

# Современные языки программирования и .NET

Двухсеместровый учебный курс

I семестр: основы функционального  
программирования и computer science

II семестр: разработка гетерогенных  
программных систем

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

## Комментарий к слайду

Предлагаем Вашему вниманию курс лекций, посвященный современным языкам программирования. Изучение языков программирования базируется на теоретическом фундаменте современных подходов и математических формализаций, принятых в мировом computer science.

В качестве технологической основы и инструментальной платформы для исследования языков программирования предлагается новейшая разработка корпорации Microsoft – уникальный по идеологии и широте поддержки языков программирования комплекс программного обеспечения на основе так называемой методологии .NET.

Курс имеет целью введение в теорию и практику разработки программных систем и состоит из двух взаимосвязанных этапов, преподавание которых физически разделено по времени на два семестра.

Первый семестр посвящен основам программирования и теоретическому введению в computer science.

Второй семестр предполагает (на основе знакомства с азами теории и практики программирования) более профессиональные аспекты реализации гетерогенных программных систем.

## Содержание лекции

1. Предмет и назначение курса
2. Структура курса
3. Классификация языков программирования
4. Преимущества и недостатки основных классов языков программирования
5. Основные подходы к программированию
6. Библиография

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Коротко о содержании лекции.

Прежде всего, необходимо познакомить слушателей с предметом учебного курса и его назначением.

Затем речь пойдет о структуре курса и назначении каждого из его этапов.

Собственно в лекции слушателям будет предложен вариант классификации языков программирования, рассмотрена история их развития и основные подходы к разработке программных систем, отмечены преимущества и недостатки каждого из языков и подходов.

Наконец, для желающих глубже исследовать предмет будут представлены ссылки на важнейшие работы теоретического и практического плана по теме лекции.

Предмет курса: современные языки программирования

Назначение курса: фундаментальное введение в современную теорию и практику программирования

Цель курса: формирование адекватного мировоззрения на разработку программного обеспечения в современных условиях и «хорошего стиля» программирования

Аудитория: студенты технических вузов, специализирующихся на программировании

Начальные знания: минимальны (фундаментальные понятия из школьного курса математики: «функция», «множество», ... на интуитивном уровне)

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Как явствует из названия, основным предметом настоящего учебного курса являются языки программирования.

Курс предназначен для фундаментального, глубокого ознакомления студентов с основами современных тенденций в теоретическом и практическом программировании. При этом в теоретическом аспекте курс опирается на апробированный фундамент семейства научных дисциплин, известных под названием computer science.

К сожалению, большинство из известных в настоящее время курсов либо изобилуют излишними техническими деталями и отвлекают слушателей от глубинной сути программирования, либо представляют собой сухие теоретические дисциплины, достаточно далеко отстоящие от практики программирования.

Нашей задачей будет формирование верного понимания основ разработки программного обеспечения в теоретическом и практическом аспектах, а также хорошего тона, вкуса, или, как принято говорить, стиля программирования, исходя из сравнительного анализа наиболее значимых современных подходов.

Существенным преимуществом курса является минимум базовых требований для слушателей, преимущественно студентов младших курсов технических вузов, специализирующихся на программировании, от которых требуется владение базовыми понятиями математики, фактически изучаемых еще в курсе средней школы: «функция», «множество» и т.п.

# Классификация языков программирования

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

## Комментарий к слайду

Перейдем непосредственно к теме первой лекции, которая посвящена истории, эволюции и классификации языков и подходов к программированию.

## Ранние языки программирования

Время появления: 1940-е г.г.

Краткая характеристика: линейная последовательность элементарных инструкций «низкого уровня»

Преимущества:

- высокая вычислительная эффективность

Недостатки:

- существенная зависимость от среды вычислений

Примеры:

- машинные коды, ассемблеры

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Первые языки программирования возникли относительно недавно. Различные исследователи указывают 20-е, 30-е и даже 40-е годы XX столетия. Нашей задачей является не установление самого раннего языка, а поиск закономерностей в их развитии.

Как и следовало ожидать, первые языки программирования, как и первые ЭВМ, были довольно примитивны и ориентированы на численные расчеты. Это были и чисто теоретические научные расчеты (прежде всего, математические и физические), и прикладные задачи, в частности, в области военного дела.

Программы, написанные на ранних языках программирования, представляли собой линейные последовательности элементарных операций с регистрами, в которых хранились данные.

Нужно отметить, что ранние языки программирования были оптимизированы под ту аппаратную архитектуру конкретного компьютера, для которого они предназначались, и хотя они обеспечивали высокую эффективность вычислений, до стандартизации было еще далеко. Программа, которая была вполне работоспособной на одной вычислительной машине, зачастую не могла быть выполнена на другой.

Таким образом, ранние языки программирования существенно зависели от того, что принято называть средой вычислений и приблизительно соответствовали современным машинным кодам или языкам ассемблера.

## Императивные (процедурные) языки программирования (1)

Время появления: 1950-е г.г.

Краткая характеристика:

программа – последовательность инструкций-*операторов*, включающих блоки типичных действий – процедуры или функции

Преимущества:

- более высокий уровень абстракции;
- меньшая машинная зависимость;
- более широкая совместимость

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Следующее десятилетие ознаменовалось появлением языков программирования так называемого «высокого уровня», по сравнению с ранее рассмотренными нами предшественниками, соответственно именуемыми низкоуровневыми языками.

При этом различие состоит в повышении эффективности труда разработчиков за счет абстрагирования или отвлечения от конкретных деталей аппаратного обеспечения. Одна инструкция (оператор) языка высокого уровня соответствовала последовательности из нескольких низкоуровневых инструкций, или команд. Исходя из того, что программа, по сути, представляла собой набор директив, обращенных к компьютеру, такой подход к программированию получил название императивного.

Еще одной особенностью языков высокого уровня была возможность повторного использования ранее написанных программных блоков, выполняющих те или иные действия, посредством их идентификации и последующего обращения к ним, например, по имени. Такие блоки получили название функций или процедур, и программирование приобрело более упорядоченный характер.

## Императивные (процедурные) языки программирования (2)

Преимущества:

- содержательная значимость текстов программ;
- унификация программного кода;
- повышение производительности труда программистов

Недостатки:

- большие трудозатраты на обучение;
- меньшая эффективность программного кода

Примеры:

Fortran, ALGOL, PL/1, APL, BPL, COBOL, Pascal, C, Basic

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Кроме того, с появлением языков высокого уровня зависимость реализации от аппаратного обеспечения существенно уменьшилась. Платой за это стало появление специализированных программ, преобразующих инструкции возникших языков в коды той или иной машины, или трансляторов, а также некоторая потеря в скорости вычислений, которая, впрочем, компенсировалась существенным выигрышем в скорости разработки приложений и унификацией программного кода.

Нужно отметить, что операторы и ключевые слова новых языков программирования были более осмысленны, чем безликие цифровые последовательности кодов, что также обеспечивало повышение производительности труда программистов.

Естественно, для обучения новым языкам программирования требовались значительные затраты времени и средств, а эффективность реализации на прежних аппаратных возможностях снижалась. Однако трудности эти носили временный характер, и, как показала практика программирования, многие из первых языков высокого уровня оказались настолько удачно реализованными, что активно используются и сегодня.

Одним из таких примеров является язык Fortran, реализующий вычислительные алгоритмы. Другой пример - язык APL, трансформировавшийся в BPL и затем в C. Основные конструкции последнего остаются неизменными вот уже несколько десятилетий и присутствуют в языке C#, который нам предстоит изучить.

Примеры других языков программирования: ALGOL, COBOL, Pascal, Basic.

## Декларативные языки программирования (1)

Время появления: 1960-е г.г.

Краткая характеристика: программа – описание действий,  
которые необходимо осуществить

Преимущества:

- простота верификации и тестирования программ;
- строгость математической формализации;
- высокая степень абстракции

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

В 60-х г.г. возникает новый подход к программированию, который до сих пор успешно конкурирует с императивным, а именно, декларативный подход.

Суть подхода состоит в том, что программа представляет собой не набор команд, а описание действий, которые необходимо осуществить.

Этот подход, как мы увидим впоследствии, существенно проще и прозрачнее формализуем математическими средствами. Отсюда следует тот факт, что программы проще проверять на наличие ошибок (тестировать), а также на соответствие заданной технической спецификации (верифицировать).

Высокая степень абстракции также является преимуществом данного подхода. Фактически, программист оперирует не набором инструкций, а абстрактными понятиями, которые могут быть достаточно обобщенными.

## Декларативные языки программирования (2)

Недостатки:

- сложность эффективной реализации;
- необходимость фундаментальных математических знаний

Примеры:

LISP (Interlisp, Common Lisp, Scheme), SML, Haskell, Prolog

### Комментарий к слайду

На начальном этапе развития декларативным языкам программирования было сложно конкурировать с императивными в силу объективных трудностей при создании эффективной реализации трансляторов. Программы работали медленнее, однако, они могли решать с меньшими трудозатратами более абстрактные задачи.

В частности, язык SML, который мы будем изучать в данном курсе, был разработан как средство доказательства теорем.

Различные диалекты языка LISP (основные из них представлены на слайде), возникли потому, что ядро и идеология этого языка оказались весьма эффективными при реализации символьной обработки (анализе текстов).

Другие характерные примеры декларативных языков программирования приведены на слайде.

## Функциональные языки программирования (1)

Время появления: 1960-е г.г.

Краткая характеристика:

программа – функция, аргументы которой, возможно, также являются функциями

Преимущества:

- полностью автоматическое управление памятью компьютера («сборка мусора»);
- простота повторного использования фрагментов кода;
- расширенная поддержка функций с параметрическими аргументами (параметрический полиморфизм);

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Одним из путей развития декларативного стиля программирования стал функциональный подход, возникший с появлением и развитием языка LISP.

Отличительной особенностью данного подхода является то обстоятельство, что любая программа, написанная на таком языке, может интерпретироваться как функция с одним или несколькими аргументами. Такой подход дает возможность прозрачного моделирования текста программ математическими средствами, а, значит, весьма интересен с теоретической точки зрения.

Сложные программы при таком подходе строятся посредством агрегирования функций. При этом текст программы представляет собой функцию, некоторые аргументы которой можно также рассматривать как функции. Таким образом, повторное использование кода сводится к вызову ранее описанной функции, структура которой, в отличие от процедуры императивного языка, прозрачна математически.

Более того, типы отдельных функций, используемых в функциональных языках, могут быть переменными. Таким образом обеспечивается возможность обработки разнородных данных (например, упорядочение элементов списка по возрастанию для целых чисел, отдельных символов и строк) или полиморфизм.

Еще одним важным преимуществом реализации языков функционального программирования является автоматизированное динамическое распределение памяти компьютера для хранения данных. При этом программист избавляется от рутинной обязанности контролировать данные, а при необходимости может запустить функцию «сборки мусора» – очистки памяти от тех данных, которые больше не потребуются программе (обычно этот процесс периодически иницируется компьютером).

## Функциональные языки программирования (2)

Преимущества:

- абстрагирование от машинного представления данных;
- прозрачность реализации самоприменяемых (рекурсивных) функций;
- удобство символьной обработки данных (списки, деревья)

Недостатки:

- нелинейная структура программы;
- относительно низкая эффективность

Примеры:

LISP, SML, CaML, Haskell, Miranda, Hope

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Таким образом, при создании программ на функциональных языках программист сосредотачивается на области исследований (предметной области) и в меньшей степени заботится о рутинных операциях (обеспечении правильного с точки зрения компьютера представления данных, «сборке мусора» и т.д.).

Поскольку функция является естественным формализмом для языков функционального программирования, реализация различных аспектов программирования, связанных с функциями, существенно упрощается. В частности, интуитивно прозрачным становится написание рекурсивных функций, т.е. функций, вызывающих самих себя в качестве аргумента. Кроме того, естественной становится и реализация обработки рекурсивных структур данных (например, списков – базовых элементов, скажем, для семейства языков LISP, деревьев и др.)

Благодаря реализации механизма сопоставления с образцом, такие языки как ML и Haskell весьма хорошо применимы для символьной обработки.

Естественно, языки функционального программирования не лишены недостатков. Часто к ним относят нелинейную структуру программы и относительно невысокую эффективность реализации. Однако, первый недостаток достаточно субъективен, а второй успешно преодолен современными реализациями, в частности, рядом последних трансляторов языка SML, включая и компилятор для среды Microsoft .NET.

Характерные примеры декларативных языков программирования приведены на слайде.

## Логические языки программирования (1)

Время появления: 1970-е г.г.

Краткая характеристика:

программа – совокупность правил или логических высказываний с причиной и следствием

Преимущества:

- высокий уровень абстракции;
- удобство программирования логики поведения;
- удобство применения для экспертных систем;
- механизм откатов (backtrack)

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

В 70-х г.г. возникла еще одна ветвь языков декларативного программирования, связанная с проектами в области искусственного интеллекта, а именно, языки логического программирования.

Согласно логическому подходу к программированию, программа представляет собой совокупность правил или логических высказываний. Кроме того, в программе допустимы логические причинно-следственные связи, в частности, на основе операции импликации.

Таким образом, языки логического программирования базируются на классической логике и применимы для систем логического вывода, в частности, для так называемых экспертных систем. На языках логического программирования естественно формализуется логика поведения, и они применимы для описаний правил принятия решений, например, в системах, ориентированных на поддержку бизнеса.

Важным преимуществом подхода является достаточно высокий уровень машинной независимости, а также возможность откатов – возвращения к предыдущей подцели при отрицательном результате анализа одного из вариантов в процессе поиска решения (скажем, очередного хода при игре в шахматы), что избавляет от необходимости поиска решения полным перебором вариантов и увеличивает эффективность реализации.

## Логические языки программирования (2)

Недостатки:

- ограниченный круг задач;
- нелинейная структура программы;
- недостаточно эффективная реализация

Примеры:

Prolog, Mercury

### Комментарий к слайду

Одним из недостатков логического подхода в концептуальном плане является специфичность класса решаемых задач.

Другой недостаток практического характера состоит в сложности эффективной реализации для принятия решений в реальном времени, скажем, для систем жизнеобеспечения.

Нелинейность структуры программы является общей особенностью декларативного подхода и, строго говоря, является оригинальной характеристикой, а не объективным недостатком.

В качестве примеров языков логического программирования можно привести Prolog (название возникло от слов PROgramming in LOGic) и Mercury.

## Объектно-ориентированные языки программирования (1)

Время появления: 1970-е г.г.

Краткая характеристика:

программа – описание объектов, их совокупностей,  
отношений между ними и способов их взаимодействия

Преимущества:

- интуитивная близость к произвольной предметной области;
- моделирование сколь угодно сложных предметных областей;
- событийная ориентированность;

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Важным шагом на пути к совершенствованию языков программирования стало появление объектно-ориентированного подхода к программированию и соответствующего класса языков.

В рамках данного подхода программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов их взаимодействия и операций над объектами (или методов).

Несомненным преимуществом данного подхода является концептуальная близость к предметной области произвольной структуры и назначения. Механизм наследования атрибутов и методов позволяет строить производные понятия на основе базовых и таким образом создать модель сколь угодно сложной предметной области с заданными свойствами.

Еще одним теоретически интересным и практически важным свойством объектно-ориентированного подхода является поддержка механизма обработки событий, которые изменяют атрибуты объектов и моделируют их взаимодействие в предметной области.

## Объектно-ориентированные языки программирования (2)

Преимущества:

- высокий уровень абстракции;
- повторное использование описаний;
- параметризация методов обработки объектов

Недостатки:

- сложность тестирования и верификации программ

Примеры:

C++, Visual Basic, C#, Eiffel, Oberon

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Перемещаясь по иерархии классов от более общих понятий предметной области к более конкретным (или от более сложных – к более простым) и наоборот, программист получает возможность изменять степень абстрактности или конкретности взгляда на моделируемый им реальный мир.

Использование ранее разработанных (возможно, другими коллективами программистов) библиотек объектов и методов позволяет значительно сэкономить трудозатраты при производстве программного обеспечения, в особенности, типичного.

Объекты, классы и методы могут быть полиморфными, что делает реализованное программное обеспечение более гибким и универсальным.

Сложность адекватной (непротиворечивой и полной) формализации объектной теории порождает трудности тестирования и верификации созданного программного обеспечения. Пожалуй, это обстоятельство является одним из самых существенных недостатков объектно-ориентированного подхода к программированию.

Наиболее известным примером объектно-ориентированного языка программирования является язык C++, развившийся из императивного языка C. Его прямым потомком и логическим продолжением является язык C#. Другие примеры объектно-ориентированных языков программирования представлены на слайде.

## Языки сценариев (1)

Время появления: 1990-е г.г.

Краткая характеристика:

программа – совокупность описаний возможных сценариев обработки данных

Преимущества:

- интуитивная ясность;
- близость к предметной области;
- высокая степень абстракции;
- высокая переносимость

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Развитием событийно управляемой концепции объектно-ориентированного подхода стало появление в 90-х г.г. целого класса языков программирования, которые получили название языков сценариев или скриптов.

В рамках данного подхода программа представляет собой совокупность возможных сценариев обработки данных, выбор которых инициируется наступлением того или иного события (щелчок по кнопке мыши, попадание курсора в ту или иную позицию, изменение атрибутов того или иного объекта, переполнение буфера памяти и т.д.). События могут инициироваться как операционной системой (в частности, Windows), так и пользователем.

Основные достоинства языков данного класса унаследованы от объектно-ориентированных языков программирования. Это интуитивная ясность описаний, близость к предметной области, высокая степень абстракции, хорошая переносимость.

## Языки сценариев (2)

Преимущества:

- возможность повторного использования кода;
- совместимость с инструментальными средствами автоматизированного проектирования (CASE) и быстрой разработки (RAD) прикладного программного обеспечения

Недостатки:

- сложность тестирования и верификации программ;
- множественные побочные эффекты

Примеры:

VBScript, PowerScript, LotusScript, JavaScript

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Широкие возможности повторного использования кода также унаследованы сценарными языками от объектно-ориентированных предков.

Существенной положительной чертой языков сценариев является их совместимость с передовыми инструментальными средствами автоматизированного проектирования и быстрой реализации программного обеспечения, или так называемыми CASE- (Computer-Aided Software Engineering) и RAD- (Rapid Application Development) средствами.

Одним из наиболее передовых инструментальных комплексов для быстрой разработки приложений является Microsoft Visual Studio .NET, изучение и использование его возможностей предстоит нам в данном курсе.

Естественно, что вместе с достоинствами объектно-ориентированного подхода языки сценариев унаследовали и ряд недостатков. К последним прежде всего относятся сложность тестирования и верификации программ и возможности возникновения в ходе эксплуатации множественных побочных эффектов, проявляющихся за счет сложной природы взаимодействия объектов и среды, представленной интерфейсами с подчас многочисленными одновременно работающими пользователями программного обеспечения, операционной системой и внешними источниками данных.

Характерные примеры сценарных языков программирования представлены на слайде.

## Языки параллельных вычислений (1)

Время появления: 1980-е г.г.

Краткая характеристика:

программа – совокупность описаний процессов, которые могут выполняться одновременно или псевдопараллельно

Преимущества:

- высокая вычислительная эффективность для больших программных систем (тысячи одновременно работающих пользователей или компьютеров);
- высокая эффективность функционирования в системах реального времени (системы жизнеобеспечения и принятия решений)

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Еще одним весьма важным классом языков программирования являются языки поддержки параллельных вычислений.

Программы, написанные на этих языках, представляют собой совокупность описаний процессов, которые могут выполняться как в действительности одновременно, так и в псевдопараллельном режиме. В последнем случае устройство, обрабатывающее процессы, функционирует в режиме разделения времени, выделяя время на обработку данных, поступающих от процессов, по мере необходимости (а иногда с учетом последовательности или приоритетности выполнения операций).

Языки параллельных вычислений позволяют достичь заметного выигрыша в эффективности при обработке больших массивов информации, поступающих, например, от одновременно работающих пользователей, либо имеющих высокую интенсивность (как, например, видеоинформация или звуковые данные высокого качества).

Другой весьма значимой областью применения языков параллельных вычислений являются системы реального времени, в которых пользователю необходимо получить ответ от системы непосредственно после запроса. Системы такого рода отвечают за жизнеобеспечение и принятие ответственных решений.

## Языки параллельных вычислений (2)

Недостатки:

- высокая себестоимость разработки относительно небольших программ (сотни строк кода);
- относительно узкий спектр применения

Примеры:

Ada, Modula-2, Oz

### Комментарий к слайду

Обратной стороной достоинств рассматриваемого класса языков программирования является высокая стоимость разработки программного обеспечения, следовательно, разработка относительно небольших программ широкого (например, бытового назначения) зачастую нерентабельна.

Примерами языков программирования с поддержкой параллельных вычислений могут служить Ada, Modula-2 и Oz.

## Подходы к программированию

Основные подходы к программированию:

- структурный, модульный;
- функциональный;
- логический;
- объектно-ориентированный;
- смешанный;
- компонентно-ориентированный (.NET);
- чисто объектный

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

Итак, в данной лекции были рассмотрены история и эволюция языков программирования и основные подходы к разработке программных систем. Была сделана попытка классификации языков и подходов к программированию, а также проведен анализ достоинств и недостатков, присущих тем или иным подходам и языкам.

Заметим, что приведенную классификацию не следует считать единственно верной и абсолютной, поскольку языки программирования постоянно развиваются и совершенствуются, и недавние недостатки устраняются с появлением необходимых инструментальных средств или теоретических обоснований.

Подводя итогу, кратко перечислим рассмотренные в течение лекции подходы к программированию:

- ранние неструктурные подходы;
- структурный или модульный подход (задача разбивается на подзадачи, затем на алгоритмы, составляются их структурные схемы и происходит реализация);
- функциональный подход;
- логический подход;
- объектно-ориентированный подход;
- смешанный подход (некоторые подходы возможно комбинировать);
- компонентно-ориентированный (программный проект рассматривается как множество компонент, такой подход принят, в частности, в .NET);
- чисто объектный подход (идеальный с математической точки зрения вариант, который пока не реализован практически).

## Библиография

1. Pratt T.W., Zelkovitz M.V. Programming languages, design and implementation (4<sup>th</sup> ed.).- Prentice Hall, 2000
2. Appleby D., VandeKopple J.J. Programming languages, paradigm and practice (2<sup>nd</sup> ed.).- McGraw-Hill, 1997
3. Peyton Jones S.L. The implementation of functional programming languages.- Prentice Hall, 1987
4. Landin P. The next 700 programming languages. Communications of ACM, 3, 1966
5. Gilmore S. Programming in Standard ML '97: a tutorial introduction. <http://www.dcs.ed.ac.uk/home/stg>

© Учебный Центр безопасности информационных технологий Microsoft  
Московского инженерно-физического института (государственного университета), 2003

### Комментарий к слайду

К сожалению, в рамках одной лекции невозможно в полном объеме представить картину развития языков и подходов к программированию. Мы ограничились рассмотрением лишь наиболее существенных для наших целей подходов, прежде всего, функционального и объектно-ориентированного.

Для более детального ознакомления с последними достижениями и проблемами в области языков программирования, рекомендуется следующий список литературы:

1. Pratt T.W., Zelkovitz M.V. Programming languages, design and implementation (4<sup>th</sup> ed.).- Prentice Hall, 2000
2. Appleby D., VandeKopple J.J. Programming languages, paradigm and practice (2<sup>nd</sup> ed.).- McGraw-Hill, 1997
3. Peyton Jones S.L. The implementation of functional programming languages.- Prentice Hall, 1987
4. Landin P. The next 700 programming languages. Communications of ACM, 3, 1966
5. Gilmore S. Programming in Standard ML '97: a tutorial introduction. <http://www.dcs.ed.ac.uk/home/stg>

Кратко остановимся на источниках. В работе [1] приведен наиболее полный анализ истории развития и особенностей языков программирования с классификацией по областям применения. В работах [2-4] рассмотрены теоретические проблемы и практические аспекты реализации инновационных конструкций в языках программирования. Работа [5] посвящена описанию языка программирования Standard ML (или SML), изучение основ которого планируется в данном семестре.