01. ВВЕДЕНИЕ В WEB

ЛЕКЦИИ ПО ИНФОРМАТИКЕ ДЛЯ СТУДЕНТОВ ВТОРОГО КУРСА
ВЫСШЕЙ ШКОЛЫ ИТИС КФУ
2019

М.М. АБРАМСКИЙ

СТАРШИЙ ПРЕПОДАВАТЕЛЬ

КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ

«Окультуриваемся»

- Принимаем осваиваем картину мира
- Знание сути вещей.
    - **никаких белых пятен!**
- Полное понимание того, что мы делаем на других предметах.
- Знать так, чтобы объяснить другому.
- Облегчаем задачу преподавателям практики.
- ~~Следим за базаром~~ Осваиваем терминологию, правильное использование слов, профессиональный сленг.
    - «Мы вчера запушили, а потом Андрюха пропуллился и смерджил и на продакшн залил, но в итоге деплой вальнулся, при билде джарники на боевом серваке не нашлись»

Software Engineering

Программная инженерия (rus)

1. Программа = Software?
2. Что такое программная инженерия?

Информация

- Сведения, независимо от формы их представления, воспринимаемые **человеком или специальными устройствами** как отражение фактов материального мира в **процессе коммуникации** (ГОСТ 7.0-99).

- Знания о предметах, фактах, идеях и т. д., которыми **могут обмениваться люди** в рамках конкретного контекста (ISO/IEC 10746-2:1996);

Ее нет без коммуникации между **объектами!**

Действия с информацией

- Сбор (считывание)
- Преобразование (кодирование, декодирование)
- Передача
- Обработка
- Хранение
- Отображение (воспроизведение)

- **Информационный процесс** – набор действий с информацией

- **Информационная система** – система, где реализованы информационные процессы.

Примеры

Информационный процесс	Информационная система
Регистрация на сайте (*сбор, преобразование, передача, хранение*)	Сайт (приложение)
Лайк фото в Instagram (*сбор, передача, хранение, обработка, воспроизведение*)	Мобильное приложение
Программирование на Java (сбор, преобразование, хранение, воспроизведение)	Среда разработки

Информация не простая…

…а **цифровая (digital)!**

- Может быть закодирована в виде последовательности цифр (и раскодирована обратно).

- Какие виды представления информации мы знаем?

Представления информации

- *Текст*
- *Картинка*
- *Звук*
- *Видеоряд*
- …

Все это представимо в виде цифровом виде

- Цвет в формате RGB – 3 числа от 0 до 255
- Символ A имеет код 65 = 01000001_2

Данные (data) - информация, представимая в закодированном (в нашем случае – цифровом виде).

Цифровые (информационные) технологии

Технологии работы с [цифровой] информацией.

Управление

В информационных системах есть **управление**.

- Информационные процессы должны идти так, как требуется,
- Что-то в информационных системах должно анализировать информацию и на основании этого принимать решение об изменении в каком-либо информационном процессе.

- **Управление** на греческий язык переводится как …

Кибернетика

Наука о *управлении*

- ***XIX в. – Андре-Мари Ампер***
- ***1948 г. – Норберт Винер***

Нужно организовывать управление информационными процессами в информационных системах.

Алгоритм

араб. Al Khorezmi – хорезмский ученый IX века.

- *Благодаря ему мы используем слова цифра, шифр, алгебра*

Последовательность шагов

Не буду требовать школьного определения и свойств алгоритма

Алгоритмы везде и всегда

Мы впервые познакомились с ними в математике:

- Сложение/умножение/деление столбиком
- Применение теоремы Виета
- Разложение числа на множители
- …

Формализовать понятие алгоритма **пришлось** в 20-30е гг. XX века.

КАРТИНА МИРА

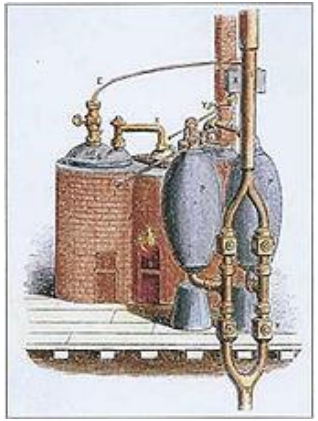
Промышленные революции

В отличии от революций политических – не событие, а **период смены ключевых технологий**, что влечет за собой **изменение** экономической, политической, социальной и других картин мира.

РАЗ - Индустриальная

вторая половина XVII века

текстильная промышленность
паровой двигатель
ковкий чугун
открытие производств



ДВА - Технологическая

середина XIX века – середина XX века

бессемеровский способ выплавки стали
железные дороги
поточное производство, конвейер
химикаты
электричество
машиностроение

ТРИ - Научно-техническая & информационная

коммуникация - телеграф, телефон, радио

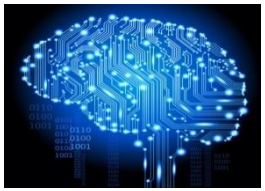
компьютеры

автоматизация и электронизация,

освоение космоса

4ПР - ?

Интернет-вещей



Искусственный интеллект

Интеллектуальная робототехника

Квантовые компьютеры

3D-печать

Big Data

Now

- Мир почти всегда живет на стыке двух ПР.

- Мы живем на стыке 3й и 4й, при этом пользуемся еще и объектами 1й и 2й.
  - Каждая следующая ПР влияет на объекты предыдущих.

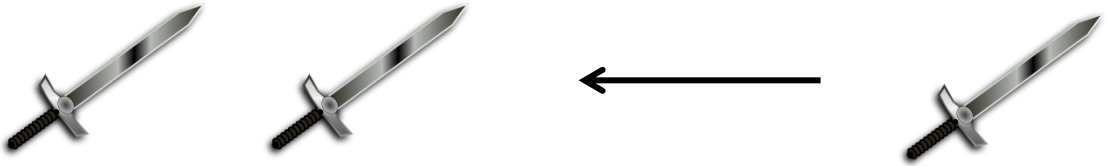
Пример влияния

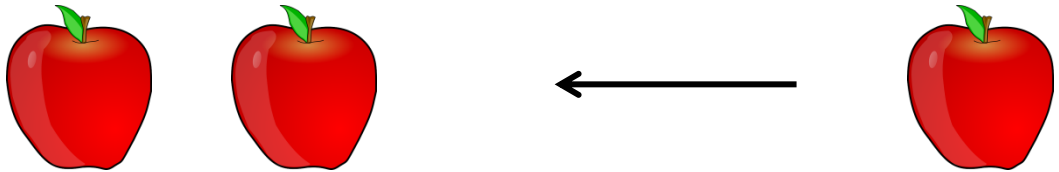
Используем ли мы подобные предметы с той же функцией сейчас?

Можно ли перенести современные аналоги в прошлую ПР?

2. НЕМНОГО О МАТЕМАТИКЕ

Где начинается математика?





A - Абстракция

Суть математики!

Благодаря абстракции математика – царица наук

- *Простым языком:* математике не важно, 12 литров или 12 вольт, поэтому она применяется везде.

Кто такие математики?

Гуру абстракций!

Ученые, профессионально занимающиеся различными абстракциями

- Формулы, уравнения, теоремы – лишь частные случаи абстракций

Кстати

Одно из главных признанных отличий человека от животного – наличие **абстрактного мышления** (умения строить абстракции)

У вас за плечами 10 лет школьной математики (минимум)

К математическим абстракциям вы привыкли, что даже не задумываетесь об этом.

Хотите эксперимент?

Не забыли 1 класс школы?

Определение сложения.
+ работает на двух числах.

https://ds02.infourok.ru/uploads/ex/003e/0005f3ad-63c52be5/310/img1.jpg

Задача

Чему равно **сумма(1, 4, 10, 10)**?

Проведите эксперимент сами

Покажите задачу своим детям/внукам! Кто-то ответит, а кто-то нет – поймете, как у них развито абстрактное мышление.

Как работает абстракция в этом примере:

- Вспомнить, что есть **«сумма»**
- Вспомнить, что сумма связана с оператором «+»
- Понять, что можно сложить первые два числа, к результату суммы прибавить третье число, и т.д.
- Выполнить действия в уме.

3. КАКАЯ СВЯЗЬ С ПРОМЫШЛЕННЫМИ РЕВОЛЮЦИЯМИ?

ПР и математика

Математика пронизывает всё.

Всё развивается – значит и математика должна развиваться!

«Даёшь новым наукам и технологиям – свою, новую математику!»

Когда что изобрели

Что	Когда
Десятичные дроби	XVI
Логарифмы	XVI
Система координат	XVII
Теория вероятности	XVII
Производные	XVII
Дифференциальные уравнения	XVIII
Интегралы	XVIII

Ничего не напоминает?

Какие наблюдения?

XIX век (с одной стороны)

- *С одной стороны* – продолжается **эволюционное** развитие
  - Математический анализ, теория дифференциальных уравнений, линейная алгебра, теория групп

- Математика **перестает** быть просто наукой о числах и счете.
  - Объекты исследования – функции, матрицы, векторы, а также ***абстрактные структуры*** (группы, моноиды, кольца, поля, пространства, алгебры и др.)

XIX век (с другой стороны)

Математики начинают «рефлексировать» - заниматься ~~самокопанием~~ самоанализом:

- Каковы фундаментальные основы математики?
  - » По аналогии с аксиомами геометрии.
- Почему мы используем именно такой язык математики, а не другой?
- Корректно ли вообще все то, чем мы занимаемся?

Один из виновников

V постулат Евклида:
Через точку, не лежащую на данной прямой, можно провести лишь одну прямую, параллельную данной.

Н.И. Лобачевский

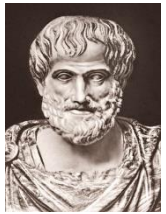
Ученый-геометр, ректор Казанского университета

– Постулат неочевиден. Можно заменить на обратную формулировку:

Через точку, не лежащую на данной прямой, можно провести множество прямых, не пересекающихся с данной.

Новые математические науки XIX

- Математическая логика
  - *Вообще говоря Аристотель (еще в IV в. до н.э)*
  - *Математизация в XIX:* Джордж Буль, Август де Морган



Аристотель

- Теория множеств
  - *Георг Кантор*

Джордж Буль

Август де Морган

Георг Кантор

Взялись претендовать на то, чтобы быть ОСНОВОЙ математики как таковой!

Математическая логика

- A – высказывание, которое может быть истинным или ложным.
  - Пример: A – «Земля плоская»
- $\bar{A}$ – отрицание A («не A»)
  - Пример:
    - » A – «Я сегодня выспался»,
    - » $\bar{A}$ – «Я сегодня не выспался»

Закон двойного отрицания

$$\overline{\overline{A}} = A$$

«Я не не ел» = «Я ел»

Логическая основа доказательств «от противного» между прочим!

- Предположить, что это ***не так***…
- Получить противоречие, значит это ***не не так***, а ***так***.

Вроде логично, но…

Задача

Пусть А = «Данное высказывание ложно»

А – истинно или ложно?

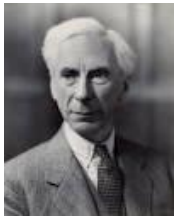
Незадача

- Если А – истинно, то «это высказывание ложно, значит, А – ложно.

- Если А – ложно, значит высказывание не ложно, значит А – истинно.

Парадокс лжеца – частный случай парадокса Рассела в теории множеств (1901)

– *а парадоксов там еще вагон*



Бертран Рассел

2 школы (раскол)

Формалисты

Гилберт, Аккерман, фон Нейман и др.

«Математика – это **формальные логические системы**. Надо просто аккуратно строить непротиворечивые логические системы без возможности парадоксов – и все будет хорошо!»

Давид Гилберт

Вильгельм Аккерман

VS

Интуиционисты

Брауэр, Гейтинг и др.:

«Математика – это абстрактные построения, смысловые **конструкции**. Истинность высказываний должна быть конструктивной - должна строиться или **вычисляться**!»

Джон фон Нейман

Лёйтзен Брауэр

Аренд Гейтинг

Разница простым языком (*не показывайте математикам)*

A + B

Формальный подход:

Если выражение A корректно, и B корректно, и между ними разрешена операция «+», то выражение A + B – тоже корректно **(разрешено)**.

Интуиционистский подход:

Для начала надо определиться, что такое A и B, из чего они состоят, какой у них смысл. Также проверить, что «+» можно посчитать для «A» и «B». Тогда A + B – корректно **(имеет смысл)**.

Были еще логицисты и теоретико-множественная школа, но они на втором плане.

Versus Battle

Гилберт формулирует «Программу развития математики» (*20 проблем Гилберта*).

- И заодно топит интуиционистов: в 1920 году Брауэр **исключен** из Mathematische Allen – главного журнала того времени.

Цитаты Давида Гилберта:

- «Интуиционисты стремятся разрушить и изуродовать математику» (1922 год)
- «Отнять у математиков закон исключенного третьего — это то же самое, что забрать у астрономов телескоп или запретить боксерам пользование кулаками.» (1927 год)

Но

Теорема о неполноте (1931, Курт Гёдель)

В любой формальной системе можно сформулировать правдивое утверждение, которое нельзя доказать.

- Своеобразный аналог «парадокса Лжеца»

Простым языком: формалистам сказали:

– Ха-ха! Как бы вы там не играли в формальные системы, все проблемы математики вы ими не решите!

ИНТУИЦИОНИЗМ = КОНСТРУКТИВИЗМ

Помните, что такое функция?

«Каждому аргументу сопоставлено единственное значение»

(5 класс)

Теперь функция – конструктивное понятие. Она должна быть **вычислимой** - можно описать *последовательность шагов*, как ее посчитать.

- Пример: функция – сложение, способ вычисления – «столбиком».

Последовательность шагов

Алгоритм

Надо как-то описать!

Можно своими словами (*интуитивная вычислимость*). Но после XIX века никто не доверяет «своим словам». И вычислительные устройства не понимают слова.

Появились **абстрактные конструкции**, с помощью которых можно определять и вычислять функции:

- Рекурсивные функции
- Лямбда-выражения Чёрча
- Нормальные алгоритмы Маркова
- …
- **Машина Тьюринга**

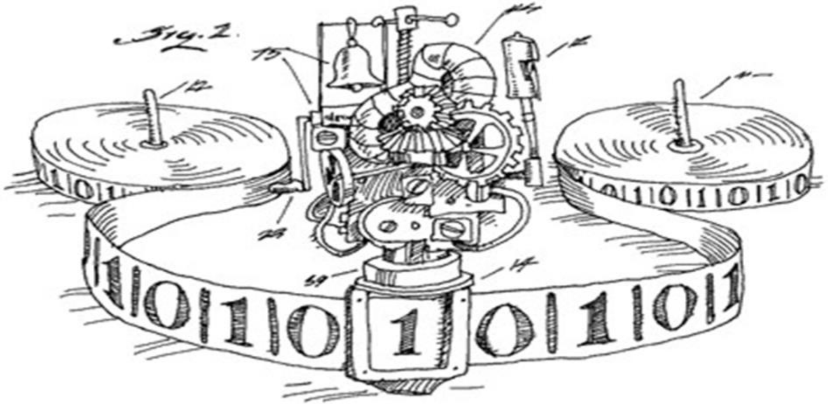
Алонзо Чёрч

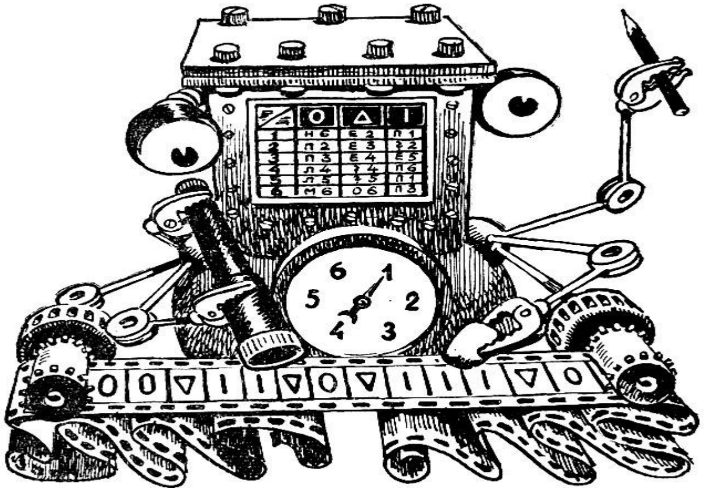
Алан Тьюринг

Андрей Марков

Машина Тьюринга (МТ)

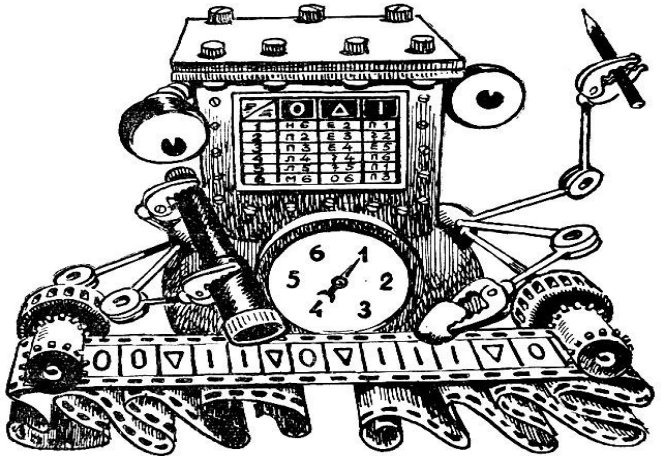
- Алан Тьюринг, 1936
- *Абстрактная модель вычислительного устройства*





Устройство МТ

- Алфавит
- Состояния (память)
- Лента (*бесконечная*)
- Считывающая головка
- ***Программа:***



	s1	s2
0	s1 →	1 stop
1	s1 →	0 s2 ←
^	s2 ←	1 stop

Это функция f(x) = x + 1

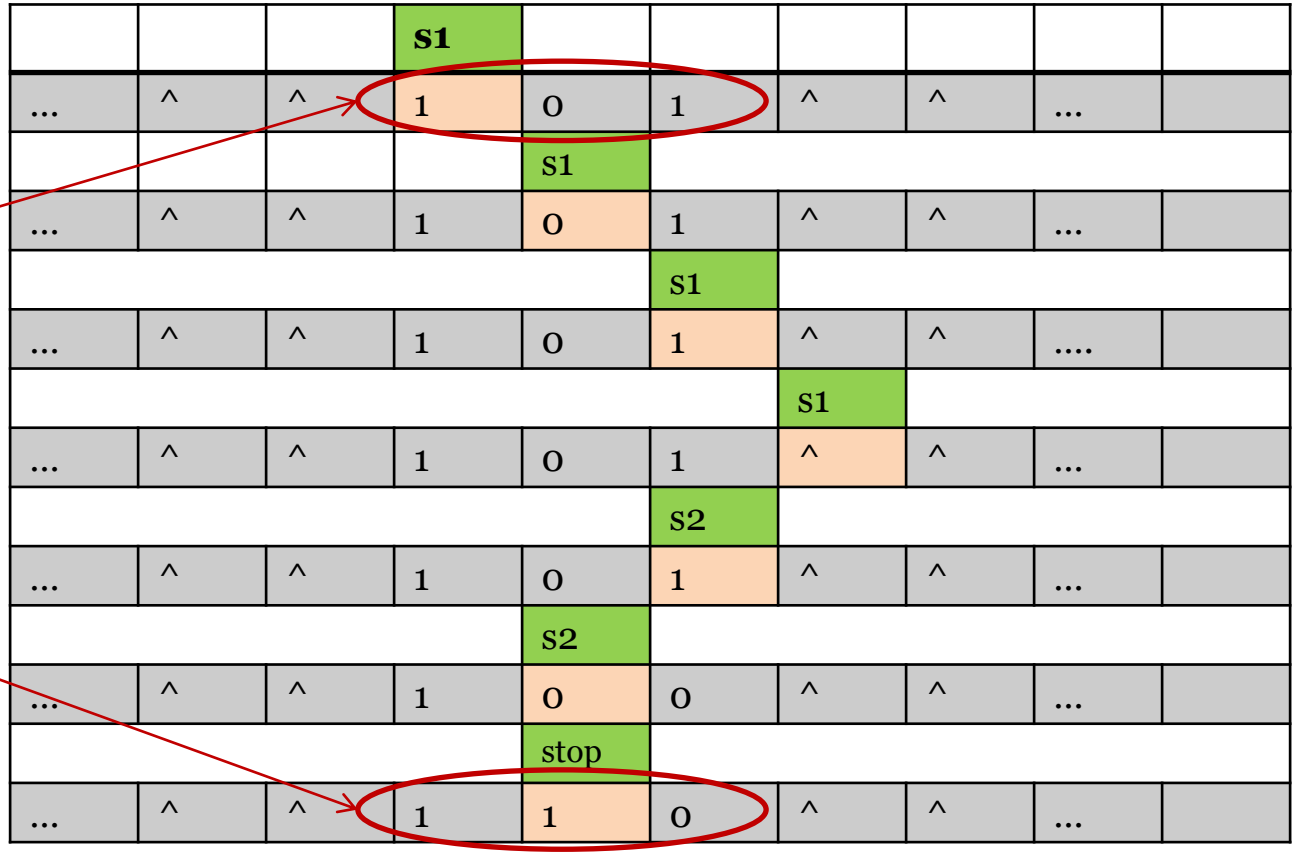
Работа Машины Тьюринга

	s1	s2
0	s1 →	1 stop
1	s1 →	0 s2 ←
^	s2 ←	1 stop

Это функция f(x) = x + 1

Вход (аргумент функции, которую реализует МТ)
101 – двоичный код числа 5.

Выход, результат работы, значение функции f(x) = x + 1
110 – двоичный код числа 6.



Тезис Чёрча-Тьюринга

- Любой [интуитивно вычислимый] алгоритм может быть реализован на машине Тьюринга.

- Написание программ для машины Тьюринга – ***программирование.***

Программа = текст

- Можно выписать в одну строчку
  - Выписываем каждую клеточку, # - разделитель информации о клетках:
    - 0, 1, s1, -> # 0, s2, 1, stop # ...

	s1	**s2**
0	s1 →	1 stop
1	s1 →	0 s2 ←
^	s2 ←	1 stop

- Эту строчку можно подать на вход другой машины Тьюринга

Универсальная машина Тьюринга

Машина Тьюринга, моделирующая работу других МТ

- На вход подают код другой МТ и входные данные
- Универсальная МТ выдает ответ, как если бы работала эта другая МТ

Теорема об универсальной машине (1949, Тьюринг): **Универсальная машина Тьюринга существует!**

Внимание

- Машина Тьюринга решает конкретную задачу.
- Но если взять универсальную машину, мы сможем выполнять **ВСЕ** возможные алгоритмы на **ОДНОМ** устройстве.
  - главное – уметь писать коды других программ!
- Ничего не напоминает? Одно устройство, много алгоритмов, код программы…

Ура!

Теорема о существовании универсальной машины Тьюринга **дала жизнь программированию!**

- Нам не нужно строить кучу разных устройств для каждого алгоритма!

- Один вычислитель (computer), который будет выполнять программы, записанные на определенном языке
  - Язык, на котором пишут программы – **язык программирования**!

Дело за малым

- Нужно построить электронное устройство для универсальной машины.
- Но по факту они уже давно появились.
- Просто не осознавались как устройства для решения всех задач.

Начало

1830е годы

Чарльз Бэббидж придумал аналитическую машину (но не смог построить)

- *Память, ввод/вывод с помощью перфокарт, арифметическое устройство*

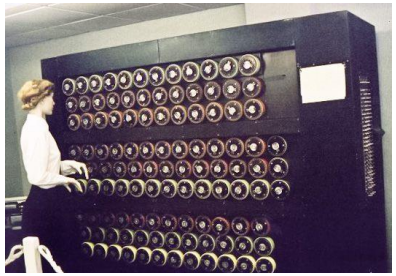
1840е годы

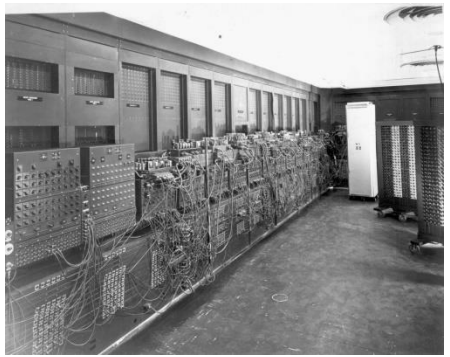
Ада Лавлейс написала первый алгоритм для аналитической машины Бэббиджа

- *Цикл, ячейка памяти*
- Ада Лавлейс считается первым программистом!

2ПР

- Электрический ток, химия, металлургическая промышленность.
- Появились и электронные устройства, способные считать:
  - 1884-1887 – Табулятор Холлерита (США)
  - 1941 – Машина Z3 Конрада Цузе (Германия)
  - 1942 – Цифровой компьютер ABC Атанасова и Берри (США)
  - 1943 – «Колосс» Алана Тьюринга (Великобритания)
  - 1946 – ENIAC (США)
  - 1948-1950 – появление первых советских ЭВМ





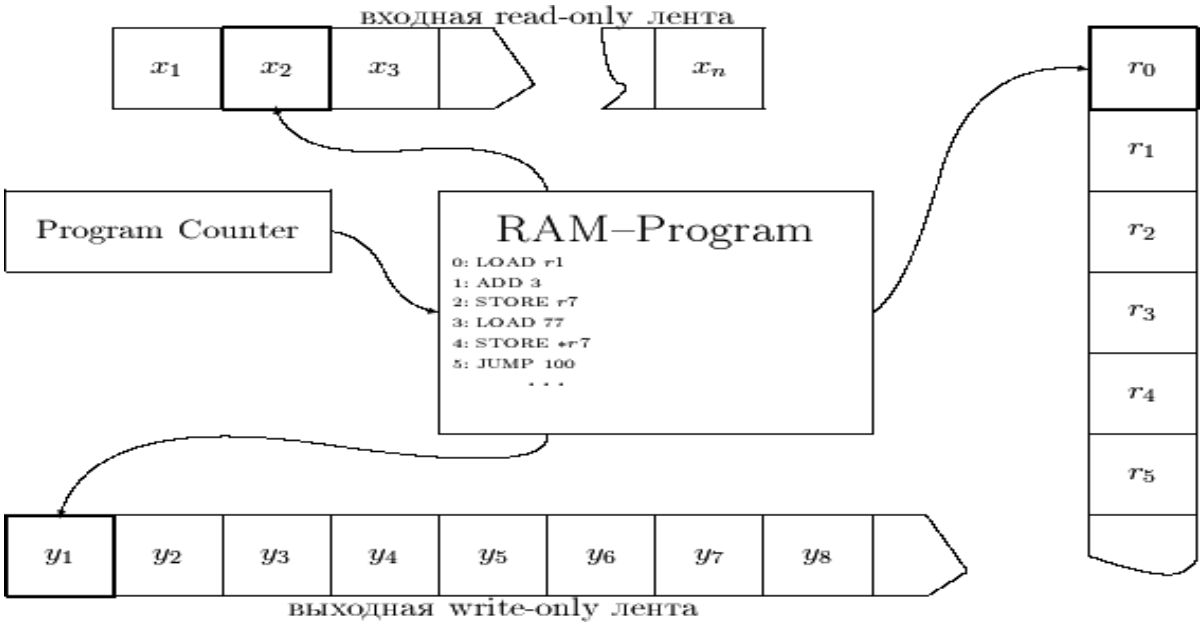
Резюме половины лекции

1. Кризис математики привел к формализации понятия алгоритм.

2. 2ПР революция привела к появлению технологий, позволяющих строить вычислительные устройства.

3. Тезис Чёрча-Тьюринга и теорема об универсальной машине обусловили появление **программирования.**

Random Access Machine



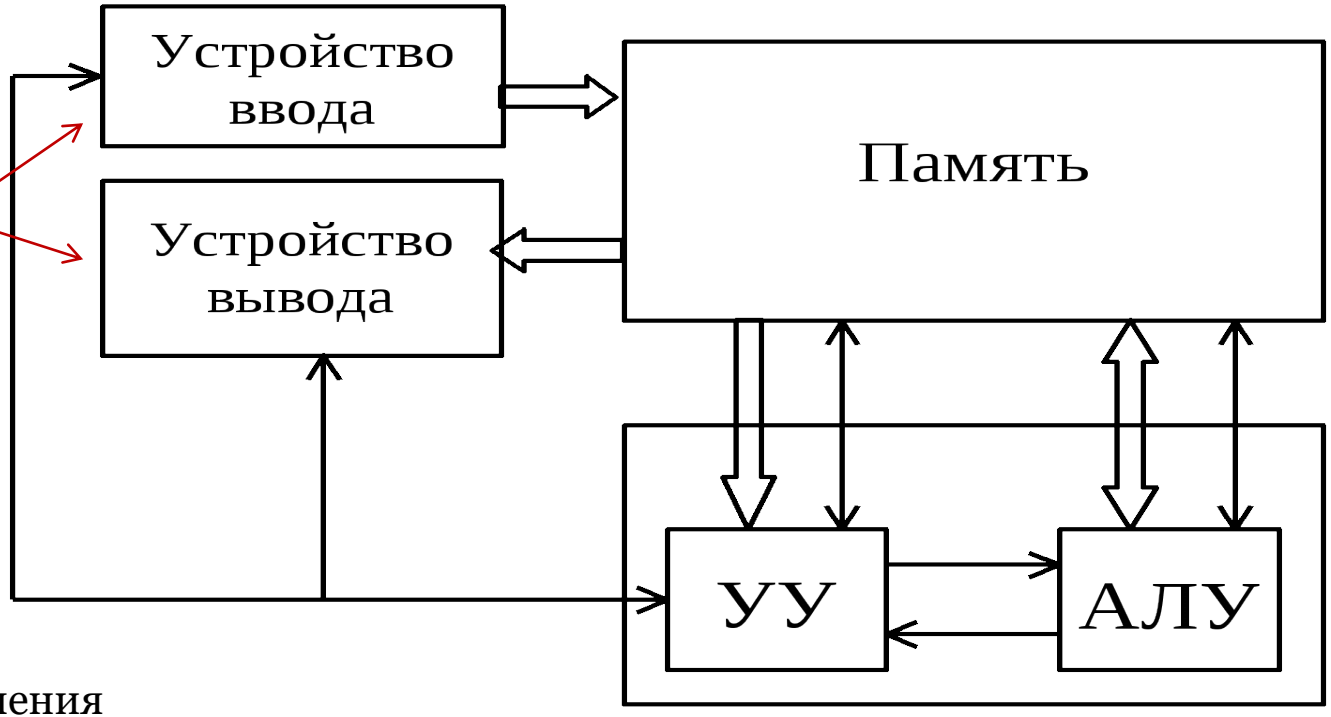
Архитектура фон Неймана

- Джон фон Нейман, 1945

- Принципы:
  - Однородность памяти
    - команды и данные хранятся в общей памяти (нет привязки команд к устройству)
  - Адреса
    - Память – пронумерованные ячейки
  - Программное управление
    - Программа – последовательность команд
  - Двоичное кодирование

Архитектура фон Неймана

Не только ввод/вывод, а любые внешние устройства, работающие с данными (платы, жесткий диск, порты, устройства ввода/вывода и т.д.)

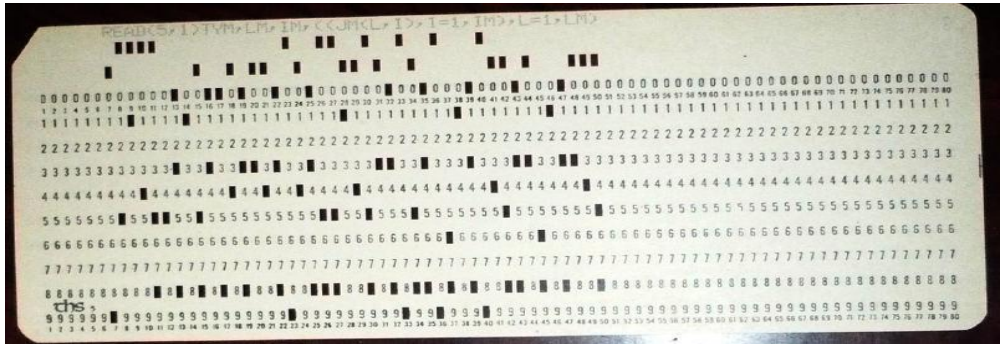


УУ – устройство управления
АЛУ – арифметико-логическое устройство
Память – оперативная память

Машинные коды и ассемблер

- Ассемблер – языки низкого уровня (детализация на уровне процессора)
- Пример: вывести 10 букв A.

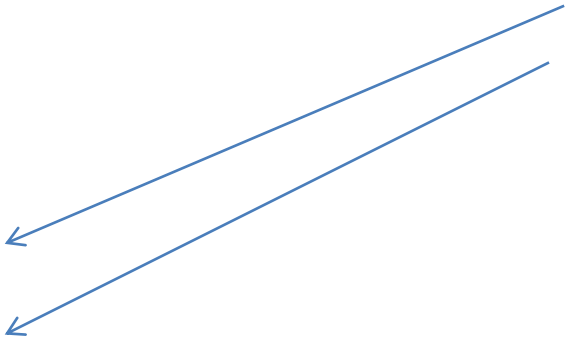
```
_main:
mov $10, %cl
loopy:
push $65
call _putchar
dec %cl
cmp $0, %cl
jne loopy
```



Разберем алгоритм

В ассемблере нет циклов. Но есть метки.

```
_main:
mov $10, %cl
loopy:
push $65
call _putchar
dec %cl
cmp $0, %cl
jne loopy
```



Сравнить с нулем
Jump not equals (если не равно)
Прыгаем на loopy
что-то напоминает?

Языки высокого уровня

- Уход от адресов, регистров и операций «переноса» (низкоуровневых операций),
- Переход к понятию «переменная».
- Возможность использовать условия и циклы.
- Акцент на обработке данных:
  - Алгоритм обрабатывает цифровые данные (цифры и числа), значит он должен уметь что-то делать с числами – а это математические операции
- Первый язык высокого уровня – Fortran (1957)

Что должен включать в себя язык программирования

- Синтаксис языка (его *грамматика*) – как писать на нем правильные программы

- Программа для компьютера, которая умеет превращать правильный код на языке программирования (текстовый файл, вообще говоря) в машинный код
  - *Еще раз: .pas, .cpp, .cs, .java – это все текстовые файлы*

Компилятор / интерпретатор

- **Компилятор** – выполняет целиком трансляцию программы из языка высокого уровня в машинный код
  - Сразу всю программу
  - Дальше вы можете запустить этот машинный код (например, .exe)
- **Интерпретатор** – выполняет построчную (покомандную) трансляцию и выполнение
  - Например, Python, PHP – интерпретаторы
- **Java** – компилируемый язык, но превращается в т.н. байт-код, который запускается виртуальной машиной Java (.class-файлы).
  - В литературе называется **интерпретатором компилирующего типа**

В 60-е годы XX века языки высокого уровня…

~~…не забывали корни~~

…были тесно связаны с математическим аппаратом:

- Программа = алгоритм, алгоритм = это способ вычисления функции
- Значит, программа ведет себя как функция
  - преобразует вход-аргумент в выход-значение.
- Значит, можно проверять правильность программы, проверяя свойства функции, которую она реализует
  - область определения, область значения т.д.

Пример на Pascal

```
read(x, y);
z := x / y;
write(z);
```

Корректность работы программы напрямую связана со свойствами функции f(x, y) = x / y

Верификация (тогда, в те времена) – формальная (математическая) проверка правильности программы

Более того

- Каждая программа – сложная функция, т.к. представляет набор команд, каждая из которых тоже является функцией:

```
begin
    read(x);
    x := x * x;                 // отработала функция f(t) = t * t
    x := x + 100500;            // отработала функция g(t) = t + 100500   (на значении f(x)
    write(x);
end
```

В итоге программа представляет собой сложную функцию

$$F(x) = g(f(x))$$

иногда ее также обозначают как $F = g \circ f$

Более того

```
begin
    x := x * x;
    x := x + 100500
end;
```

; - не просто разделитель команд, а **оператор сложной функции**.

- Именно поэтому его не нужно было ставить перед end (значение последней функции никуда уже не подставлялось)
- А вот после end – обязательно (т.к. весь блок мог быть вставлен в другой блок целиком, становясь кусочком сложной функции)

1970е

- Начало появления ПК
  - Какая ПР?

- Проникновение компьютеров в бизнес и общество – происходит разрыв между математическим понятием алгоритма и разрабатываемыми приложениями.
  - Возможна ли формальная верификация Word-а? Или Quake? Или Linux?

Язык C

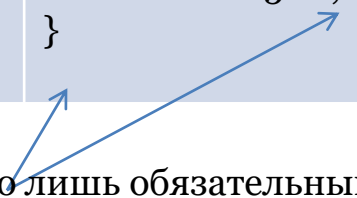
Денис Ритчи, Кен Томпсон, 1973

Оказал огромнейшее влияние на всю дальнейшую разработку.

Создавался программистами для программистов, чтобы быстро и легко разрабатывать новые приложения.

Pascal	C

begin
x := x * x;
x := x + 100500
end;
	{
	x = x * x;
	x = x + 100500;
	}



Обратите внимание, точка с запятой теперь – всего лишь обязательный разделитель команд.

Что принес C?

- Работу с памятью через указатели
- Библиотека языка
- Namespaces
- struct, union
- Активное использование препроцессора

C++

1980е, Бьерн Страуструп

Си с возможностями объектно-ориентированной разработки
+ еще много интересного.

- *Р.Керниган: «Си — инструмент, острый, как бритва: с его помощью можно создать и элегантную программу, и кровавое месиво»*
- C++ является таким еще в большей степени (т.к. стал применяться в бизнес-разработке).

Java

- “Oak”, James Gosling
- 1996 Java 1.0, сейчас Java 8 (1.8) (2014)

- Объектно-ориентированный, императивный, кросс-платформенный, C-образный синтаксис

- Применение:
  - Корпоративные приложения
  - Клиент-серверные приложения
  - Мобильные устройства (застали телефоны с Java ME)?

JVM

- Java Virtual Machine – виртуальная машина Java, реализующая кросс-платформенность для любого приложения на Java
  - “Write once, run everywhere”

- Java-приложение транслируется в байт код, который потом выполняется JVM (Just-In-Time)

Какие есть джавы?

- **Java Micro Edition (ME)** – для мобильных устройств
- **Java Standard Edition (SE)** – это классическая Java
- **Java Enterprise Edition (EE)** – разработка корпоративных клиент-серверных приложений
- **Java Card** – для смарт-карт и других устройств с очень ограниченным объемом памяти и вычислительными ресурсами.

Где Java?

- В продуктах Oracle (владеет Java)

- Amazon, eBay, LinkedIn, Yahoo
- Практически все Android-приложения
- Системы e-Commerce
    - Банки часто держат отделы квалифицированных java-разработчиков
- *и т.д.*

Чтобы все работало

- Минимум – Java Development Kit (JDK)
- Для выполнения байт кода – Java Runtime Environment (JRE)

Утилиты JDK:

- **javac** – компилятор, компилирует .java файлы в .class-файлы (байт код)
- **java** – запуск JVM, которая выполняет .class файлы.

Вопросы для самоконтроля

- Что такое информация?
- Какие действия возможны с информацией?
- Что такое информационный процесс, информация система, информационные технологии? Приведите пример (не из лекции).
- Какие представления информации вы знаете? Приведите пример, как представление закодировать в цифровом виде.
- Что есть управление в информационных системах?
- Приведите пример алгоритма из математики, из информатики (не из лекции)
- Какие способы представления алгоритма вы знаете?
- Что такое машина Тьюринга? Как машина Тьюринга задает функцию? Как устроена машина Тьюринга, как она работает?
- Приведите пример машины Тьюринга, вычисляющую какую-нибудь функцию (не из лекции).
- Что такое тезис Черча-Тьюринга? Что такое Тьюринг-полнота?
- Что такое универсальная машина Тьюринга? В чем ее важность для развития информационных технологий?
- *Продолжение на следующем слайде…*

Вопросы для самоконтроля

- Что такое RAM-машина? Опишите принцип ее работы?
- Опишите принципы архитектуры фон Неймана
- Опишите устройства архитектуры фон Неймана
- Что относится к языкам программирования низкого уровня?
- Как организован цикл в языке ассемблер?
- Чем отличаются языки высокого и низкого уровня? Приведите примеры языков высокого уровня.
- Что такое компилятор, интерпретатор? Приведите примеры компилируемых и интерпретируемых языков.
- Поясните, что означает "программа по сути представляет собой (сложную) функцию".
- К каким типам языков программирования относится Java?
- Что такое Java Virtual Machine? Как происходит процесс компилирования и запуска программы на Java?
- В чем разница между Java ME/SE/EE?
- Что такое JRE и JDK, чем они друг от друга отличаются?

Литература по курсу *(первый семестр)*

- Ильдар Хабибуллин «Технология Java»
  - *и другие его книги по Java.*
- Брюс Эккель «Философия Java»
- Герберт Шилдт «Полный справочник по Java»
- Джошуа Блох «Java – эффективное программирование»
- Гради Буч «Объектно-ориентированный анализ и проектирование»
  - *Примеры на C++, но одна из лучших книг по ОО-парадигме.*
- Дональд Кнут «Искусство программирования на ЭВМ», т.1
  - *Классика жанра по алгоритмизации.*

Ссылки

- Javarush.ru – интерактивное онлайн обучение на Java
- www.lektorium.tv/course/22896 - видеолекции по Java (на этом ресурсе есть и другие видео лекции)
- http://www.quizful.net/category/java - тесты на знание Java
- *и многие другие*

Погуглить самостоятельно

! Определение новых информационных технологий

! Гарвардская архитектура (альтернативная фон Нейману)