01. ORIGINS

курс лекций по информатике и программированию
для студентов первого курса ИТИС КФУ (java-поток)
2023/2024

М.М. Абрамский

кандидат технических наук, доцент кафедры программной инженерии

Формальности

- 18 лекций на 2 семестра
- В январе и июне – письменный экзамен.
- Контактная почта: mma@kpfu.ru

+ презентации все высылаю.

– лекции в онлайн не дублирую.

Задачи предмета

- «Взрослый» взгляд на знакомые темы,
- Новая картина мира с точки зрения данных, алгоритмов и сущностей,
- 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 смерджил и на продакшн залил, но в итоге деплой упал, при билде джарники на боевом серваке не нашлись»

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)
- Кодирование
- Передача
- Обработка
- Хранение
- Декодирование
- Отображение (воспроизведение)

- **Информационный процесс** – последовательность действий с информацией
- **Информационная система** – система, где реализованы информационные процессы.

Информационный процесс	***Информационная система***
Регистрация в приложении (*сбор, преобразование, передача, хранение*)	Приложение
Лайк фото в ВК (*сбор, передача, хранение, обработка, воспроизведение*)	Мобильное приложение / Сайт

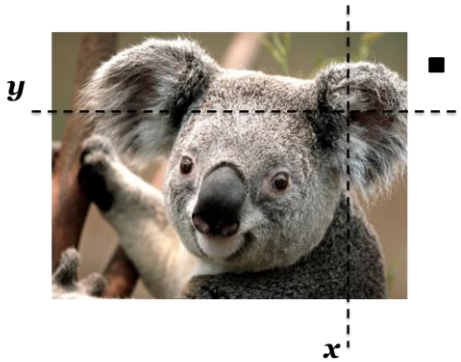
Информация не простая…

… а **цифровая (digital)** – может быть закодирована в виде последовательности цифр (и раскодирована обратно).

Текст, изображение, звук, видео – *все это представимо в виде цифровом виде*

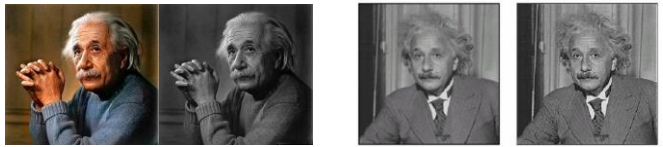
Символ – Каждый символ каждого алфавита каждого языка в мире имеет свой собственный номер (например, 65 – заглавная буква А). В памяти устройства хранится число (65), а на экране мы видим А.

Изображение



Цвет (R, G, B) – три числа от 0 до 255

Обработка изображений – операции над этими числами



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

Цифровые (информационные) технологии

Технологии работы с [цифровой] информацией («работы» – в смысле реализации информационных процессов)

Иоганн Бекманн
1772
Придумал термин «технология»

Who coined the term information technology?
1958 Harvard Business Review,
Harold J. Leavitt and Thomas C. Whisler
“the new technology does not yet have a single established name. We shall call it Information Technology.”

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.
- Значит, в информационных системах есть **управление.**

Это управление – не менеджмент, не контроль, не руководство (не «imperio»)

Кибернетика

Κυβερνητική (греч.) – искусство управления

- 1843 г. – Андре-Мари Ампер – наука об управлении государством
- 1948 г. – Норберт Винер – наука о закономерностях процессов управления и работе с информацией

Нужно организовывать управление информационными процессами в информационных системах.

Кибернетика – не только в компьютерах

- κυβερνητ (греч. «кибернит») – губернатор
- т.е. кибернетика, киборг, government, губернатор, гувернантка – родственные слова

Алгоритм

Способ реализации управления

Al Khorezmi (IX век)

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

Последовательность шагов

Алгоритмы везде и всегда

Мы впервые познакомились с ними в математике (хотя явно и не упоминали как алгоритмы):

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

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

НЕМНОГО О МАТЕМАТИКЕ

Где начинается математика?

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?

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)**?

Да, 25, но как Вы это узнали?

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.

Когда что изобрели в математике

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

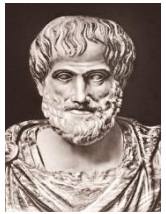
Евклид:
Через точку, не лежащую на данной прямой, можно провести лишь одну прямую, параллельную данной.

Все (много веков подряд):
Верим Евклиду!

Н.И. Лобачевский (и еще несколько человек)
Чет не убедительно! Поэтому давайте наоборот: 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?

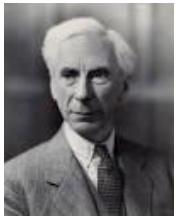
Незадача

- 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

МАТЕМАТИКА КОНЦА XIX – НАЧАЛА XX

ПОПЫТКИ ПОСТРОИТЬ КОНСТРУКТИВНУЮ МАТЕМАТИКУ

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».

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:

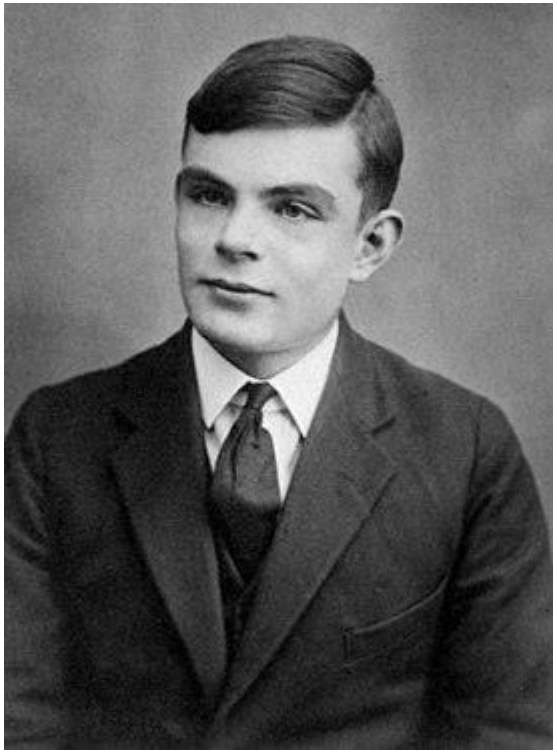
- Рекурсивные функции
- Лямбда-выражения Чёрча
- Нормальные алгоритмы Маркова
- …
- **Машина Тьюринга**

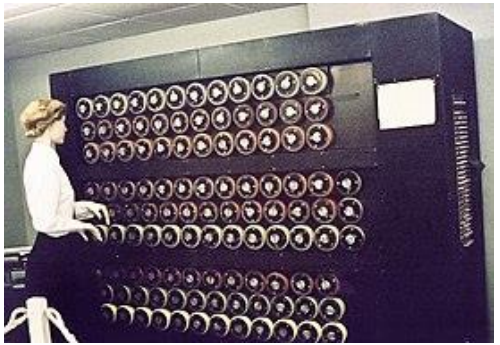
Алонзо Чёрч

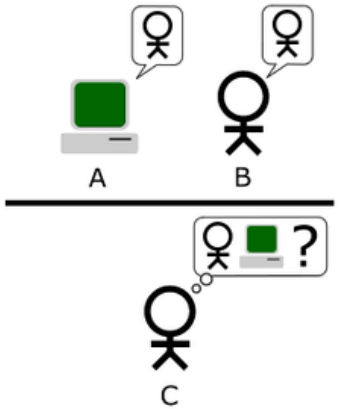
Андрей Марков

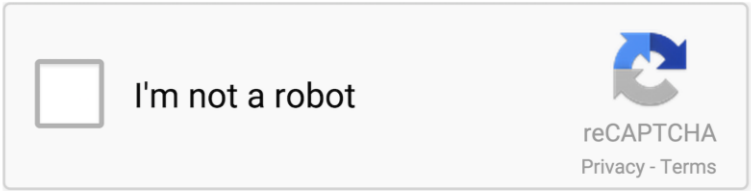
Алан Тьюринг

Алан Тьюринг (1912 – 1954)









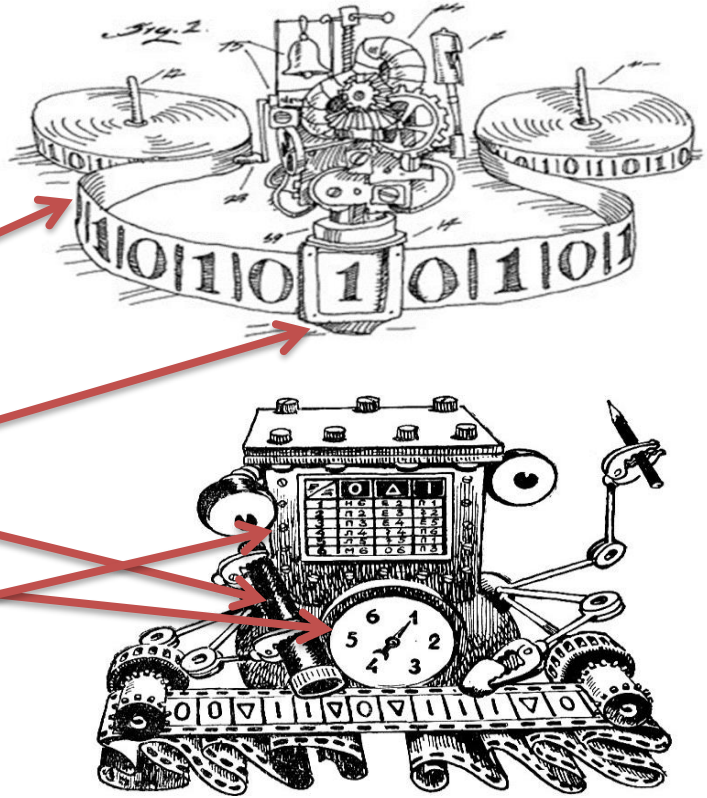
COMPLETELY AUTOMATED PUBLIC TURING TEST TO TELL COMPUTERS AND HUMANS APART (**CAPTCHA**)

Машина Тьюринга (МТ)

Абстрактная модель вычислительного устройства

Устройство (простым языком):

- Лента (*бесконечная*) с символами из *алфавита*
- Считывающая / записывающая головка,
- Состояния (память)
- Программа

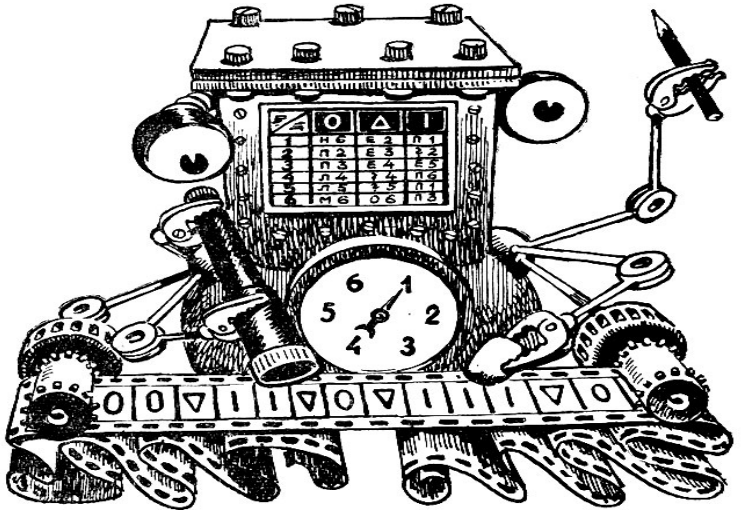


Программа МТ

и есть алгоритм ее работы.

Пример: *программа МТ для функции* $f(x) = x + 1$.

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



Работа Машины Тьюринга

$f(x) = x + 1$

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

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.

				s1						
Шаг 1	...	^	^	1	0	1	^	^	...	
					s1					
Шаг 2	...	^	^	1	0	1	^	^	...	
						s1				
Шаг 3	...	^	^	1	0	1	^	^	....	
							s1			
Шаг 4	...	^	^	1	0	1	^	^	...	
						s2				
Шаг 5	...	^	^	1	0	1	^	^	...	
					s2					
Шаг 6	...	^	^	1	0	0	^	^	...	
					stop					
Шаг 7	...	^	^	1	1	0	^	^	...	

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 (1936, 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**!

Ограничения

Не все задачи, которые можно сформулировать, имеют алгоритм! *Пример:* ***проблема остановки***

- *Помните машину Тьюринга? А то, что ее программу можно закодировать? А помните, что закодированная информация – это число? Получается – у каждой МТ есть уникальный номер.*
- **f(x) = 1**, если машина Тьюринга с номером (кодом) **x** останавливается (не зацикливается) на входе **x**, а иначе **f(x) = 0.**
- ***Нет алгоритма, вычисляющего f(x)***
  - ***Доказанная теорема***

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*

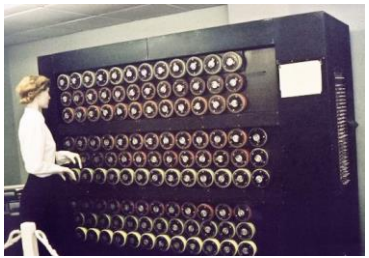
1840е годы

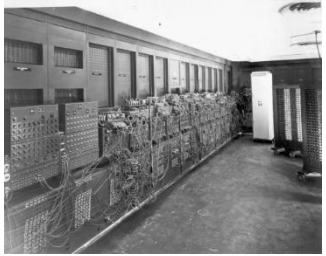
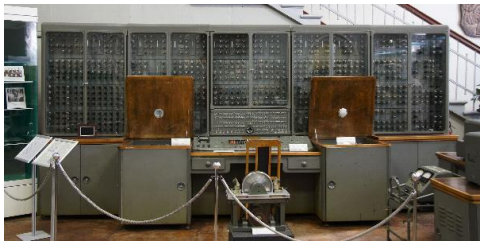
Ада Лавлейс написала первый алгоритм для аналитической машины Бэббиджа

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

Технологии XX века

- 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



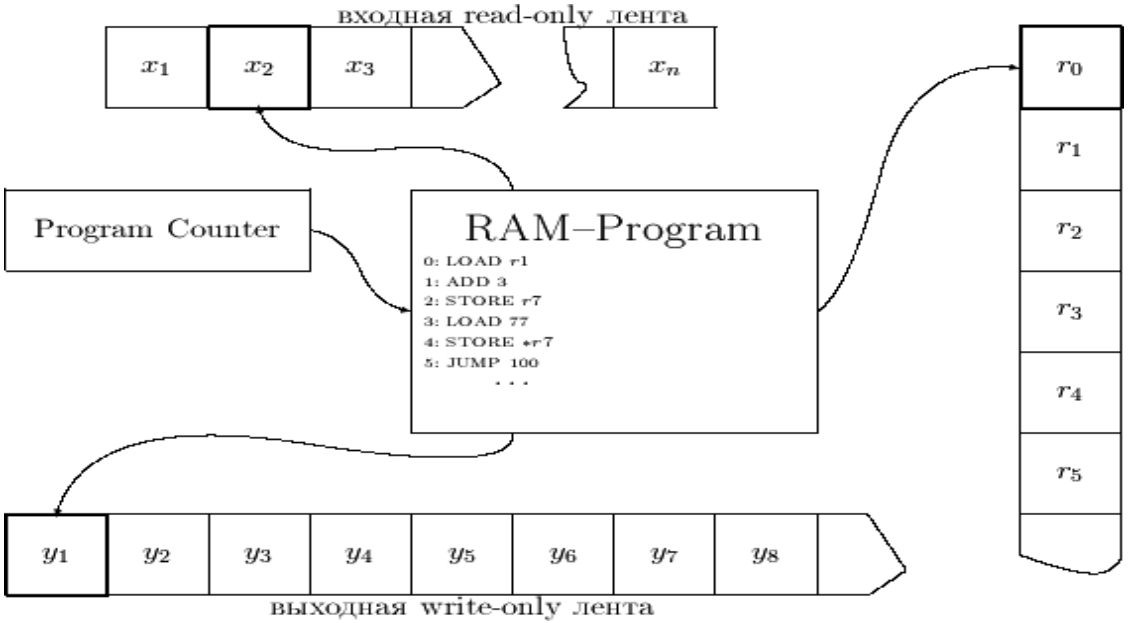
Обобщим

1. Кризис математики привел к формализации понятия алгоритм.

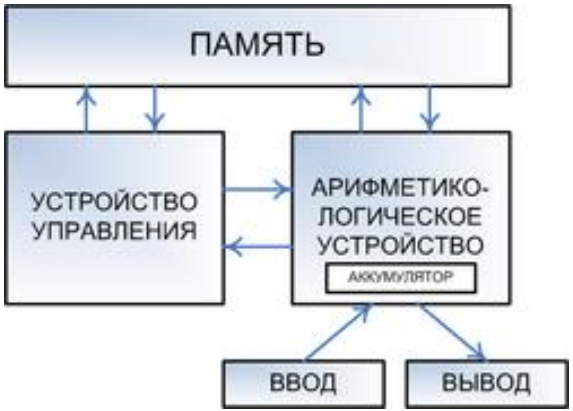
2. Появились технологии, позволяющие строить вычислительные устройства.

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

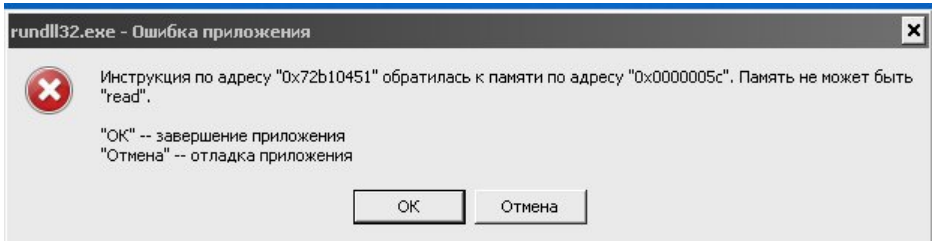
Random Access Machine



Джон фон Нейман (1903 - 1957)



1101001010
1010010101
0010101010
10101001111
0110



Принципы архитектуры фон Неймана

- Однородность памяти
  - команды и данные хранятся в общей памяти
- Адреса
  - Память – пронумерованные ячейки
- Программное управление
  - Программа – последовательность команд
- Двоичное кодирование

«Физика»

- Компьютер – физическое устройство

- Если мы всю информацию кодируем в числах, то должен быть какой-нибудь физический процесс, умеющий кодировать в себе 10 разных состояний

  (это много)

Двоичная система счисления

Двоичная – т.е. разрешены только цифры 0 и 1 (меньше 2)

Все числа – наборы 0 и 1

$1_2 = 1_{10}$

$10_2 = 2_{10}$

$11_2 = 3_{10}$

$100_2 = 4_{10}$

$101_2 = 5_{10}$

$110_2 = 6_{10}$

$111_2 = 7_{10}$

Что вообще такое число?

$$1234 =$$

$$1000 + 200 + 30 + 4 =$$

$$1 \cdot 1000 + 2 \cdot 100 + 3 \cdot 10 + 4 \cdot 1 =$$

$$1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$$

Основание системы счисления

Любое натуральное число можно разбить на сумму степеней 2

$$45_{10} = 32 + 13 = 32 + 8 + 5 =$$

$$32 + 8 + 4 + 1 = 2^5 + 2^3 + 2^2 + 2^0 =$$

$$101101_2$$

5 4 3 2 1 0

Почему программисты любят числа, равные степеням двойки

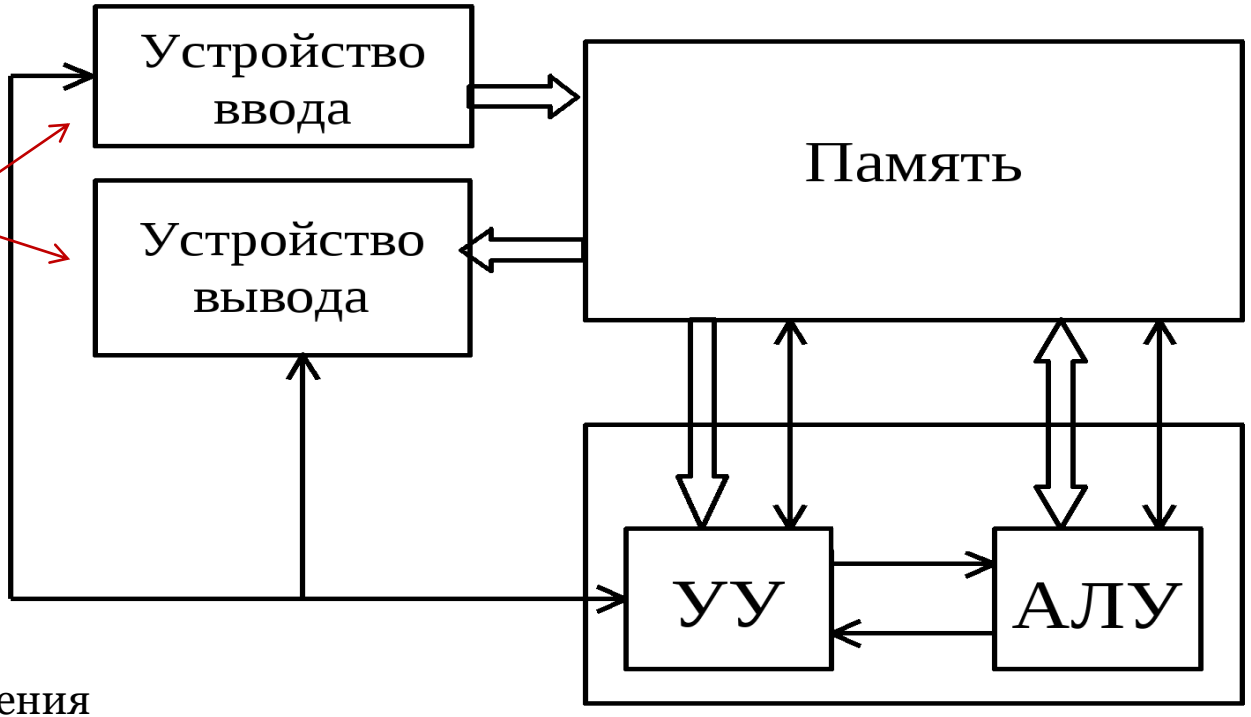
- $2^k = 100 \ldots 0$

- $4 = 2^2 = 100_2$
- $8 = 2^3 = 1000_2$
- $16 = 2^4 = 10000_2$

УПРАЖНЕНИЕ – ПОДСЧИТАТЬ ДО 32 НА ОДНОЙ РУКЕ

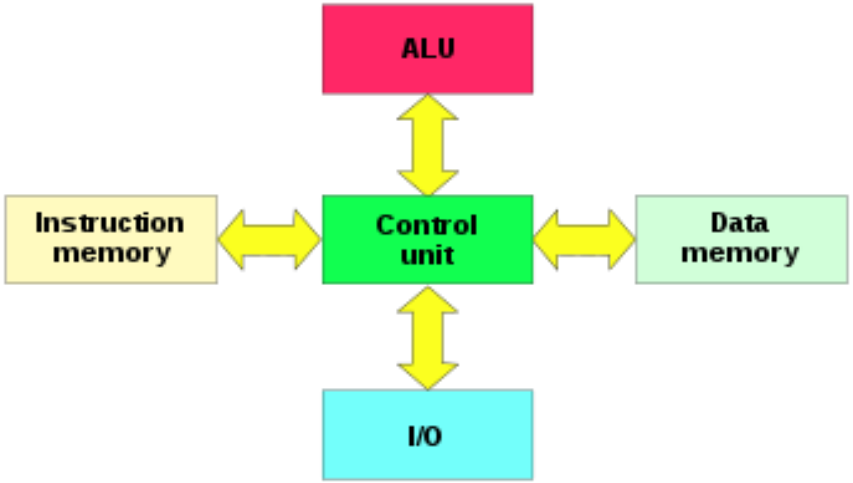
Архитектура фон Неймана

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

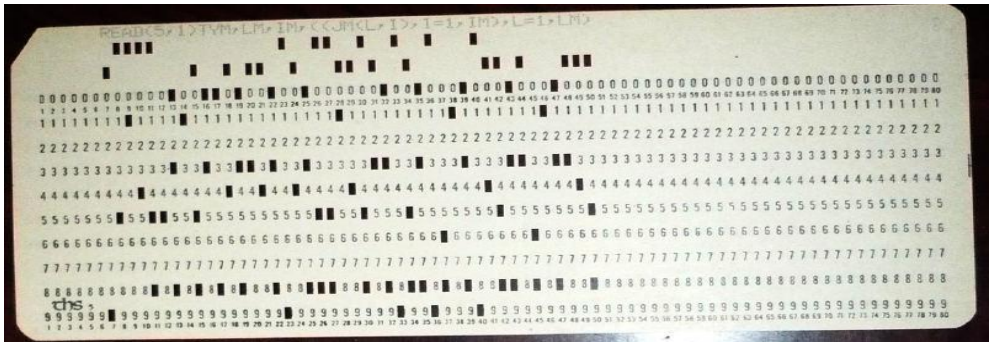
Гарвардская архитектура



Машинные коды и ассемблер

- Ассемблер – языки низкого уровня (детализация на уровне процессора)

Пример: вывести 10 букв А.



```
_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)

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

- 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

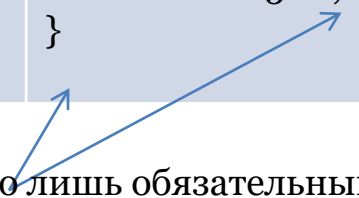
Денис Ритчи, Кен Томпсон, 1973

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 появилось в C?

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

Java

James Gosling 

1996 Java 1.0, сейчас Java 19 (2023), *ждем Java 20 в сентябре 2023*.

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

Применение:

- Корпоративные приложения (банки, CRM, распределенные системы)
- Клиент-серверные приложения
- Мобильные устройства

Что такое 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

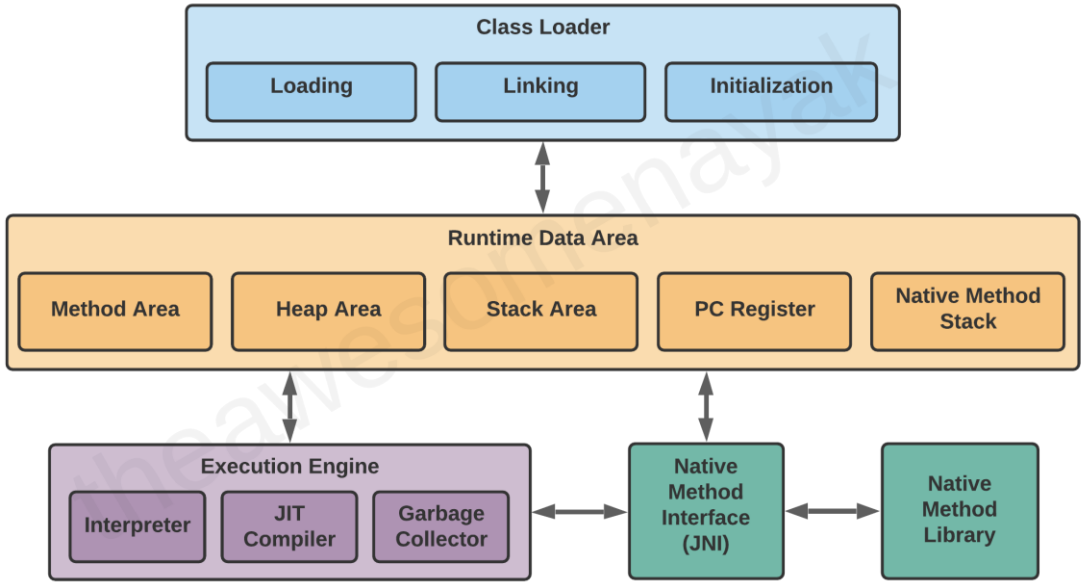
- **Компилятор** – выполняет целиком трансляцию программы из языка высокого уровня в машинный код
  - Сразу всю программу
  - Дальше вы можете запустить этот машинный код (например, .exe)
- **Интерпретатор** – выполняет построчную (покомандную) трансляцию и выполнение
  - Например, Python, PHP – интерпретаторы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

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 (*.class-файлы), который потом выполняется JVM (Just-In-Time)



https://theawesomenayak.com/jvm-tutorial


To Be Continued