Лекция 1. Парадигмы программирования

План:

1. **Системы программирования**
2. **Языки программирования: их уровни и поколения**

https://youtu.be/D2RmmRA5myg

Что такое программирование?

https://youtu.be/Ih4cpq5qDl0

Кто такой программист | Самая лучшая профессия

Парадигма программирования - это совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию).

Это способ концептуализации, определяющий организацию вычислений и структурирование работы, выполняемой компьютером.

Вопрос 1. СИСТЕМЫ ПРОГРАММИРОВАНИЯ

В парадигмах программирования существует 2 высказывания о системах программирования:

1. Система программирования – это комплекс инструментальных программных средств, предназначенный для работы с программами на одном из языков программирования.
2. Система программирования - это комплекс программ и файлов, позволяющий выполнить полный набор операций, связанных с изготовлением программы и работой с ней.

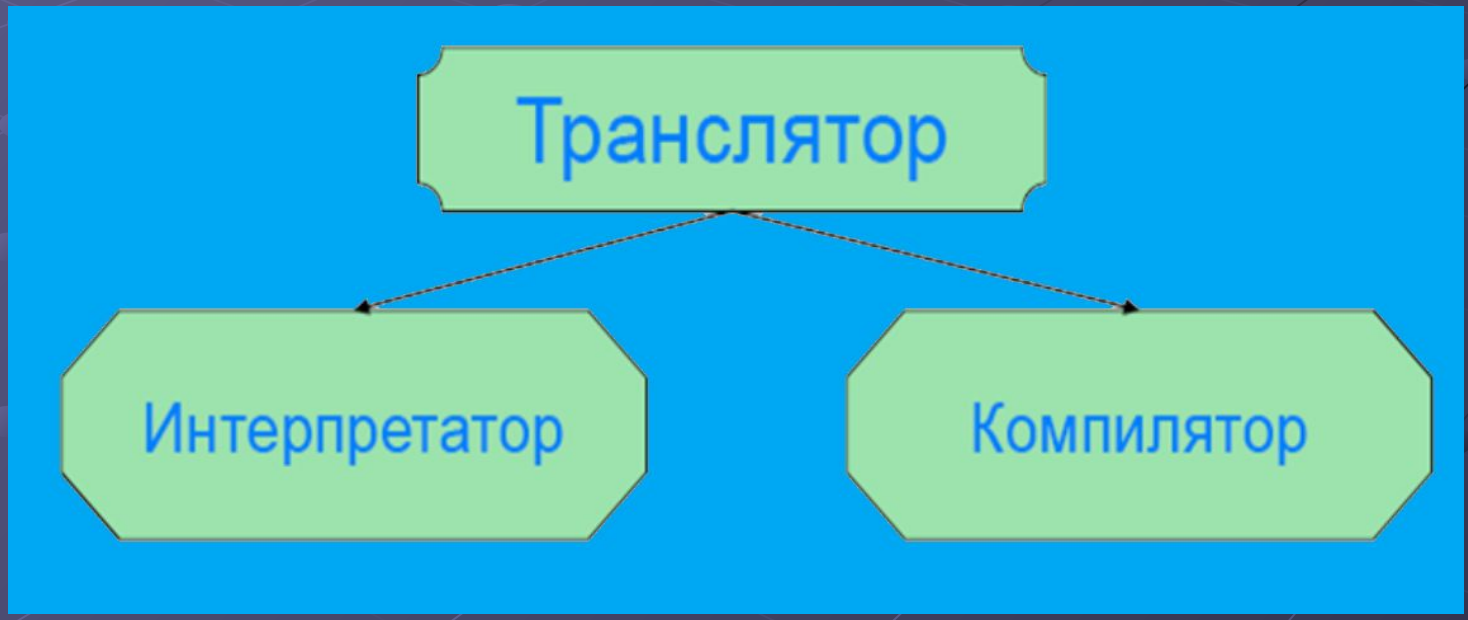
Система программирования включает в свой состав:

- транслятор (компилятор или интерпретатор);
- интегрированная среда разработки;
- средства создания и редактирования текстов программ;
- библиотеки стандартных подпрограмм и функций;
- отладочные программы;
- «дружественная" к пользователю диалоговая среда;
- многооконный режим работы;
- мощные графические библиотеки;
- утилиты для работы с библиотеками;
- встроенный ассемблер;
- встроенная справочная служба;
- другие специфические особенности.

Транслятор (англ. «translator» переводчик) - это программа-переводчик.

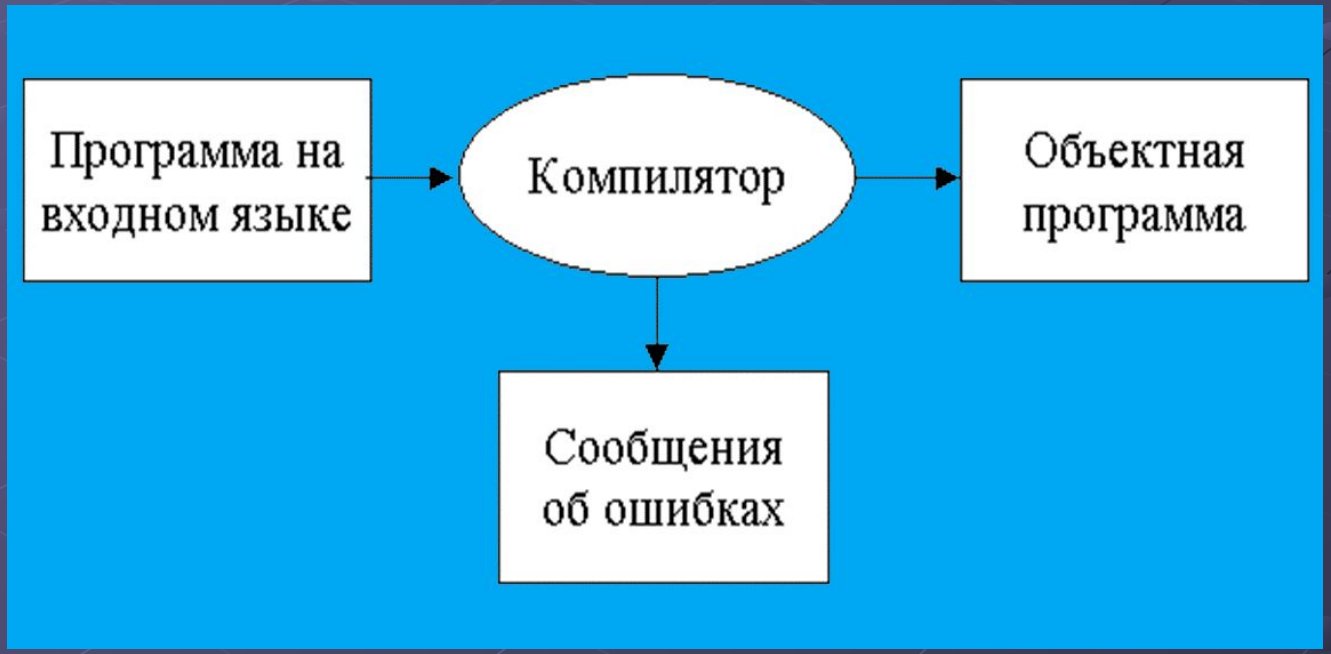
Она преобразует программу, написанную на одном из языков высокого уровня, в программу, состоящую из машинных команд.

Трансляторы реализуются в виде компиляторов или интерпретаторов.



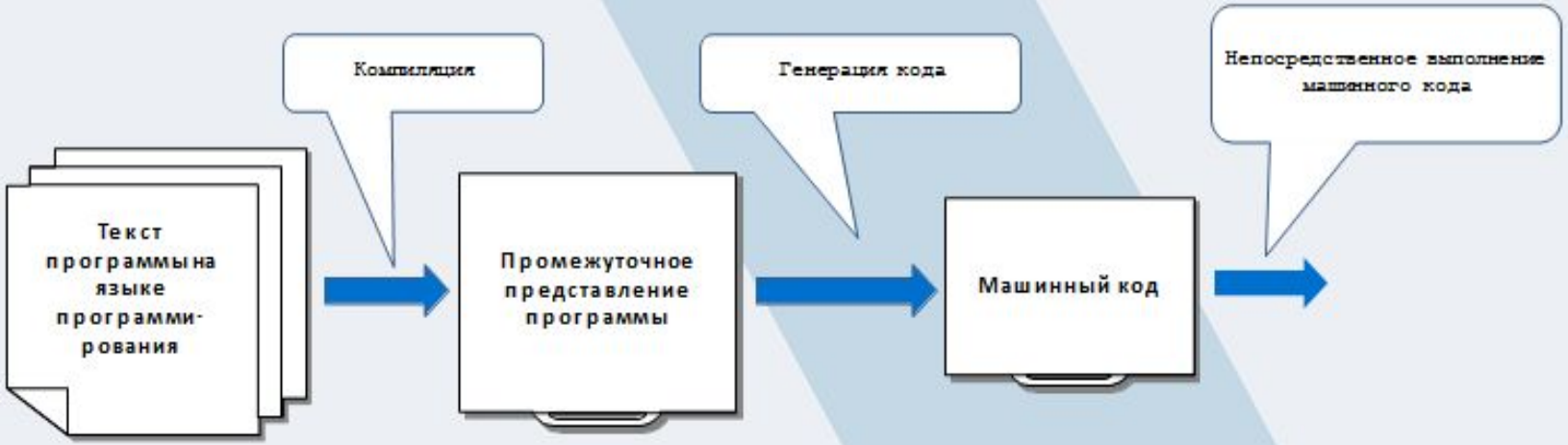
Компилятор (англ. «compiler» - составитель, собиратель) читает всю программу целиком, делает ее перевод и создает законченный вариант программы на машинном языке, который затем и выполняется.

Откомпилированные программы работают быстрее.



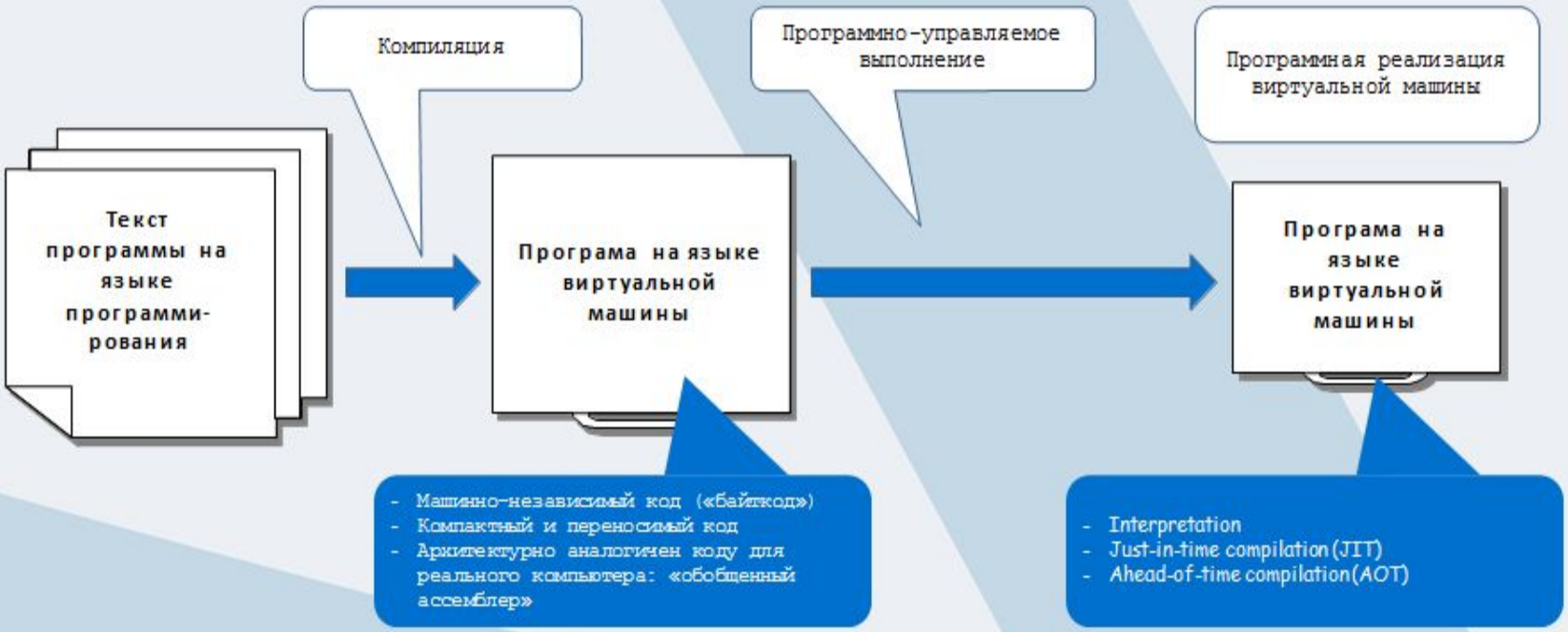
Компиляция и выполнение: традиционная модель

C/C++, Ada, Eiffel



Компиляция и выполнение: модель Java

Java, C#, Python



Интерпретатор (англ. «interpreter» - истолкователь, устный переводчик) переводит и выполняет программу строка за строкой.

После того, как программа откомпилирована, исходная программа и компилятор больше не нужны. В то же время программа, обрабатываемая интерпретатором, должна заново переводиться на машинный язык при каждом очередном запуске программы.

Интерпретируемые программы проще исправлять и изменять.

Компилируемые языки программирования – язык, исходный код которого преобразуется компилятором в машинный код и записывается в файл, с особым заголовком и/или расширением, для последующей идентификации этого файла операционной системой, формирующей ***окончательный исполняемый/выполнимый код*** для непосредственного выполнения центральным процессом компьютера.

Примеры: C, C++, Pascal, Object Pascal, Delphi, FORTRAN, Ada и мн.др.

Интерпретируемые языки программирования – язык, в котором исходный код программы не преобразуется в машинный код для непосредственного выполнения центральным процессором (как в компилируемых ЯП), а выполняется/исполняется с помощью специальной программы-интерпретатора, т.е. это язык, реализующий ***интерпретируемый (управляемый) код*** .

Примеры: Java, LISP, Perl, PROLOG, C# и мн. др.

Особенность среды программирования Visual Studio:
позволяет создавать на C++, как традиционные приложения с ***выполнимым кодом***, так и приложения ***интерпретируемого (управляемого) кода***.

Вывод: Язык программирования C# является и компилируемым, и интерпретируемым.

Интегрированная среда разработки (IDE)

– это система программных средств, используемая программистами для разработки программного обеспечения

Среда **IDE** включает в себя:

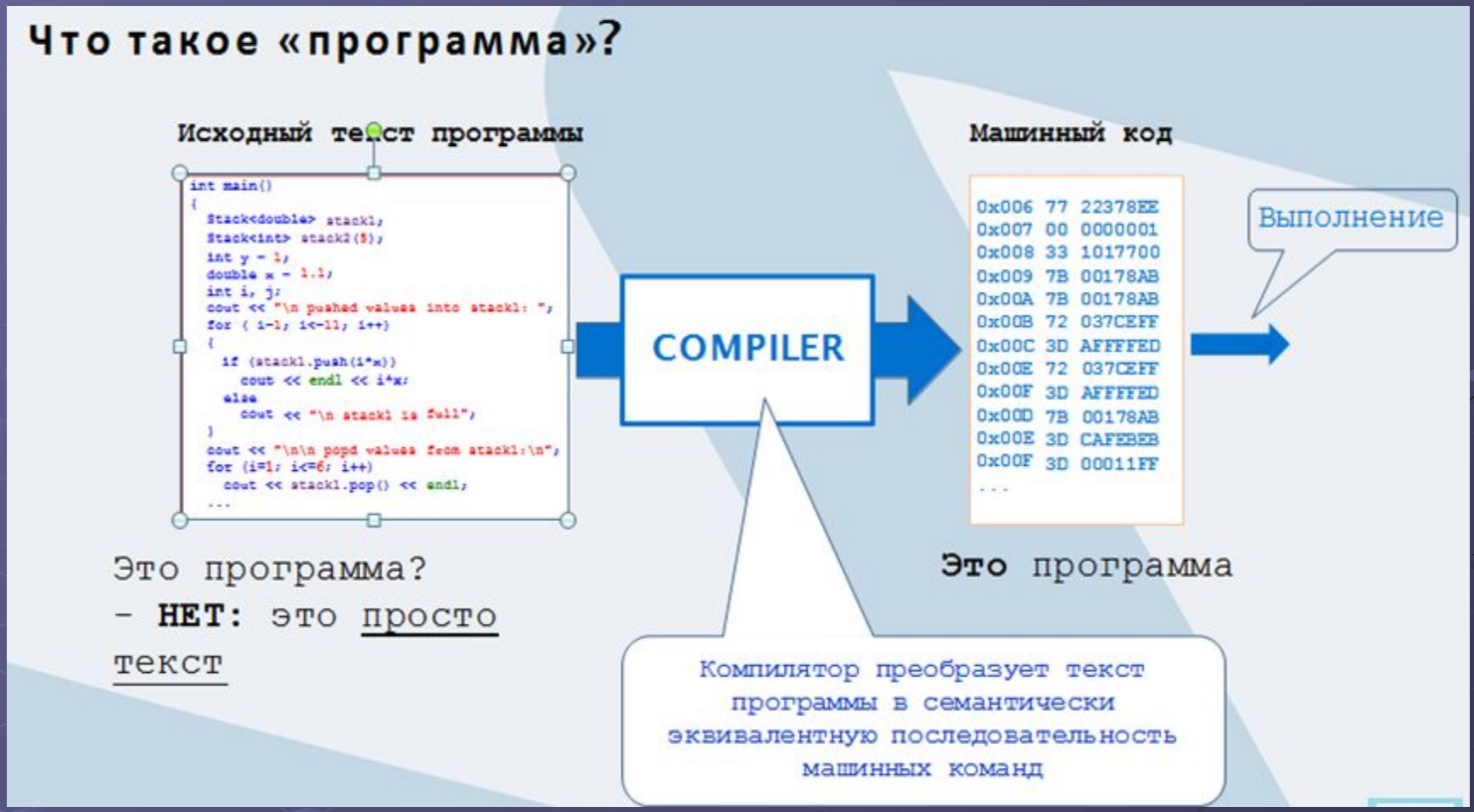
- текстовый редактор;
- компилятор и/или интерпретатор;
- средства автоматизации сборки;
- отладчик.

Язык	IDE
Си/C++	Borland C++, C++ Builder ...
Бэйсик	Turbo Basic, Visual Basic ...
Паскаль	Delphi, Turbo Pascal ...
ActionScript	Adobe Flash, Adobe Flash Builder ...
Универсальные IDE	Visual Studio, Komodo, Eclipse

Отладочные программы (отладчики) – специальные средства, позволяющие исследовать внутреннее поведение программы

Возможности:

- пошаговое исполнение программы с остановкой после каждой команды (оператора);
- просмотр текущего значения любой переменной или нахождение значения любого выражения;
- установка в программе «контрольных точек», т.е. пошаговое выполнение программы.



Программа – это алгоритм, записанный на понятном компьютеру языке программирования

Общая модель памяти

Концептуальный взгляд

Программа использует три вида памяти:
- Программный код
- Динамическая апмять ("Heap")
- Стек выполнения

Программа

Последовательность машинных команд

Программа **не может модифицировать** код: самомодифицирующие программы не допускаются

Heap

Динамически размещаемые объекты

Дисциплина использования дин. памяти определяется **динамической семантикой** программы, т.е. во время ее выполнения

Стек

Local objects

Дисциплина использования стека определяется (статической) **структурой** программы

Вопрос 2. Языки программирования: их уровни и поколения

Языки программирования - это формальные языки, кодирующие алгоритмы в привычном для человека виде

(в виде предложений), т.е. специально созданная система слов, букв и чисел

О чём и зачем этот курс?

- Языки программирования – основной инструмент создания программ.
- «Чтобы научиться программировать, надо программировать».
- Чтобы программировать, надо понимать не только правила составления программ, но прежде всего смысл языковых конструкций.

«Средства, которыми мы пользуемся, оказывают глубокое и тонкое влияние на наши способы мышления и, следовательно, на нашу способность мыслить»

Эдсгер Дейкстра
о языках программирования

Сколько языков следует знать?

- Нет «наилучшего» языка программирования. Любой язык хорош для своей области применения и, возможно, не подходит для некоторой другой.

-Языки сильно отличаются, однако…

-Некоторые общие принципы часто аналогичны!

Не изучать язык – изучать принципы! Понимая принципы, любой язык можно изучить за пару недель.

- Однако, чем больше языков специалист знает, тем лучше. Профессиональный программист должен хорошо знать несколько языков программирования.

Синтаксис или семантика?

Что значит «выучить язык»?

Синтаксис:

Правила, задающие структуру программ и их частей (конструкций)

Семантика: Смысл конструкций языка

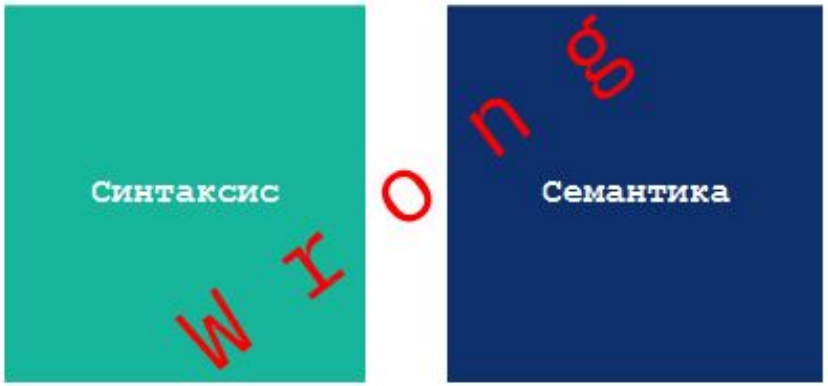
Статическая семантика:

- Как программы компилируются

Динамическая семантика:

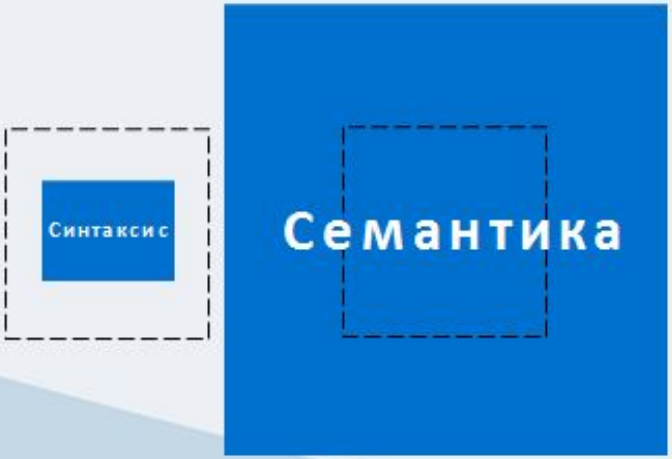
- Как программы выполняются

«Обычный» взгляд на ЯП



Синтаксис или семантика?

Реальность:



Выводы для программистов

- Выучить синтаксис нового ЯП можно за неделю
- Прежде всего следует изучать семантику языка – это гораздо более важно и существенно труднее

Синтаксис или семантика?

```
void f(int p)
{
    int a, b;
    *a = 777;
    return xyz*a+b*f;
}
// ...
f();
int x = f(1,2);
```

Illegal operator (dereferencing) for an object of type int

Use of uninitialized variables a and b

Undefined variable xyz

Illegal operand types for the * operator

Function f cannot return a value (void type)

Illegal number of arguments in calls to f function

Illegal position of the call to f

Синтаксически совершенная программа, но семантически некорректная

Алгоритмическим языком

называется совокупность символов и правил, позволяющая описывать алгоритм и однозначно истолковывать содержание его описания

Языки программирования делятся на:

1. ***Машинно-зависимые или машинноориентированные языки (низкого уровня):*** Автокоды, Ассемблеры – позволяют управлять вычислительным процессом напрямую, при помощи машинных команд
2. **Машинно-независимые *(высокого уровня)***

Машинно-зависимые языки
(низкого уровня)

Основные конструктивные средства таких языков позволяют учитывать особенности архитектуры и принципы работы определенной ЭВМ, т.е. ориентированы на конкретный тип процессора

Достоинства: программы компактны и работают эффективно

Применяют: для написания небольших системных приложений, драйверов устройств, компьютерных вирусов

Машинный язык (40-50 годы XXв.)

Программы на машинном языке имеют длинные последовательности единиц и нулей, являлись машинно-зависимыми (т.е. для каждой ЭВМ необходимо было составлять свою программу)

Ассемблер (начало 50-х годов XXв.)

Ассемблер обеспечивает возможность применения символических имен в исходной программе и избавляет программиста от утомительного труда (неизбежного при программировании на языке машинных команд) по распределению памяти компьютера для команд, переменных и констант.

```
0400 2073FE JSR $FE73          s~
0403 A200   LDX #$0            "□
0405 BD8004 LDA $480,X         =□↖
0408 F006   BEQ $410           p√
040A 2075FE JSR $FE75           u~
040D E8     INX                h
040E D0F5   BNE $405           Pu
0410 00     BRK                □
0411 B9     *=$480
0480 48     'H                 H
0481 45     'E                 E
0482 4C     'L                 L
0483 4C     'L                 L
0484 4F     'O                 O
0485 00     #0                 □
0486 67     !
```

Машинно-независимые языки (высокого уровня)

Значительно ближе и понятнее человеку, нежели компьютеру. Здесь особенности компьютерных архитектур не учитываются, поэтому создаваемые программы легко переносятся на другие платформы, для которых создан транслятор этого языка

Классификация языков программирования высокого уровня

Процедурное программирование - представляет собой последовательность команд, определяющих алгоритм решения задачи.

Основная идея - использование памяти для хранения данных.

Основная команда - присвоение, с помощью которой определяется и меняется память компьютера.

Программа производит преобразование содержимого памяти, изменяя его от исходного состояния к результирующему.

Языки процедурного программирования

Операционные языки (привязанные к конкретной архитектуре ЭВМ):

Фортран (начало 50-х г.) - первый компилируемый язык для программирования научно-технических задач;
Кобол (конец 60-х г.) - для решения задач обработки больших объемов данных, хранящихся на различных носителях данных;
Алгол (1960 г.) – многоцелевой расширенный язык, в котором впервые введены понятия «блочная структура программы» и «динамическое распределение памяти»;
BASIC (середина 60-х г.) - характеризуется простотой освоения и наличием универсальных средств для решения научных, технических, экономических и игровых задач.

Структурные языки (не привязанные к конкретной архитектуре ЭВМ):

PL-1 (1963-1966гг.) - хорошо приспособлен для исследования и планирования вычислительных процессов, моделирования, решения логических задач, разработки систем математического обеспечения.
PASCAL (1968-1971гг.) - популярный для ПК, в основу которого положен подход от общей задачи к частным (более простым и меньшим по объему).
АДА (1979 г) - назван в честь первой программистки Ады Лавлейс. Его отличает модульность конструкций.
СИ (начало 70-х г.) - первоначальный его вариант планировался как язык для реализации операционной системы Unix вместо языка Ассемблера. Особенность языка - это то, что различия между выражениями и операторами сглаживаются, что приближает его к функциональным языкам программирования.
Модула (1980 г.) - для профессиональных системных программистов.

Объектно-ориентированное программирование (ООП)

- это метод программирования, при использовании которого главными элементами программ являются объекты.

Объединение данных и свойственных им процедур обработки в одном объекте, называется инкапсуляцией и является одним из важнейших принципов ООП.

Объектно-ориентированные языки

позволяют разрабатывать программные приложения для большого круга задач, имеющих общность в реализуемых компонентах

Объектные языки:
C++ , Java.

Визуальные языки (середина 90-х г.) - интерфейсная часть программного продукта создается в диалоговом режиме, практически без написания программных операторов: Visual Basic, Delphi, C++ Builder, Visual C++, Object PAL, dBase.

Язык VBA (Visual Basic for Application) – язык приложений Microsoft Office (Excel, Word, Power Point и др), который соблюдает основной синтаксис языка и правила программирования языков Basic – диалектов, что позволяет создавать макросы для автоматизации выполнения операций и графический интерфейс пользователя, интеграцию между различными программными продуктами.

Языки программирования для компьютерных сетей являются интерпретируемыми. Интерпретаторы для них распространяются бесплатно, а сами программы – в исходных текстах. Такие языки называются **скрипт – языками**.

- **Perl** (1987 г.) язык для обработки больших текстов и файлов и расшифровывается, как язык для практического извлечения данных и составления отчетов. С помощью него можно создать скрипт, который открывает один или несколько файлов, обрабатывает информацию и записывает результаты.

- **PHP** (1995-1997гг.) обладает средствами доступа к БД и используется создателями динамических сайтов во всем мире.

- **Tcl/Tk** (конец 80-х г.) состоит из мощных команд, предназначенных для работы с абстрактными нетипизированными объектами и позволяет создавать программы с графическим интерфейсом.

- **VRML** (1994 г.) создан для организации виртуальных трехмерных интерфейсов в Интернете.

- **XML** - с 1996 г. идет работа над созданием универсального языка структуры документов. Может стать заменой языка HTML.

Декларативные языки программирования – это функциональные и логические языки программирования

Функциональное программирование - это способ составления программ, в которых единственным действием является вызов функции. Программа, написанная на функциональном языке, представляет собой последовательность описания функций и выражений. Выражение вычисляется сведением сложного к простому. Все выражения записываются в виде списков.

Логическое программирование - это программирование в терминах логики.

Языки декларативного программирования

Язык функционального программирования:

Лисп (1959 г) - позволяет обрабатывать большие объемы текстовой информации.

Язык логического программирования:

Пролог (1973 г.) - язык искусственного интеллекта. Программа на языке Пролог строится из последовательности фактов и правил, затем формулируется утверждение, которое Пролог пытается доказать с помощью правил. Язык сам ищет решение с помощью методов поиска и сопоставления, которые в нем заложены.

«Язык формирует наш способ мышления и определяет, о чем мы можем мыслить»

Б.Л.Ворф

Введение в программирование

Сложно ли научиться программировать?

И да, и нет!

ЗАДАНИЕ

Предположим, вы хотите поручить роботу-исполнителю приготовить яичницу из трех яиц. Составьте для него план решения этой задачи.

Когда сделаете, нажмите кнопку «Готово»

Готово

Что у вас получилось?

Обычно предлагают такой план:

1. Зажечь газ
2. Поставить на плиту сковородку
3. Положить кусочек масла
4. Разбить яйца
5. Посолить по вкусу
6. Через 3-4 минуты выключить газ. Яичница готова.

Вы тоже так думаете?

Если да, то вы *психологически* еще не готовы к серьезному изучению программирования.

Может быть лучше попробовать себя в другой области?

Казалось бы ничего сложного. Когда мы жарим яичницу, то почти не задумываемся о своих действиях. Но не забывайте, что мы поручили это задание роботу-исполнителю. А поскольку машина не обладает человеческим интеллектом, то нет никакой гарантии, что **конечный результат** будет достигнут.

ПОЧЕМУ?

Робот-исполнитель может выполнять только конкретные и однозначно понимаемые команды. Даже такая простая команда как «Взять кусочек масла» может поставить его в тупик, он «зависнет».

Чем зажечь газ?

Сколько взять масла?

Какого: сливочного, топленого?

Сколько разбить яиц?

А **что значит** «посолить по вкусу»?

Так **сколько же**: 3 или 4 минуты жарить яичницу?

Допустим, каждую команду мы сделали предельно конкретной.

Но и этого недостаточно!

А если спичек, масла, соли и т.д. не окажется?

А яйца разбивать на сковородку вместе со скорлупой?

И так далее, и так далее.

Допустим, мы и это предусмотрели.

У-ф-ф! Ну теперь, кажется, все. Посмотрим, что у нас получилось.

Фу-у-у! Яйцо оказалось тухлым!

Хороший программист не тот, кто хорошо знает язык, а тот, кто умеет прогнозировать!

Задание для самостоятельной работы

Подготовить отчет:

1. Составить краткий конспект в формате MS Word по темам «Что такое программирование?» и «Кто такой программист. Самая лучшая профессия» на основе видео-лекций по ссылкам:

https://youtu.be/D2RmmRA5myg

https://youtu.be/Ih4cpq5qDI0

2. Составить не менее 40 вопросов по лекции-презентации «Парадигмы программирования» и на каждый вопрос дать краткий ответ в формате MS Word.

Готовый файл-отчет прикрепить в личный кабинет студента