Московский государственный университет геодезии и картографии (МИИГАиК)

кафедра информационно-измерительных систем (ИС)

дисциплина: «Языки программирования»

Лекция 2:

«Определение и проблемы языков программирования»

1. Анализ определения понятия «*язык программирования*»
1.1. Проблемы языков программирования
1.2. Определения понятия «язык программирования»
2. Области применения языков программирования
3. Виды языков программирования
3.1. Парадигмы программирования
3.2. Императивные языки программирования
3.3. Объектно-ориентированные языки программирования (ООП).
3.3.1. Парадигма ООП
3.3.2. Язык программирования Delphi
3.2.3. Язык программирования C#
4. Развитие и стандартизация языков программирования

1.1. Проблемы языков программирования

Основная задача XX века (три последних десятилетия: 70ые, 80ые, 90 гг.) – это развитие *аппаратных* компьютерных средств (обусловлено высокой стоимостью обработки и хранения данных и как результат: резкое увеличение производительности компьютера и значительное снижение его стоимости)

Основная задача XXI века:

- это совершенствование качества компьютерных систем, возможности которых целиком определяются программным обеспечением (ПО).

(современный персональный компьютер имеет производительность большой ЭВМ 80ых гг. На сегодняшний сняты практически все аппаратные ограничения на решение задач. Оставшиеся ограничения приходятся на долю ПО)

Актуальные проблемы языков программирования

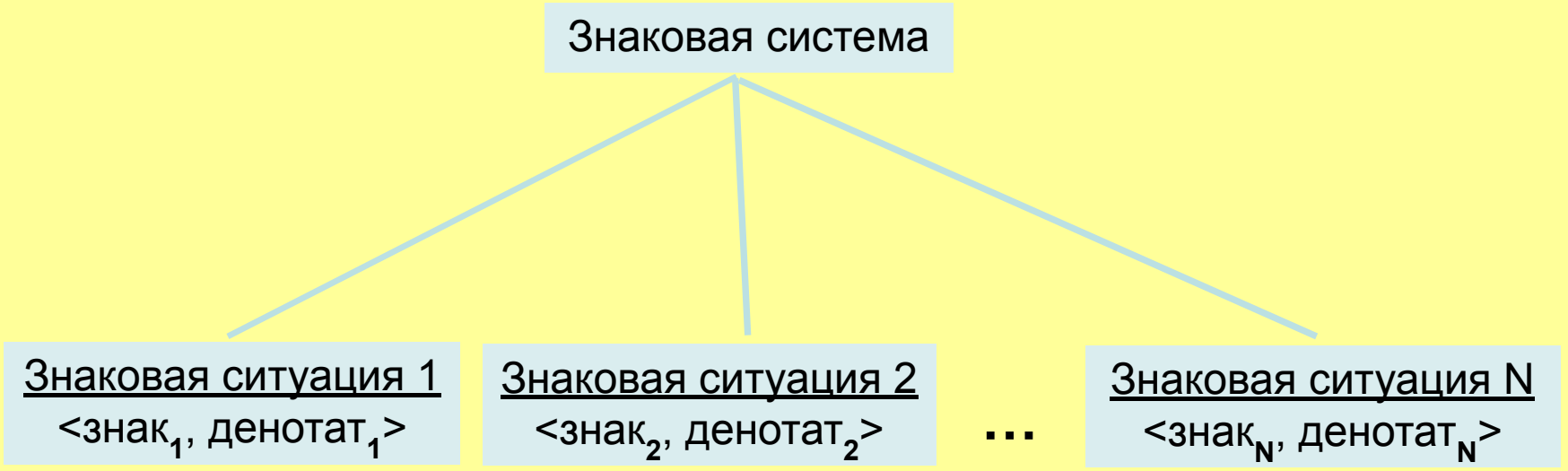
1. **Аппаратная сложность опережает наше умение строить ПО, использующее потенциальные возможности аппаратуры;**
2. **Умение разрабатывать новые программы отстает от требований к новым программам;**
3. **Возможности эксплуатировать существующие программы угрожает низкое качество их разработки**

Ключ к решению этих пробоем – понимание и грамотное использование *языка программирования как основного инструмента для создания ПО*, разумное организация процесса разработки программного приложения.

1.2. Определения понятия «язык программирования»

Исходное определение языка программирования:

Язык программирования (ЯП) – это знаковая система для планирования поведения компьютера.
Знаковая система – совокупность соглашений, определяющих набор знаковых ситуаций:



1.2. Определения понятия «язык программирования»

Знаковая ситуация – первичное понятие семиотики, элементами которой являются *знак* и *денотат*

***Знак* – обозначение или имя.** ***Денотат* – значение, смысл.**

Примеры знаков и денотатов:

Знак:	Денотат:
Сообщение	Смысл
Буква	Звук
Слово	Понятие
Sqrt	Вычисление квадратного корня
If…then…else	Разветвление вычислений

Тогда:

Синтаксис языка – ***это структура допустимых знаков***

Семантика языка – ***это правила, определяющие денотаты, которые соответствуют допустимым знакам***

Модифицированное определение понятия «язык программирования» с т.з. семиотики…

1.2.1. Семиотическое определение понятия «язык программирования»

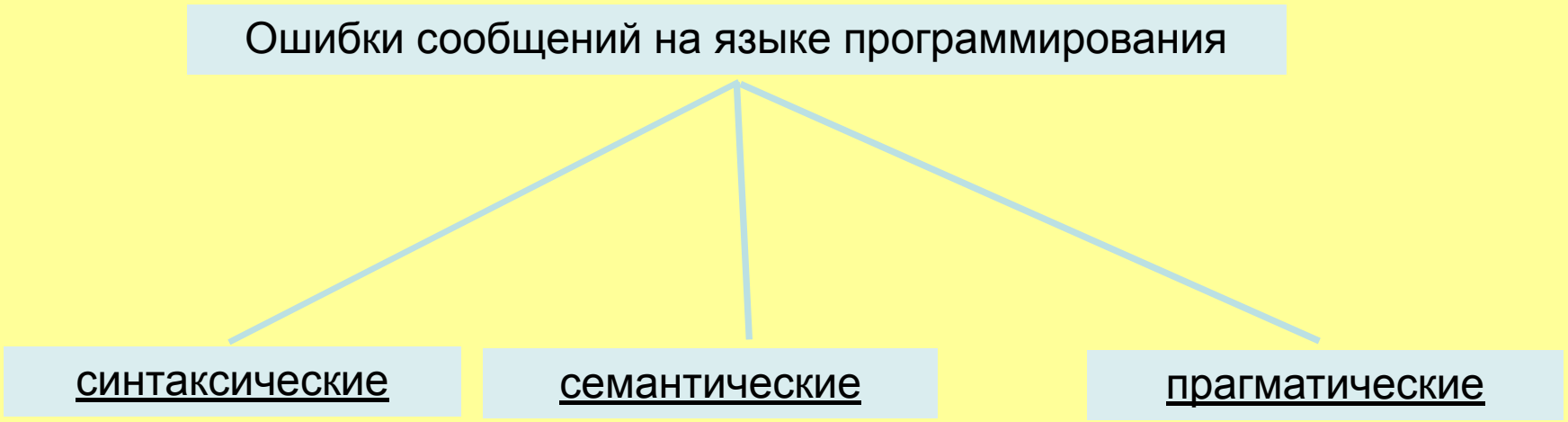
Язык программирования (ЯП) – это правила образования знаков (синтаксис) и согласованные с ними правила образования денотатов (семантика.

В ЯП знаки – это элементы программ; денотаты – характеристики поведения исполнителя (компьютера)

1.2.2. Практическое определение понятия «язык программирования»

Язык программирования (ЯП) – это средство общения между человеком (пользователем) и компьютером (исполнителем).

С помощью ЯП формируются сообщения для компьютера, котрые должны быть понятны компьютеру:



1.2.3. Технологическое определение понятия «язык программирования»

Язык программирования (ЯП) – это инструмент для производства программных услуг.

Основные свойства программ:

- надежность (не содержать ошибок);
- устойчивость (сохранять работоспособность даже в неблагоприятных условиях эксплуатации);
- заботливость (объяснять свои действия и ошибки пользователю)

Сложность программирования:

- семантический разрыв (между уровнем элементарных операций и уровнем потенциально возможных услуг);
- незнание компьютером реального мира (компьютер не может контролировать соответствие указаний программиста поставленной цели и задачам для ее достижения)

Примеры серьезных программных ошибок:

Прочитать самостоятельно – с.26 [С.А.Орлов. Теория и практика языков программирования]

1.2.3. Технологическое определение понятия «язык программирования»

Один из примеров серьезных программных ошибок:

Космический аппарат «Маринер-2» (1962 г.). Цель – Венера. Источник ошибки – пробел и пропуск запятой в операторе цикла DO на ЯП Фортран (программа управления)

***Подробнее* – с.26 [С.А.Орлов. Теория и практика ЯП]**

Средство борьбы с семантическим разрывом* – *использование в программах аппарата абстракции-конкретизации языка, который является основой ориентации на проблему. Пример: использование подпрограмм, состоящей из формальных параметров, а обращение к ней – с фактическими параметрами.

Средство борьбы с незнанием реального мира* – *использование в программах аппарата прогноза-контроля ЯП. Пример: в ЯП как средство прогноза используют встроенные типы данных; контроль предусматривается семантикой языка. Однако средств управления таким контролем на сегодняшний день нет.

В н.в. развиваются – языки искусственного интеллекта, цель которых – предоставление знаний о мире, области знаний и контроль действий - как программы, так и пользователя.

Вывод:

***Технологический критерий качества ЯП* – язык тем лучше, чем проще производство на его основе программных услуг**

2. Области применения языков программирования

Главный критерий выбора ЯП – принадлежность задачи к конкретной предметной области знаний, наиболее представительные из которых:

1. **Научные вычисления**
2. **Обработка деловой информации**
3. **Искусственный интеллект**
4. **Системная область (развитие операционных систем для компьютеров, системное ПО)**
5. **Web – обработка**

(***Докладчик*** **–** *см. Орлов, стр.27-30)*

Критерии эффективности ЯП

1. **Читабельность** *(легкость понимания текста программ)*
2. **Легкость создания программ** *(удобство языка для создания программ в выбранной предметной области)*
3. **Надежность** *(минимум ошибок при работе с программой)*
4. **Стоимость** *(всего жизненного цикла программ - выполнения, трансляции, создания и тестирования, сопровождения)*
5. **Переносимость программ** *(на разные платформы - ОС)*
6. **Универсальность** *(применимость к широкому кругу задач)*
7. **Четкость** *(полнота и точность официального описания языка)*

(***Докладчик*** **–** *см. Орлов, стр.31-40)*

3. 1. Парадигмы программирования

Парадигма – это главная идея какого-либо сложного понятия, в частности «язык программирования»

Применительно к «языкам программирования» различают парадигмы:

- **Императивное программирование**
- **Функциональное программирование**
- **Логическое программирование**
- **Объектно-ориентированное программирование**

Задание для компьютера формируется:

- ✔ **в императивном программировании - в виде последовательности команд;**
- ✔ **в объектно-ориентированном программировании – в виде объектов.**
- ✔ **в функциональном программировании - в виде указания функций;**
- ✔ **в логическом программировании – в виде т.н. высказываний;**

***Каждую* парадигму поддерживает *свой* язык программирования:**

- **Императивные ЯП – ориентированы на последовательность действий, производимая операторами;**
- **Объектно-ориентированные ЯП – вычисления реализуются совокупностью объектов;**
- **Функциональные ЯП - вычисления реализуются как вызовы функций;**
- **Логические ЯП - вычисления реализуются с помощью формальной логики**

Прочитать самостоятельно *– гл.3 [Орлов С.А. Теория и практика ЯП]*

3. 2. Императивные языки программирования

Императивные языки программирования – языки, задающие вычисления как последовательность команд (операторов)

Основные понятия императивных ЯП тесно связаны с компонентами компьютера:

- **Переменные различных типов (моделируют ячейки памяти)**
- **Операторы присваивания (моделируют пересылки данных)**
- **Повторение действий в форме итерации (моделируют хранение информации в смежных ячейках памяти)**

Как работает оператор присваивания:

- ***операнды* выражения (*правая часть оператора*) передаются из оперативной памяти в процессор, результат вычисления выражения заносится в ячейку памяти, называемую *левой частью оператора*.**

Вычисления основываются на понятии «состояние компьютера».

Состояние компьютера – это множество всех значений его памяти.

Программа состоит из последовательности операторов, выполнение каждого из которых влечет за собой изменение значения в одной или нескольких ячейках памяти.

Синтаксис программы на императивном ЯП в общем виде:

```
оператор1;
оператор2;
...
```

3. 3. Объектно-ориентированные языки программирования (ООП)

3. 3.1. Парадигма ООП

Парадигма ООП – это развитие императивного программирования («впитали» в себя лучшие понятия и механизмы императивных ЯП)

Цели:

- **сократить размеры программ за счет повышения размера строительных элементов («маленькие» переменные заменяются «большими» объектами) и т.о. обеспечить возможность создания (за то же время!) более крупных программных приложений;**
- **упростить процесс создания новых программ на базе старых (за счет применения механизма наследования)**

Объектно-ориентированные ЯП задают вычисления как взаимодействие программных объектов

Объект – это именуемый модуль, заключающий в себе данные и операции для их обработки.

Программный объект во многом похож на физический объект реального мира: имеет *свое состояние* и демонстрирует свое *поведение*.

***Состояние* объекта характеризуется перечнем данных и их значений.**

***Поведение* задается последовательностью выполняемых операций.**

3. 3.1. Парадигма ООП

Объекты *взаимодействуют* друг с другом с помощью сообщений. Сообщение посылается объектом-источником в адрес объекта-приемника. Каждое сообщение – это запрос на выполнение операции объектом-приемником.

Класс – описание объектов с общей структурой и поведением. Как и переменные, единичные объекты создаются по их описанию. Но в роли описаний:

- **для *переменных* выступают типы данных;**
- **для *объектов* – классы.**

Объект – это экземпляр класса!

Особенности объектно-ориентированных ЯП основаны на *3х принципах*:

- **инкапсуляция (сокрытие своего содержимого от внешнего мира);**
- **наследование (возможность получения потомками структуры и поведения предков);**
- **полиморфизм (использование одного и того же имени для выражения различных действий и объектов)**

3. 3. 1. Парадигма ООП

Инкапсуляция – каждый объект помещен в защитную оболочку , сквозь которую другие объекты видят лишь самое необходимое: заголовки операций, которые выполняет объект.

Наследование – внедрение в новый класс элементов данных и операций старого класса, обеспечивая возможность их модификации.

Полимофизм – поддержка возможности существования целого семейства различных операций с одинаковым именем.

Первый ООП ЯП – Simula, 1967 г. – норвежцы Нигаард К. и Дал У. (в основе императивный ЯП – Algol 60). Этот язык опередил свое время и был забыт.

Второй ООП ЯП – Smalltalk, 1972-1980 гг – фирма Xerox: автор Алан Кей

Гибридные языки, реализующие сразу несколько парадигм: Ada 2005, Object Pascal – императивная и объектно-ориентированная парадигмы

3.3.2. Язык программирования Delphi в примерах (3 принципа ООП) [докладчик!]

3.3.3. Язык программирования C# в примерах (3 принципа ООП [докладчик!]

4. Стандартизация языков программирования

Необходимость стандартов ЯП – компиляция одной и той же программы различными компиляторами должна давать одинаковый результат

Вопросами стандартизации занимается 3 основные организации:

1. **Американский национальный институт стандартов – ANSI (American National Standards Institute)**
2. **Институт инженеров по электротехнике и электронике – IEEE (Institute of Electrical and Electronic Engineers)**
3. **Организация международных стандартов – ISO (International Standards Organization), где есть комитет JTC – занимается специально вопросами стандартизации ЯП**

Примеры стандартизации –
***Прочитать самостоятельно* – с.5-10 [Баженова И.Ю. Языки программирования]**