Лекция № 10

Языки программирования

Вопросы лекции:

1. Классификация языков программирования
2. Системы программирования
3. Синтаксис и семантика
4. Классы языков программирования

Классификация языков программирования

Языки программирования это формальная знаковая система, предназначенная для описания алгоритмов в форме, которая удобна для исполнителя.

Каждый язык программирования предназначен для решения определенного класса задач:

Фортран – старейший язык программирования, предназначен для решения математических задач .

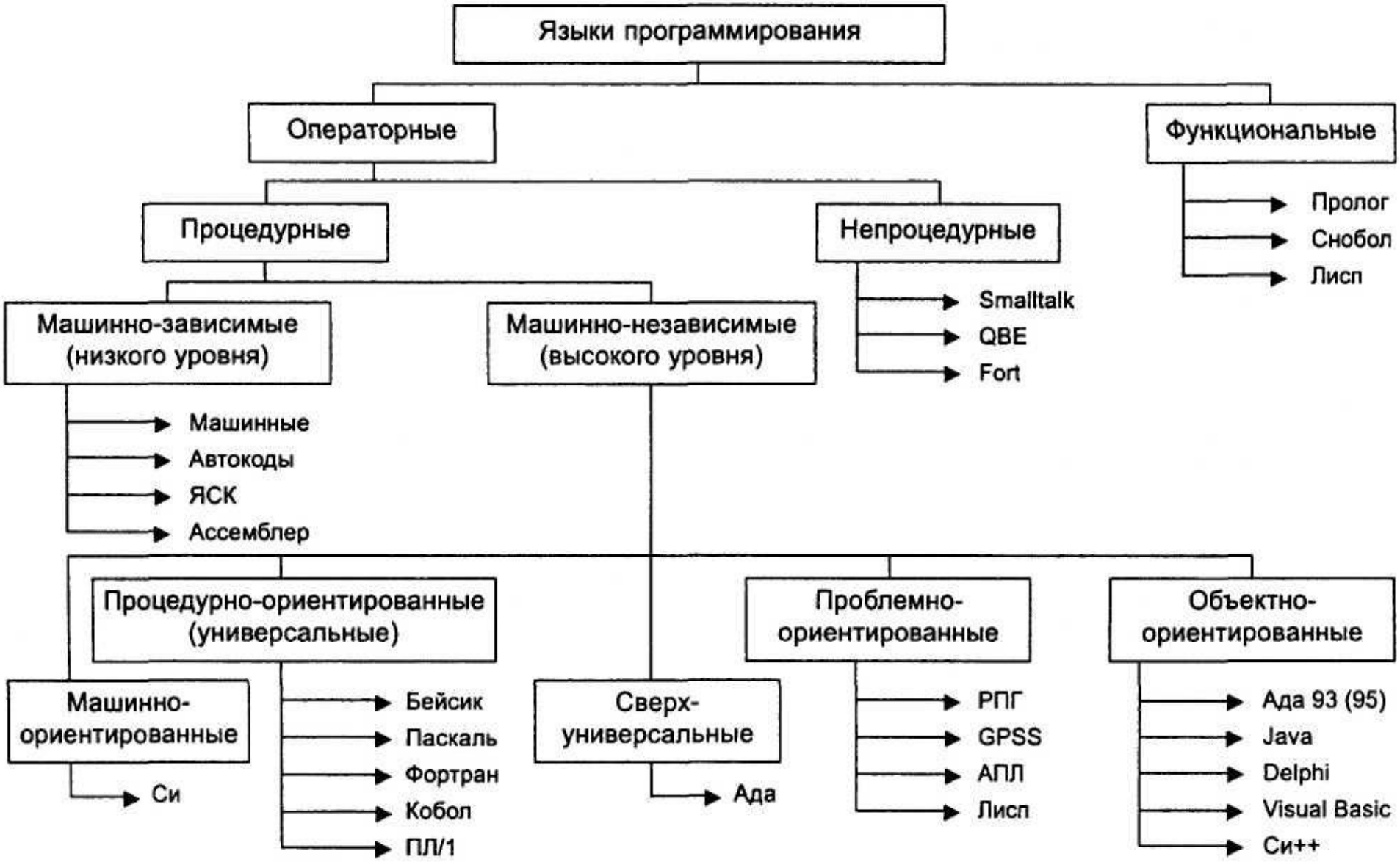
Кобол – для решения экономических задач

Бейсик , Pascal – для обучения

Java (джава) – язык сетевого программирования.

Для системного программирования наиболее подходят языки С, С++ и Ассемблер. С и – язык разработанный для написания операционной системы UNIX (обычно ядро операционных систем писали на Assembler).

Классификация языков программирования



Системы программирования

Системы программирования – это комплексы программ и прочих средств, предназначенных для разработки и их эксплуатации на конкретном языке программирования для конкретного вида ЭВМ.

Система программирования включает:

- **текстовый редактор** (Текст программы, написанный на конкретном алгоритмическом языке);
- **программа-отладчик** (отладка исходного текста программы (поиск и устранение ошибок));
- **транслятор** – программа переводчик с конкретного алгоритмического языка на машинного ориентированный (программа на машинно-ориентированном языке);
- **компоновщик** (редактор связей) - объединяет оттранслированные модули в единые загрузочные, готовые к выполнению;
- программа, обеспечивающая запуск программы;
- библиотека подпрограмм;
- Help (справка, помощь).

Системы программирования

Выделяют два вида трансляторов: **интерпретаторы** и **компиляторы**.

Интерпретатор переводит на язык машинных кодов поочередно каждый оператор исходной программы, проверяет правильность записи оператора и немедленно выполняет его.

Компилятор осуществляет перевод на машинный язык всей исходной программы. Преимуществом компиляторов по сравнению с интерпретаторами является быстродействие, а недостатком – громоздкость. Большинство современных компиляторов работают в режиме трансляции.

В некоторых языках, вместо машинного кода генерируется **интерпретируемый двоичный код " виртуальной машины "**, также называемый **байт-кодом** (byte-code). Такой подход применяется в Forth, Lisp, Java , Perl, Python, а также в языках платформы Microsoft .NET.

Например: Программы на Java выполняются в два этапа. Сначала исходный текст компилятором переводится на промежуточный аппаратно-независимый язык. В таком виде полуфабрикат программы (байт-код) хранится на интернет-сервере, откуда по запросу клиента пересылается ему по сети. У клиента байт-код исполняется специальным интерпретатором, этот интерпретатор называется **виртуальной Java-машиной**, он встроен во все современные браузеры.

Среда визуальной разработки — среда разработки программного обеспечения, в которой наиболее распространённые блоки программного кода представлены в виде графических объектов. Применяются для создания прикладных программ и любительского программирования.

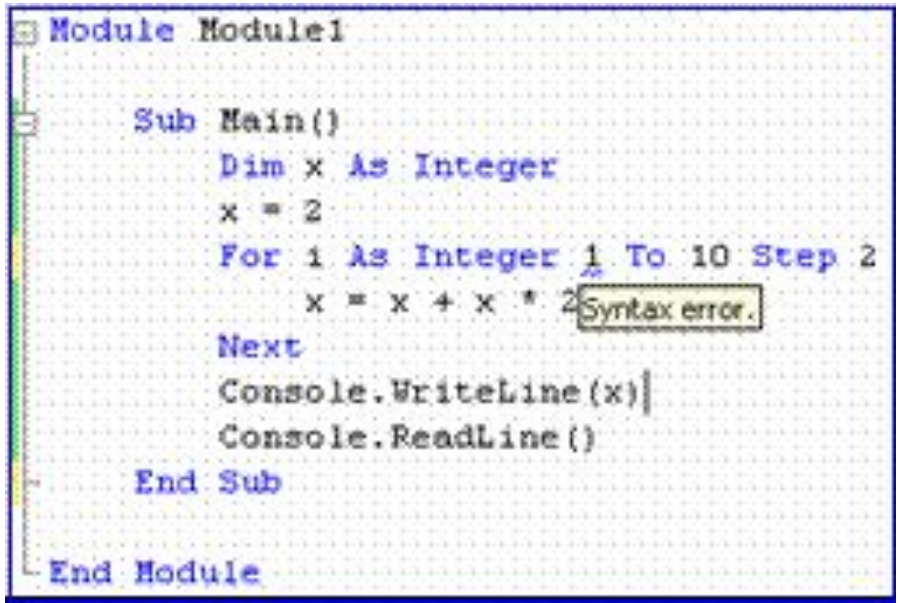
Синтаксис и семантика

Каждый язык программирование обладает своими лексическими, синтаксическими и семантическими правилами, которые необходимо соблюдать при составлении компьютерной программы.

Синтаксис – описывает структуру программ как наборов символов (обычно говорят — безотносительно к содержанию).

Пример синтаксической ошибки : употребление оператора цикла For без To или Next, или отсутствие знака равенства в приведенной на рисунке программе.

Синтаксические ошибки распознаются встроенным синтаксическим анализатором. Синтаксису языка противопоставляется его семантика. Синтаксис языка описывает «чистый» язык, в то же время семантика приписывает значения (действия) различным синтаксическим конструкциям.



Синтаксис и семантика

Семантика – определяет смысловое значение предложений алгоритмического языка.

Пример семантической ошибки :

1) For i As Integer = 1 To 10 Step -2

2) Если надо вычислить , то запись x = a / b * c содержит семантическую ошибку, т.к. приоритет операций деления и умножения одинаков, то вначале a делиться на b , а затем полученный результат умножает на c .

Поиск этих ошибок происходит с помощью логического анализа работы программы и ее тестирования.

```
Module Module1

    Sub Main()
        Dim x, a, b, c As Integer
        x = 2 : a = 5 : b = 3 : c = 4
        For i As Integer = 1 To 10 Step -
            x = a / b * c
            a = a + 3
            b = b - 1
            c = c / 2
        Next
        Console.WriteLine(x)
        Console.ReadLine()
    End Sub

End Module
```

Классы языков программирования

<table>
<tr><th colspan="3">Программирование</th></tr>
<tr><td rowspan="2">Императивное</td><td colspan="2">Декларативное</td></tr>
<tr><td>функциональное</td><td>логическое</td></tr>
</table>

Императивные языки программирования – Бейсик, Паскаль, Си и прочие (включая объектно-ориентированные).

Императивное программирование наиболее популярное. Характеризуются последовательным, пошаговым изменением состояния вычислителя. При этом управление изменениями полностью определено и полностью контролируемо.

Одна из характерных черт императивного программирования – наличие переменных с операцией "разрушающего присвоения". То есть, была переменная A, было у нее значение X. Алгоритм предписывает на очередном шаге присвоить переменной A значение Y. То значение, которое было у A, будет "навсегда забыто".

Если задача описывается последовательным исполнением операций ("открыть кран, набрать воды"), то такие задачи идеальные кандидаты на императивную реализацию.

Классы языков программирования

Декларативные языки́ программирования — это языки программирования высокого уровня, в которых программистом не задается пошаговый алгоритм решения задачи ("как" решить задачу), а некоторым образом описывается, "что" требуется получить в качестве результата. Механизм обработки сопоставления с образцом декларативных утверждений уже реализован в устройстве языка. Типичным примером таких языков являются языки логического программирования (языки, основанные на системе правил).

В программах на языках логического программирования соответствующие действия выполняются только при наличии необходимого разрешающего условия.

Характерной особенностью декларативных языков является их декларативная семантика. Основная концепция декларативной семантики заключается в том, что смысл каждого оператора не зависит от того, как этот оператор используется в программе. Декларативная семантика намного проще семантики императивных языков, что может рассматриваться как преимущество декларативных языков перед императивными.

Наиболее распространённым языком логического программирования является язык Пролог.

Классы языков программирования

Функциональные языки программирования – LISP, ISWIM (If you See What I Mean), ML (Meta Language), Miranda

В языках функционального программирования основными конструктивными элементами являются функции. Тексты программ на функциональных языках программирования описывают «как решить задачу», но не предписывают последовательность действий для решения.

Способ решения задачи описывается при помощи зависимости функций друг от друга (в том числе возможны рекурсивные зависимости) без указания последовательности шагов.

Функциональное программирование, как и другие модели "неимперативного" программирования, обычно применяется для решения задач, которые трудно сформулировать в терминах последовательных операций. Практически все задачи, связанные с искусственным интеллектом, попадают в эту категорию. Среди них следует отметить задачи распознавания образов, общение с пользователем на естественном языке, реализацию экспертных систем, автоматизированное доказательство теорем, символьные вычисления. Эти задачи далеки от традиционного прикладного программирования, поэтому им уделяется не так много внимания в учебных программах по информатике.

Классы языков программирования

Логические языки программирования – Prolog.

Если в функциональном программировании программы - это выражения, и их исполнение заключается в вычислении их значения, то в логическом программировании программа представляет из себя некоторую теорию (описанную на достаточно ограниченном языке), и утверждение, которое нужно доказать. В доказательстве этого утверждения и будет заключаться исполнение программы.

Логическое программирование и язык Пролог появились в результате исследования группы французских ученых под руководством Колмерье в области анализа естественных языков. В последствии было обнаружено, что логическое программирование столь же эффективно в реализации других задач искусственного интеллекта, для чего оно в настоящий момент, главным образом, и используется. Но логическое программирование оказывается удобным и для реализации других сложных задач; например, диспетчерская система лондонского аэропорта Хитроу в настоящий момент переписывается на Прологе. Оказывается, логическое программирование является достаточно выразительным средством для описания сложных систем.

Классы языков программирования

Программирование	
Процедурное	Объектно-ориентированное

Процедурные языки программирования – используют процедуры (подпрограммы, методы или функции). Процедуры содержат последовательность шагов для выполнения. В ходе выполнения программы любая процедура может быть вызвана из любой точки.

Ада, Бейсик, Си, C++, C# (из Microsoft) КОБОЛ, Паскаль, Delphi, Фортран, Java, Перл, Visual Basic, PHP

Объектно-ориентированный подход к программированию - это подход к разработке программного обеспечения, основанный на объектах, а не на процедурах.

Java, Си, Visual Basic

При процедурном программировании программа разбивается на части в соответствии с алгоритмом: каждая часть (подпрограмма, функция, процедура) является составной частью алгоритма. При объектно-ориентированном программировании программа строится как совокупность взаимодействующих объектов.

Классы языков программирования

Объект – это базовое понятие ООП. Любой объект принадлежит одному или нескольким классам, которые в свою очередь определяют, описывают поведение объекта.

Примеры классов: "Гном", "Хоббит", "Маг".

Примеры объектов: "хоббит по имени Фродо Бэггинс", "маг по имени Гэндальф".

Каждый объект характеризуется свойствами, методами и событиями.

Свойства – описание объекта. Примеры атрибутов: "имя", "рост". Набор конкретных значений определяет текущее состояние объекта.

Метод – это действие объекта, изменяющее его состояние или реализующее другое его поведение. Пример методов: "назвать свое имя", "стать невидимым".

Объект, класс, метод, свойства, события – это базовые понятия ООП.

Действие в ООП инициируется посредством передачи сообщений объекту, ответственному за действия. Сообщение содержит запрос на осуществление действия и сопровождается дополнительной информацией (аргументами), необходимой для его выполнения.

Классы языков программирования

К концепции ООП относится:

Полиморфизм – это взаимозаменяемость объектов с одинаковым интерфейсом. Кратко смысл полиморфизма можно выразить фразой: «Один интерфейс, множество методов». В зависимости от типа объекта одно и то же сообщение может соответствовать различным действиям – методам для достижения требуемого результата.

Пример полиморфизма: в ответ на призыв "К оружию! Защищайся!" гном схватит боевой топор, эльф приготовит лук и стрелы, а хоббит спрячется за дерево (у него нет метода, для выполнения требуемых действий). Объекты реагируют на одно и тоже сообщение строго специфичным для них образом.

Наследование - возможность порождать один класс от другого с сохранением всех свойств и методов класса-предка (иногда его называют суперклассом) и добавляя, при необходимости, новые свойства и методы. Наследование призвано отобразить такое свойство реального мира, как иерархичность.

Пример наследования: на основании класса "Личность" создаются его подклассы "Хоббит", "Маг", "Эльф" и "Человек", каждый из которых обладает свойствами и поведением "Личности", но добавляет собственные свойства и меняет поведение.

Классы языков программирования

Инкапсуляция — это принцип, согласно которому любой класс должен рассматриваться как чёрный ящик — пользователь класса должен видеть и использовать только интерфейс (от английского interface — внешнее лицо, т. е. список декларируемых свойств и методов) класса и не вникать в его внутреннюю реализацию.

Этот принцип (теоретически) позволяет минимизировать число связей между классами и, соответственно, упростить независимую реализацию и модификацию классов. Свойство объекта скрывать некоторые свои свойства и методы. Смысл инкапсуляции состоит в том, что внешний пользователь не знает детали реализации объекта, работая с ним путём предоставленного объектом интерфейса.

Классы языков программирования

Программирование	
Неструктурное	Структурное

Неструктурное программирование допускает использование в явном виде команды безусловного перехода (в большинстве языков GOTO). Типичные представители неструктурных языков - ранние версии Бейсика и Фортрана.

Однако в языках высокого уровня наличие команды перехода влечет за собой массу серьезных недостатков: программа превращается в "спагетти" с бесконечными переходами вверх-вниз, ее очень трудно сопровождать и модифицировать.

Фактически неструктурный стиль программирования не позволяет разрабатывать большие проекты. Ранее широко практиковавшееся первоначальное обучение программированию на базе неструктурного языка (обычно Бейсика) приводило к огромным трудностям при переходе на более современные стили.

Классы языков программирования

Структурное программирование:

задача разбивается на большое число мелких подзадач, каждая из которых решается своей процедурой или функцией (декомпозиция задачи). При этом проектирование программы идет по принципу сверху вниз: сначала определяются необходимые для решения программы модули, их входы и выходы, а затем уже эти модули разрабатываются. Такой подход вместе с локальными именами переменных позволяет разрабатывать проект силами большого числа программистов.

Как доказал Э. Дейкстра, любой алгоритм можно реализовать, используя лишь три управляющие конструкции:

- последовательное выполнение,
- ветвление,
- цикл.

И не должно быть безусловных переходов!