Для получения результата всегда проводится определённая процедура, которую составляют различные последовательности манипуляций. Процедурные языки дополнительно разделяются на:   * **Структурные.** В них используется один оператор для записи цельных алгоритмом: циклов, функцию, ветвлений и остального. Более известны: Паскаль, Ада и С. * **Операционные.** Применяют несколько различных действий. Среди самых известных разновидностей: *Фокал, Фортран и Бейсик*. |
| 1. Непроцедурные | Языки программирования имеют декларативную структуру, появление которой приходится на 70-е года. Активнее всего начали развиваться в 80-х годах после появления проекта формирования 5 поколения ЭВМ. Основная задача – создание возможностей для построения *высокоинтеллектуальных* машин. Они также разделяются на:   * **Функциональные.** Программа выполняет исчисление определённой функции, которая берёт за основу другие относительно простые алгоритмы и более простые задачи. В основе функционального направления используется основной элемент – рекурсия. Она подразумевает расчёт значений функции с помощью задействования её в других элементах. В языке отсутствуют циклы и методика присваивания значений. * **Логические.** Программа вовсе не требует описание действий, её основу составляют соотношения данных и их значения. Только после расчёта можно получать ответы на вопросы. После перебирания известных параметров выводится ответ. В программе отсутствует метод или порядок обнаружения ответа, он неявным образом устанавливается языком. Ярким представителем является Пролог. Из направления полностью устранено алгоритмическое мышление, только статические отношения между объектами, а вся динамика сокрыта от разработчика и сводится к перебору данных. |
| 1. Объектно-ориентированные языки | все они являются разновидностью высокого уровня программирования. Подобные языки не нуждаются в описании чёткой последовательности манипуляций для получения результата задачи, но отдельные компоненты процедурного направления присутствуют. Пользователям значительно проще работать с такими языками, так как они обладают доступным и богатым интерфейсом. Лучшим примером подобного направления с визуальным общением является Object Pascal. |



Существуют языки для написания сценариев, известными являются Rexx, Tcl, Perl и Python, а также языки оболочек систем Unix. В них разрабатывается индивидуальный стиль написания кода, который отличается от известного принципа системного уровня программирования. Они не используются для создания приложений на нижнем уровне, скорее для комбинирования различных компонентов из разных языков, из которых составляется набор отдельных функций.

Активнее всего стали развиваться по мере распространения интернета, от чего стали широко применяться языки сценариев. Для создания сценариев чаще всего применяется Perl, а для Web-части пользуется популярностью JavaScript.

# Заключение

Современные информационные технологии– это совокупность методов, производственных процессов и программно-технических средств, объединенных в технологическую цепочку, обеспечивающую сбор, регистрацию, обработку, накопление, хранение, отображение, поиск, анализ, защиту и распространение информации. В основе современных информационных технологий лежит вычислительная техника.

Современный этап развития ЭВМ можно охарактеризовать как этап развития машинного интеллекта. Вычислительные системы будущего будут ориентированы на обработку знаний и должны располагать развитыми возможностями логического вывода. Важнейшая черта их должна состоять в том, чтобы используемый интерфейс был непосредственно рассчитан на человека. Главными особенностями машин будущего будут речевой ввод-вывод информации и самообучаемость. Технический базис ее должна составить развивающаяся технология сверхбольших интегральных схем, создание памяти повышенного объема, возрастающие возможности высокоскоростных элементов.

В процессе написания данной курсовой работы мною была изучена история возникновения самого программирования. Также была проведена систематизация знаний о подходах и принципах создании новых языков программирования. В заключении следует отметить, что рассмотренная тема, позволяет просмотреть путь становления технологий и языков программирования и является интересной с точки зрения специалиста в области информационных технологий.

# СПИСОК ЛИТЕРАТУРЫ

1. Немнюгин С.А. Turbo Pascal (практикум): СПб.: «Питер», 2003. – 475с.
2. Кирнос В.Н. Информатика 10 – 11 класс: Кокшетау: «Кешелек»,2005. – 208с.
3. Фаронов В.В. Turbo Pascal 7.0. Начальный курс. - Нолидж, 1998. -620 с.
4. Грызлов В.И., Грызлова Т.П. Турбо Паскаль 7.0. - М.: "ДМК", 2000. - 416 с.
5. Зуев Е.А. Turbo Pascal. Практическое программирование. - Приор,1997. - 336с.
6. Пильщиков В.Н. Сборник упражнений по языку Паскаль. – М.: Наука, 1989.– 160 с.
7. Йенсен К., Вирт Н. Руководство для пользователя и описание языка. – М.,1982. 151 с.
8. Абрамов С.А., Зима Е.В. Начала программирования на языке Паскаль. – М.:Наука, 1987. – 112 с.
9. Павловская Т.А. Паскаль. Программирование на языке высокого уровня. - СПб: Питер, 2003.-393с.
10. Эллиот Б.К. Turbo Pascal = Turbo Pascal Web Update. — М.: Вильямс, 2005. — 896с.
11. Лукин С.Н. TURBO PASCAL 7.0. Самоучитель для начинающих.- Диалог-МИФИ, 2005.-

400с.

1. Епанешников А. М., Епанешников В. А. Программирование в среде Turbo Pascal 7.0.- 1995.-278с.
2. Немнюгин С., Перколаб Л. Изучаем Turbo Pascal.- Питер, 2007.- 320с.