Скриптовые языки программирования

Доклад на семинаре по специальности

Студент гр. 4057/2 Руцкий Владимир

27.10.2009

Содержание

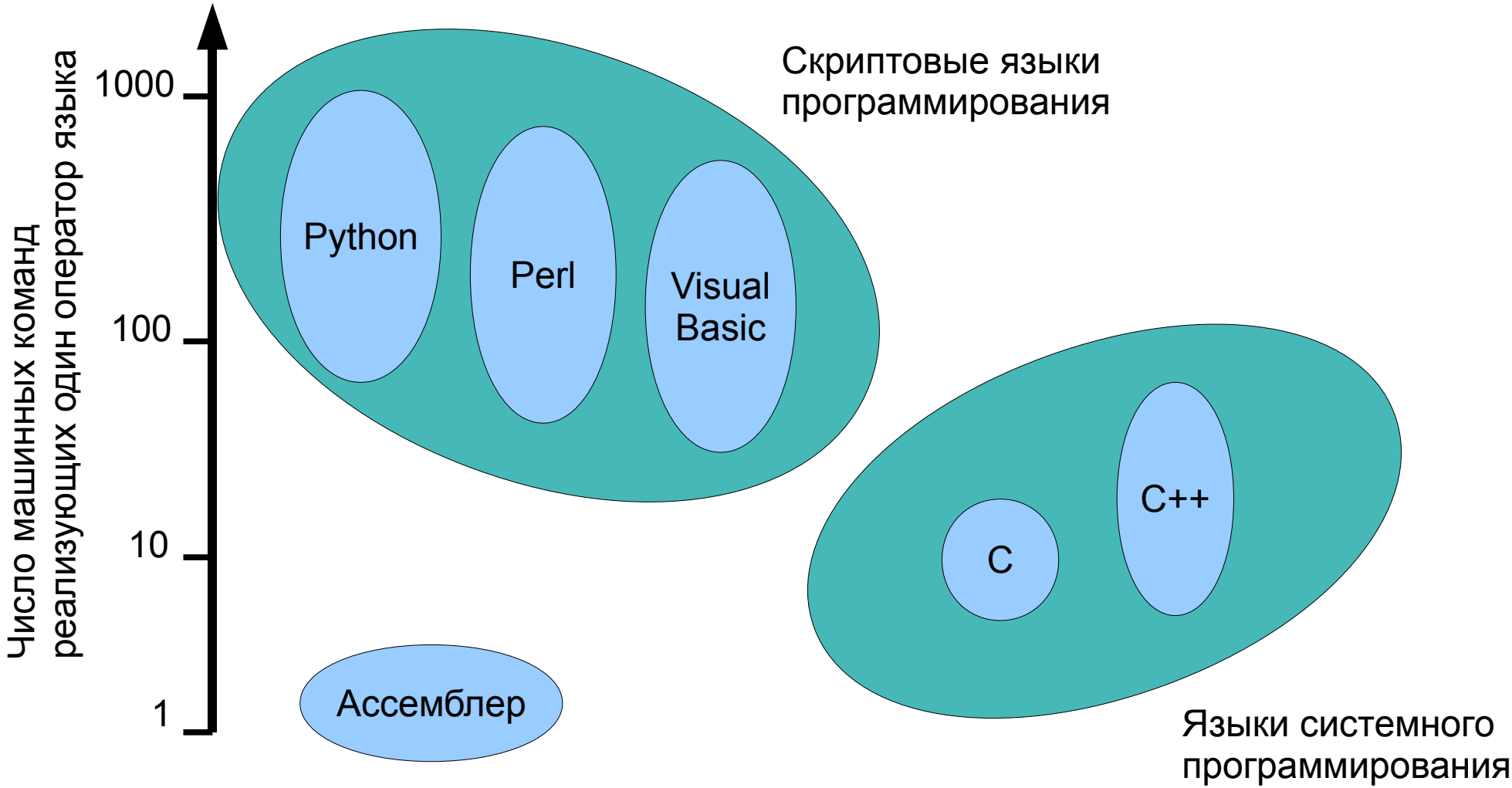
- Введение
- Особенности скриптовых языков программирования
- Типы скриптов и примеры
- Заключение

Введение

Иерархия языков программирования

- Машинные коды, язык ассемблера.
  - Работа с отдельными ячейками памяти, регистрами
- Компилируемые в машинные коды (C, C++).
  - Работа с примитивными типами данных (числа, массивы, структуры/классы)
- Компилируемые в байт-код или интерпретируемые (Java, C#, Python).
  - Более сложные типы данных и более сложные элементарные операции над ними
- Работающие со сложными типами данных (с таблицами, программами — SQL, shell)

Сложность элементарных операций языка



Эволюция языков программирования

- Программы на высокоуровневых скриптовых языках по сравнению с языками системного программирования
  - разрабатываются быстрее в 4-60 раз,
  - код короче в 2-50 раз,
  - имеют большую функциональность,
  - медленнее работа программы в 10-1000 раз
- Работа программиста стала значительно дороже вычислительной техники
- Вычислительная техника развивается столь стремительно, что применение сильно высокоуровневых языков программирования является оправданным во многих задачах.

Появление скриптов

- 1960-е годы. Shell-скрипты. Автоматизация работы человека-оператора по вводу команд ОС в системах с разделением времени
- «Скриптовый язык», «язык сценариев» — язык записи «сценария», последовательности выполнения команд ОС

Особенности скриптовых языков программирования

Назначение скриптовых языков программирования

Быстрое и простое связывание и управление готовыми объектами (функциями, программами)

- Для создания новых программ на основе существующих
- Для автоматизации различных рутинных операций
- Для быстрой разработки технологических тестов
- Для задания сценариев работы программы не программистами
- Для управления специальными данными

Типы данных

- Бестиповые — более абстрактный и универсальный код
- Универсальные типы — возможность произвольного связывания различных компонент

Недостаток:

- Нетипизированность данных не позволяет выявить ошибочное использование переменных до начала выполнения скрипта

Особенности среды выполнения скриптов

Выполнение скриптов:

- Практически никогда не компилируются в машинные коды
- Компилируются в байт-код (обычно лишь для оптимизации скорости выполнения, кешируются)
  - Возможно выполнение JIT
- Чаще всего интерпретируются «на лету»

Многие скрипты — кроссплатформенные

Особенности синтаксиса

- Чаще всего позволяет «построчное» выполнение кода — даёт возможность программирования «на лету», прямо во время выполнения программы
- Минимализм в конструкциях языка

Типы скриптов и примеры

Классификация типов языков

- Управление последовательностью команд ОС
  - Unix Bash
  - Windows cmd.exe
- Обработка текстов
  - AWK
- Web-скрипты
  - Серверные: PHP
  - Клиентские: JavaScript
- Языки общего назначения
  - Perl
  - Python
  - Tcl
  - Ruby
  - Lua
- Узконаправленные языки
  - Программирование в 3D-редакторе Autodesk 3ds Max
  - Язык статистической обработки данных GNU R
  - Простое программирование в играх
  - Программирование в Microsoft Office

Управление последовательностью команд ОС

- «Shell» — «оболочка» — интерфейс между функциями ядра/системы и конечным пользователем
  - GUI — graphical user interface
  - CLI — command-line interface
- Первые оболочки ОС — текстовые (CLI)
- Первая командная оболочка — shell — 1963 г., MIT, для ОС с разделением времени
- Четыре поколения shell:
  - Thomson shell, Mashey shell
    - Базовые синтаксические конструкции, реализованные как отдельные программы
    - Нет переменных
  - C-shell, Bourne shell
    - Более продвинутый синтаксис
    - Переменные

    Дальнейшее развитие синтаксиса и объектной модели
  - tcsh, ksh88
  - ksh93, bash, zsh, Microsoft Power Shell

Unix CLI. Bash. Общая характеристика

- **Thompson shell** — первая оболочка для ОС Unix, разработал Ken Thompson в в AT&T Bell Laboratories в 1971
- **sh** — *Bourne shell*, разработал Stephen Bourne в AT&T Bell Laboratories и был выпущен в 1977 как оболочка по умолчанию для Version 7 Unix. Заменил Thompson shell
- **Bash** — *Bourne-again shell, p*азработал в 1987 Brian Fox в рамках проекта GNU.
  - Обратно совместимое надмножество над sh
  - До сих пор продолжает развиваться и широко используется в качестве shell по умолчанию в различных Unix-like ОС
  - Сейчас — полноценный язык программирования, по историческим причинам ориентированный на задание последовательности выполнения команд ОС
  - «Стандартной библиотекой» для bash является набор стандартных утилит Unix.

Bash. Примеры

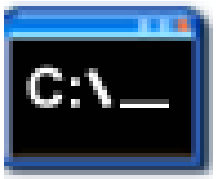

```
function gcd {
  a=$1
  b=$2
  while [ $a -gt 0 ]; do
    c=$a
    a=$[$b % $a]
    b=$c
  done

  return $b
}

gcd 42 56
echo $?
```

```
# Поиск строки в файлах
find /tmp -exec grep -H "search string" '{}' \; -print
# Создать список из мультимедийных файлов
find . -iname '*.avi' -o -iname '*.mpg' -o -iname '*.mov'
> list.m3u
# Обработка изображений в текущей директории
for i in *.JPG; do jpegtran -grayscale $i | jpegtopnm |
pnmnorm -wp 10 | pnmscale 0.5 | pnmtojpeg >
converted/`echo "$i" | tr A-Z a-z`; done
# Перекодирование имён файлов из koi8-r в utf8
find . -depth -execdir sh -c "src=\`basename '{}' \`
dest=\`echo \$src | recode -f koi8-r..utf8\`; [ \"\$src\"
== \"\$dest\" ] || mv \"\$src\" \"\$dest\" " \;
```

Windows CLI. Общая характеристика

- **COMMAND.COM** — оболочка по умолчанию для ОС DOS, и CLI ОС MS Windows 9x/Me
  - Однозадачность
- **CMD.EXE** — CLI для ОС MS Windows не ниже Windows 2000, разработал Therese Stowell для MS Windows NT
  - Скудные возможности конструкций языка и управления объектами
  - Необходимый минимум для задания последовательного выполнения команд с несложной логикой
  - Скудный набор утилит
- **Windows Script Host** — технология создания скриптов, используя Active Scripting, первоначально разрабона для Windows 98
  - Возможность написания скриптов на JScript, VBScript и других языках
  - Интерфейс для работы с COM
- **Windows PowerShell** — CLI для MS Windows не ниже Windows XP SP2, разработала Microsoft в 2006 г.
  - Предоставляет возможности полноценного скриптового языка
  - Интерфейс для работы с COM
  - Интерфейс для работы с .NET

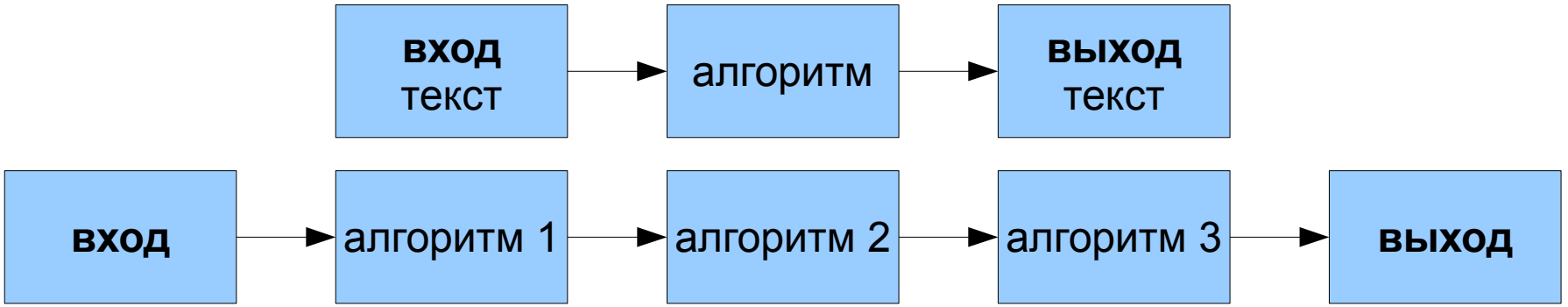
cmd.exe. Примеры




```
goto :main

:gcd
  set a=%1
  set b=%2

  :loop_start
    if not %a% gtr 0 exit /B %b%
      set /a c=%a%
      set /a a=%b% %% %a%
      set /a b=%c%.
  goto loop_start

:main
call :gcd 42 56
echo %errorlevel%

rem Разбор файла myfile.txt с табличными данными
rem Вывод столбцов 2, 3 и всех следующих через переменные %i, %j, %k
for /F "eol=; tokens=2,3* delims=," %i in (myfile.txt) do @echo %i %j %k
```

Языки для обработки текстов

- Текст — цепочка символов — универсальный тип данных
- Алгоритмы, принимающие цепочку символов и возвращающие цепочку, можно связывать друг с другом в произвольном порядке



- Shell превосходно подходит для обработки цепочек символов: вход и выход программ — цепочки символов и shell имеет встроенную возможность для перенаправления вывода одной команды ОС на вход другой

AWK

- **AWK** — язык программирования, созданный для потоковой обработки текстовых данных
- Первоначально разработан в AT&T Bell Labs в 1977 г.. Авторы: Alfred Aho, Peter Weinberger и Brian Kernighan
- Язык разбора входного потока
  - Входной поток состоит из записей (строк)
  - Записи состоят из полей (слов)
- Удобная обработка текстовых таблиц
- Программа на AWK — набор правил

  шаблон { действие }
- Встроенная поддержка регулярных выражений
- Удобный доступ к метаинформации о таблице:
  - NR - номер текущей записи,
  - NF - число полей в текущей записи,
  - Переменные для задания разделителей записей, полей.

- Примеры:
  - `# Для всех строк с третьем полем начинающимся на «Иванов» выведёт значения первого и пятого полей`

    ```
    $3 ~ /^Иванов/ { print $1, $5 }
    ```
  - `# Печать всех строк длиннее 80 символов`

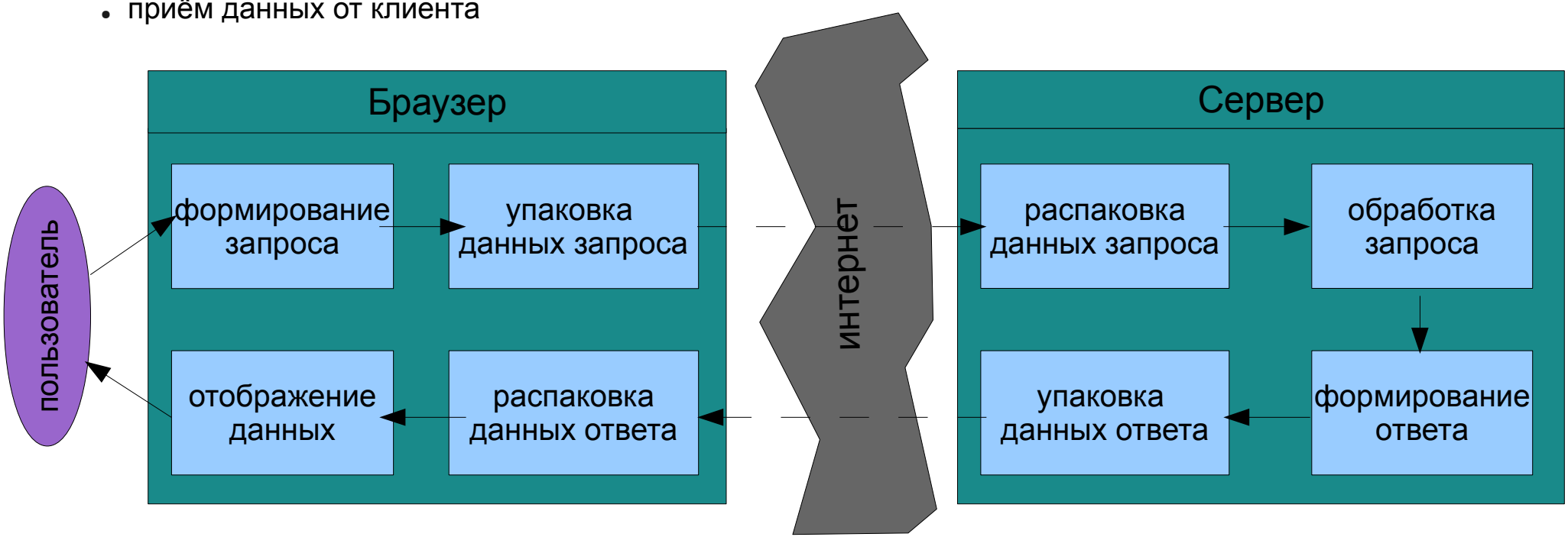
    ```
    { if (length($0) > 80) print $0 }
    ```
  - `# Подсчет числа строк, удовлетворяющим регулярным выражениям`

    ```
    /aaa/   { m["aaa"]++; }
    /bbb/   { m["bbb"]++; }
    END     { for( i in m ) print("m["i"] =", m[i]);
    ```
  - `# Подсчет числа строк, слов и символов`

    ```
    {
        w += NF
        c += length + 1
    }
    END { print NR, w, c }
    ```

Web-скрипты

- Чаще всего обмен информацией организован внутри пар клиент-сервер
- Общие для большинства схем взаимодействия клиент-сервер рутинные операции:
  - генерация данных на сервере,
  - подготовка к передаче клиенту,
  - отображение данных на стороне клиента,
  - приём данных от клиента



Серверные web-скрипты. PHP

- **PHP** — *PHP: Hypertext Preprocessor* — высокоуровневый язык, предназначенный для генерирования HTML-страниц и работы с базами данных
  - Первоначально разработал датчанин Rasmus Lerdorf в 1994 г.
- Один из самых популярных скриптовых языков для веб-программирования
- Ориентирован на
  - Обработку web-запросов
    - Предопределённые глобальные массивы с разобранными параметрами запроса
  - Генерацию web-страниц
    - Интеграция с HTML
    - Поддержка работы с базами данных
- Динамическая типизация
- Объектно-ориентированный
- Высокая скорость работы
- Безопасный
- Интерпретируемый, компилируемый в байт-код и в машинные коды
- PHP используют для web-программирования — Facebook, Wikipedia, Yahoo!, Digg, Joomla, WordPress, Drupal

PHP. Примеры

- Интеграция с HTML на сервере

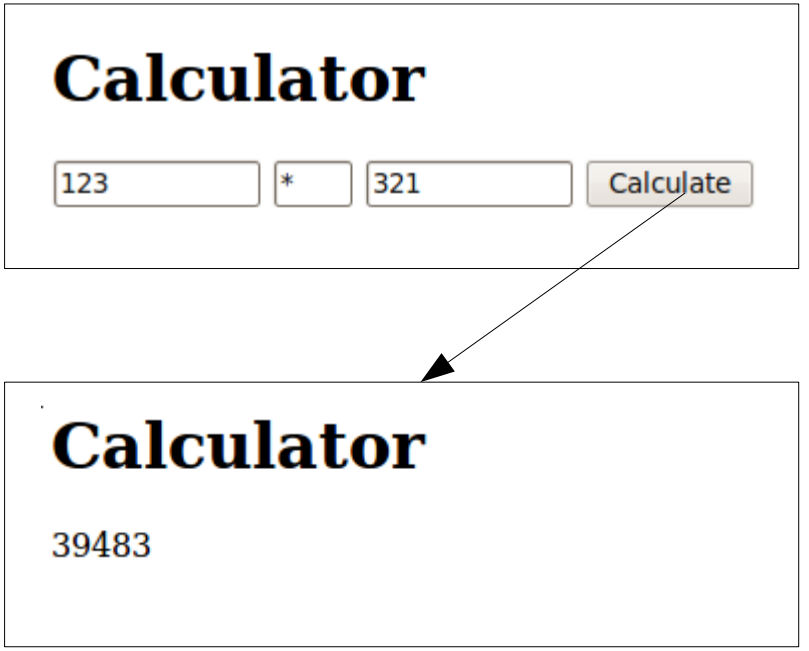
```
<html>
 <body>
  <?php echo 'Hello, world!'; ?>
 </body>
</html>
```

- GCD

```
function gcd( $a, $b ) {
  while ($a > 0)
  {
    $c = $a;
    $a = $b % $a;
    $b = $c;
  }
  return $b;
}

echo gcd(42, 56) . "\n";
```

- Калькулятор

```
<h1>Calculator</h1>
<?php
if ($submit)
{
   if ($operator == '*') {
       echo $numa * $numb;
   } elseif ($operator == '/') {
       echo $numa / $numb;
   } elseif ($operator == '+') {
       echo $numa + $numb;
   } elseif ($operator == '-') {
       echo $numa - $numb;
   }
} else { ?>
<form method="POST" action="<?php $_SERVER['PHP_SELF']; ?>">
  <input type="text" name="numa" size="10">
  <input type="text" name="operator" size="2">
  <input type="text" name="numb" size="10">
  <input type="submit" value="Calculate" name="submit">
</form>
<?php } ?>
```



Клиентские Web-скрипты. JavaScript

- **JavaScript** — высокоуровневый язык встраиваемый в web-страницы и предназначенный для программирования поведения браузера (рассматривается **Client-side JavaScript**)
  - Первоначально разработал Brendan Eich в Netscape в 1995 г.
- JavaScript — одна из реализаций стандартизированного языка ECMAScript. Реализация ECMAScript от Microsoft — JScript
- Ориентирован на
  - Встраивание в клиентскую web-страницу для динамического изменения содержимого страницы
    - Доступ к DOM страницы
  - Организация интерактивного взаимодействия с пользователем
    - Обработка нажатий клавиш, движения мышкой. Работа со временем
  - Работу с окружением браузера — cookies
- Динамическая типизация — слаботипизированный
- Объектно-ориентированный
  - Прототипный
  - Объект — ассоциативный массив, отображает имя поля в значени поля
- Поддержка парадигмы функционального программирования
  - Функции высшего порядка
  - Замыкания
- Поддерживает модель событийно-ориентированного программирования
- Механизм обработки исключений
- Поддержка регулярных выражений
- Безопасное выполнение в закрытом окружении браузера

JavaScript. Примеры

- Встраивание в HTML код на стороне клиента

```
<html>
<body>
<script type="text/javascript">
document.write("<h1>Hello World!</h1>");
</script>
</body>
</html>
```

- GCD

```
function gcd(a, b) {
  while (a > 0)
  {
    var c = a;
    a = b % a;
    b = c;
  }
  return b;
}

document.write(gcd(42, 56));
```

- Калькулятор

```
<html>
<body>
<form name="calc">
<input type="text" name="input"
size="16">
<input type="button" name="evaluate"
value="evaluate"
OnClick="calc.result.value =
eval(calc.input.value)">
<input type="text" name="result"
size="16">
</form>
</body>
</html>
```

Языки общего назначения

Perl. Общая характеристика

- **Perl** — Practical Extraction and Reporting Language — высокоуровневый язык общего назначения.
  - Первоначально разработал Larry Wall в 1987 в США для обработки отчетов
- Предоставляет широкие возможности для обработки текстов
- Ориентирован на решение практических задач, нежели красоту/изысканность синтаксических конструкций
- Процедурный, с поддержкой объектно-ориентированной и функциональной парадигм программирования
- Автоматическое управление памятью. Сборщик мусора
- Интерпретируемый. В Perl 6 появится возможность компилирования в байт-код
- Специальные синтаксические конструкции для работы с регулярными выражениями и потоками ввода/вывода
- Девизы:
  - «There's more than one way to do it» (TMTOWTDI) — «Есть больше одного способа сделать это»
  - «Easy things should be easy and hard things should be possible» — «Простые вещи должны быть простыми, а сложные вещи — возможными»

Perl. Синтаксис

- Унаследовал общую структуру синтаксиса от языка Си
- Возможно создавать сложные структуры данных, как массивы/хеши ссылок на более простые типы
- Возможность создания модулей
- Примеры.

```
sub GCD {
  my ($a, $b) = @_;
  while ($a > 0)
  {
    $c = $a;
    $a = $b % $a;
    $b = $c;
  }
  return $b;
}
print GCD(42, 56);
```

```
$number = 5;
$string = "String";
$multilined_string = <<EOF;
Multilined string,
terminating with the word "EOF".
EOF

@array = (1, 2, "three");
print $array[1]; # "2"

%hash = ('one' => 1, 'two' => 2);
print $hash{'one'}; # "1"

$ref = \$number;
print $ref; # SCALAR(0x14ef640)
print $$ref; # 5

$circle={
  center=>{x=>210, y=>297},
  radius=>53,
};
```

Perl. Особенности

- Встроенная в синтаксис поддержка регулярных выражений
  - `$x =~ m/abc/; # Истина, если $x содержит «abc»`
  - `$x =~ s/abc/def/; # Заменит «abc» на «def» в $x`
  - `$x =~ m/(\d+)\.(\d+)\.(\d+)\.(\d+)\./ # Найдёт первое вхождение IP-адреса, и сохранит его части в переменных $1, $2, $3, $4`
  - `@a = ($x =~ m/(\d+)/g); # Сохранит в массив @a все найденные в строке числа`
  - `$x =~ s/(\d+) б /$1 Б/g # Заменит «X б» на «X Б»`
- Регулярные выражения Perl удобны и включены во многие другие языки программирования
- Пример. Печать простых чисел:

```
perl -wle '(1 x $_) !~ /^(11+)\1+$/ && print while ++ $_'
```

- Встроенные в синтаксис возможности для работы с потоками ввода/вывода

```
while ( <> ){ # построчное чтение из выбранного потока в переменную $_
  if ( /print/ ){ # если строка в $_ содержит «print» (регулярное выражение)
    print;     # печатаем содержимое переменной $_
  }
}
```

Perl. Применение

- Применение Perl:
  - Разработка инструментов системного администрирования
  - Обработка почты
  - CGI-сценарии
  - Обработка текстов
  - Работа с базами данных
  - Разработка средств тестирования
- Perl используют:
  - Amazon — онлайн сервисы
  - The BBC — обработка данных
  - Ebay — онлайн сервисы
  - Yahoo! — онлайн сервисы
  - NASA — программирование в системах рассчетов

Python. Общая характеристика

- **Python** — высокоуровневый язык общего назначения
  - Первоначально разработал Guido van Rossum в 1990 в нидерландском CWI
- Акцент на производительность разработчика и читаемость кода
- Объектно-ориентированный, с поддержкой процедурной и функциональной парадигм
- Минималистичный синтаксис
- Динамическая типизация
- Автоматическое управление памятью. Сборщик мусора
- Полная интроспекция
- Механизм обработки исключений
- Интерпретируемый и/или компилируемый в байт-код
- Интерактивный режим выполнения

Python. Синтаксис

- Отступы определяют разделение программы на блоки
- Возможность написания классов
- Поддержка разделения на модули
- Переменные — имена ссылок на экземпляры классов
- Коллекции (контейнеры):
  - списки,
  - кортежи,
  - словари,
  - множества,
  - некоторые другие типы
- Примеры:

```
def GCD(a, b):
    while b > 0:
        a,b = b,a%b
    return a

print GCD(42, 56)

number = 5
complex_number = 1.5 + 0.5j
print complex_number.imag # "0.5"
big_number = 2**1000
print len(str(big_number)) # "302"
```

```
string = "String"
multilined_string = """
Multilined string,
Yep.
"""
unicode_string = u"Уникод"

list_var = [1, 2, 'three']
tuple_var = (1, 2, 'three')
print list_var[1] # "2"

dictionary = {'one': 1, 'two': 2}
print dictionary['one'] # "1"
set_var = set([1, 2, 'five'])
print 'five' in set_var # "True"

squares = [v ** 2 for v in range(0, 10) if
random.randint(0, 1) == 1]
for i, v in enumerate(squares):
  print "%2d\t%2d" % (i, v)
# Вывод:
# 0       0
# 1       1
# 2       9
# 3      16
# 4      25
# 5      49
```

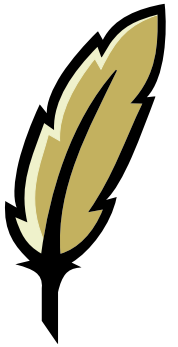
Python. Особенности

- Полная интроспекция
- Поддержка ООП
  - Абстракция — утиная типизация
  - Инкапсуляция — ограниченная ввиду интроспекции
  - Множественное наследование
  - Полиморфизм — все функции виртуальные
- Механизм обработки исключений
- Концепция итераторов
  - Единая схема итерирования
  - Лёгкое создание генераторов
  - Мощная библиотека для работы с итераторами
- Встроенная в синтаксис поддержка парадигм функционального программирования
  - Функции высшего класса
  - Лямбда-функции
  - List comprehension
- Декораторы
- Встроенное документированние кода

Python. Применение

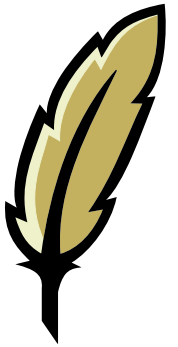
- Применение Python:
  - Часто используется как основной язык программирования
  - Быстрое прототипирование
  - Разработка инструментов для системного администрирования
  - Веб-программирование
  - Встроенный скриптовый язык
- Python используют:
  - Как встроенный скриптовый язык
    - 3D редакторы: Maya, MotionBuilder, Softimage, Blender
    - 2D редакторы: GIMP, Inkscape, Scribus, and Paint Shop Pro
    - ГИС решение ESRI ArcGIS
  - Программирование в играх: Civilization IV, EVE Online
  - YouTube — для системного многграммирования
  - Для решения широкого спектра задач: Google, Yahoo!, CERN, NASA

Tcl. Общая характеристика



- **Tcl** — *Tool Command Language* — «тикль» — высокоуровневый язык общего назначения
  - Первоначально разработал John Ousterhout в 1988 в Калифорнийском институте в Беркли, США
- Процедурный язык программирования с поддержкой
  - метапрограммирования,
  - функциональной парадигмы,
  - модели управления программой на основе событий
- Базовая реализация не содержит поддержки ООП, но есть альтернативные реализации, включающие её
- Ортогональность
- Компилируется в байт-код, а затем выполняется

Tcl. Синтаксис. Особенности



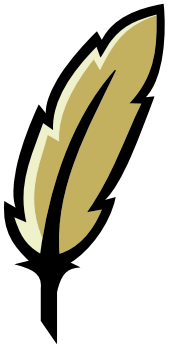
- Скрипт состоит из команд, разделённых переводом строки или точкой с запятой
- Команда:

  CommandName arg1 arg2 … argN
- Базовый тип данных — строки
- Контейнеры:
  - список строк,
  - ассоциативный массив
- Функциональная парадигма используется редко
  - Функции высшего порядка и абстракции функции
- Модель событийно-ориентированного программирования
  - Tcl/Tk — библиотека для задания GUI

- Примеры:

```
proc gcd { a b }{
  while { $b > 0 } {
    set t $b
    set b [expr $a % $b]
    set a $t
  }
  return $a
}
puts [gcd 42 56]

set number 5
set string "Some string with
    number $number"

set string_list {e1 e2 e3}
set map_var(one) 1
set map_var(two) 2
puts $map_var(two) # «2»
```

Tcl. Применение



- Применение Tcl:
  - Быстрое прототипирование
  - Создание графических интерфейсов (Tcl/Tk)
  - Встроенный скриптовый язык
  - Тестирование
  - Написание хранимых процедур PostgreSQL (PL/Tcl)
  - Иногда создание CGI-скриптов
- Tcl используют:
  - BMW — пользовательские интерфейсы и интегрирующая среда системы управления производственными процессами реального времени на заводах
  - AOL — встроенный язык программирования в веб-сервере AOLserver
  - Cisco, HP, Motorola — системы тестирования

Ruby. Общая характеристика

- **Ruby** — высокоуровневый язык общего назначения
  - Первоначально разработал японец Юкихиро Мацумото в 1995 г.
- Акцент на
  - понятность кода,
  - высокую скорость разработки,
  - простоту разработки,
  - автоматизацию решения рутинных задач
- Принцип «наименьшей неожиданности»
- Лаконичный синтаксис
- Динамическая типизация
- Объектно ориентированный, с поддержкой процедурной и функциональной парадигм
- Рефлективный
- Независимая от ОС реализация многопоточности
- Автоматическое управление памятью. Сборщик мусора
- Интерактивный режим выполнения
- Интерпретируемый. Начиная с версии 1.9 — компилируемый в байт-код и выполняемый на виртуальной машине

Ruby. Синтаксис

- Полностью объектно-ориентированный язык:
  - все данные — объекты,
  - все функции — методы
- Поддержка разделения на модули
- Переменные — имена ссылок на экземпляры классов
- Базовые коллекции:
  - массивы,
  - хеши,
  - другие типы
- Большое количество вариантов коллекций реализовано в стандартных библиотеках
- Примеры:

```
def gcd(a, b)
  while b > 0
    a,b = b,a%b
  end
  a
end

puts gcd(42, 56)
```

```
array_var = [1, 2, 'three']
puts array_var[1] # "2"

hash_var = {'one' => 1, 'two' => 2}
puts hash_var['one'] # "1"

# Range от 0 до 10
(0..10).
  # новый Range из квадратов
  collect{|v| v ** 2 }.
  # новый Range из некоторых элементов старого
  select{ rand(2) == 1 }.
  # для каждого элемента печать индекса и самого элемента
  each_with_index{|v,i| printf "%2d\t%2d\n", i, v }
# Вывод:
# 0        0
# 1        4
# 2        9
# 3       16
# 4       64
```

Ruby. Особенности

- Объектно-ориентированный
  - Инкапсуляция
    - Доступ к данным через методы
    - Public, private, protected члены
  - Вызов метода — передача сообщения
  - Абстракция — утиная типизация
  - Наследование
    - Механизм «примесей» вместо множественного наследования
  - Полиморфизм
    - Возможность динамического изменения содержимого класса, экземпляра класса
- Встроенная в синтаксис поддержка регулярных выражений
- Блоки — именованные замыкания. Итерирование
- Константы
- Механизм обработки исключений
- Встроенная в синтаксис поддержка парадигм функционального программирования
  - Функции высшего порядка
  - Лямбда-функции
  - Continuation
- Поддержка многопоточности
- Метапрограммирование
  - Метаклассы
- «Синтаксический сахар»
- Существенный недостаток: медленный

Ruby. Применение

- Применение Ruby:
  - Иногда используется как основной язык программирования
  - Веб-программирование (Ruby on Rails)
  - Встроенный скриптовый язык
- Ruby используют:
  - Как встроенный скриптовый язык:
    - Google SketchUp, Inkscape, Xchat, RPG Maker, Amarok
  - NASA — программирование имитационных моделей
  - Motorola — программирование эмуляторов, обработка данных

Lua. Общая характеристика

- **Lua** — высокоуровневый язык общего назначения
  - Первоначально разработан подразделением Tecgraf Католического университета Рио-де-Жанейро в Бразилии в 1993 г.
- Рефлективный
- Многопарадигменный:
  - императивный,
  - функциональный,
  - объектно-ориентированный
- Предоставляет гибкие возможности для расширения языка
- Динамическая типизация
- Компилируется в байт-код, затем выполняется на виртуальной машине

Lua. Синтаксис

- Ограниченный набор базовых типов данных
- Единственный вид контейнера — таблица — ассоциативная структура данных отображающая ключ в значение
- Таблицы используются для реализации
  - структур,
  - областей видимости,
  - массивов,
  - парадигмы объектно-ориентированного программирования
- Примеры:

```
point = { x = 10, y = 20 }
print(point["x"]) -- "10"
print(point.x)    -- "10"
```

```
array = { 1, 2, "three" }
print(array[1])    -- "1"

function Point(x, y)
  return { x = x, y = y }
end
array = { Point(10, 20), Point(30, 40),
Point(50, 60) }
print(array[2].y)  -- "40"

function gcd(a, b)
  while b > 0 do
    a,b = b, a % b
  end
  return a
end

print(gcd(42, 56))
```

Lua. Особенности

- Метатаблицы
- Поддерживает парадигмы функционального программирования
  - Функции высшего порядка
  - Замыкания
- Использует небольшие объёмы памяти

Lua. Применение

- Применение Lua:
  - Встроенный скриптовый язык (в основном в видео играх)
  - Язык программирования встраиваимых систем
- Lua используют:
  - Игры:
    - Baldur's Gate, Crysis, Far Cry, Ragnarok Online, World of Warcraft
  - Apache HTTP Server — можно использовать для обработки запросов
  - Cisco — реализация Dynamic Access Policies в ASA
  - Awesome — для задания конфигурации
  - Logitech — для программирования биндингов некоторых клавиатур
  - ASRC Aerospace — сбор и обработка информации для отслеживания утечек опасных газов в ходе запуска космических кораблей

Узконаправленные языки (1/3)

- MAXScript — встроенный скриптовый язык в 3D редактор Autodesk 3ds Max
  - Доступ к иерархии объектов сцены
  - Набор специальных типов
    - Геометрические примитивы
    - Материалы
    - Элементы GUI для создания дополнений
  - Интерактивный и batch режим
  - Возможность записи действий пользователя в скрипт
  - Используется для
    - Автоматизации рутинных операций
    - Написания дополнений (эффекты, преобразования геометрии)
    - Написания конвертеров форматов

- Пример: раскраска выделенных объектов

```
color_01 = color 0 0 80
color_02 = color 255 255 255

stepsize_r = (color_02.r - color_01.r)/$.count
stepsize_g = (color_02.g - color_01.g)/$.count
stepsize_b = (color_02.b - color_01.b)/$.count

for i = 1 to $.count do (
  newColor = (
    color ((i*stepsize_r)+ color_01.r-stepsize_r)
    ((i*stepsize_g)+color_01.g-stepsize_g)
    ((i*stepsize_b)+color_01.b-stepsize_b))
  $[i].wirecolor = newColor
)
```

Узконаправленные языки (2/3)

- **R** — высокоуровневый язык программирования для статистической обработки данных

  - Первоначально разработали Ross Ihaka и Robert Gentleman в 1993 г.
  - Поддерживает широкий спектр статистических и численных методов
  - Легко расширяемый с помощью модулей
  - Возможности генерации качественных графических визуализаций
  - Основные типы данных: векторы чисел и матрицы (многомерные вектора)
  - Мощные средства для работы с векторами чисел
    - В основном для решения математических задач
  - Средства для генерации последовательностей
  - Примеры

```
# Создание вектора
x <- c(1,2,3,4,5,6)
# Поэлементное возведение в квадрат
y <- x^2
```

```
gcd <- function(a, b) {
  while (a > 0) {
    c <- a
    a <- b %% a
    b <- c
  }
  b
}

print(gcd(42, 56))
```

- Простейшие скрипты в играх
  - Изменение конфигурации во время игры
  - Автоматизация последовательности действий
  - Примеры:
    - Запуск игры Quake 2 с определёнными настройками

      quake2 +set dedicated 1 +set port 27920 +set game ctf +exec server.cfg
    - Задание выполнения нескольких действий при нажатии кнопки в игре Half-Life

      bind q "+lj"

      alias +lj "+duck;wait;+jump" alias -lj "-jump;-duck"

Узконаправленные языки (3/3)

Автоматизация в **Microsoft Office**

- Внутренняя
  - VBA
- Внешняя
  - Посредством технологии Windows Script Host (VBScript, JScript)
- Предоставляется интерфейс к иерархии объектов документа и функциям работы с документом
- Возможность записи действий пользователя в скрипт

- Пример: поиск в документе Word на VBScript

```
Set objWord = CreateObject("Word.Application")
strDocPath = "MyDoc.doc"
strSearch = "word"
Set objDoc = objWord.Documents.Open(strDocPath)
With objDoc.Content.Find
    .Text = strSearch
    .Format = False
    .Wrap = wdFindStop
    Do While .Execute
       iStrCount = iStrCount + 1
    Loop
End With
If iStrCount = 1 Then
  WScript.Echo strSearch & " appears once in" &
vbCrLf & strDocPath
Else
  WScript.Echo strSearch & " appears " & iStrCount
& " times in " & vbCrLf & strDocPath
End If
objDoc.Close(wdDoNotSaveChanges)
objWord.Quit
```

Общее сравнение скриптовых языков программирования

	Язык	Популярность	Место	Применение	Возник
1	PHP	10.39%	3	Web (серверный)	1994
2	Visual Basic	8.73%	5	Встроенный язык	1998
3	Python	3.91%	7	Общее	1990
4	Perl	3.78%	8	Web (серверный), сис. адм.	1987
5	JavaScript	3.03%	9	Web (клиентский)	1995
6	Ruby	2.46%	10	Web (серверный)	1995
7	Lua	0.53%	20	Видео игры	1993
8	Tcl/Tk	0.23%	37	GUI	1988
9	AWK	0.20%	40	Обработка текстов	1977
10	Bash	-	-	Сис. адм.	1987
11	cmd.exe	-	-	Сис. адм.	1994

Данные о популярности: TIOBE Programming Community Index на октябрь 2009

Заключение

- Скриптовые языки, как языки общего назначения
  - Быстрое прототипирование и реализация программных решений
  - Легкость создания приложений
  - Надёжность
  - Относительно невысокая скорость работы
  - Относительно просты в изучении
- Скриптовые языки, как встроенные языки конкретных систем
  - Работа только с необходимыми данными
  - Решение только необходимых задач
- Язык программирования — только инструмент
  - Для решения каждой конкретной задачи можно найти наиболее подходящий язык

Источники информации

- Интернет-энциклопедия

  Wikipedia — http://wikipedia.org

- Статья

  Джон Остераут. Сценарии: высокоуровневое программирование для XXI века.

  J. K. Ousterhout. Scripting: Higher-Level Programming for the 21st Century//Computer. — 1998. — т.31, №3. http://www.osp.ru/os/1998/03/179470/

- Книга

  Р. У. Себеста. Основные концепции языков программирования, 5-е изд. — М.: Издательский дом «Вильямс», 2001 — 627 с.

Благодарю за внимание!