

# Введение в программирование

## Лекция 1

Лопатин Александр

2015

- 1 О курсе
- 2 Вводные термины
- 3 Обзор развития вычислительных систем
- 4 Парадигмы программирования
- 5 Алгоритмическое программирование
- 6 Структурное программирование

## О чем курс

- ▶ урезанная версия одноименного универовского курса (обычно рассчитанного на 1—2 семестра)
- ▶ основы и упоминания из других курсов

# Expectations

- ▶ получим кругозор в программировании
- ▶ узнаем основы нескольких языков
- ▶ научимся базовым техникам
- ▶ на практике применим некоторые техники

## Как построен курс

- ▶ разбит так, чтобы не вызвать передоз информацией
- ▶ одно занятие в неделю на 2—3 часа (включает лекцию и практику)
- ▶ 10 недель ( $\approx 2\frac{1}{3}$  месяца)
- ▶ домашки на не более 2-х часов в неделю
- ▶ один мелкий проект на 2 недели (2 викенда)
- ▶ домашки и проект будут оцениваться

## Исполнитель

- ▶ повар
- ▶ военнослужащий
- ▶ гитарист
- ▶ коммерческая компания
- ▶ вычислительная система (компьютер, смартфон, бытовой прибор и т.п.)

## Инструкции исполнителя

- ▶ **пункты рецепта** («42. Положить ложку сахара») для повара
- ▶ **приказы** («нале-во!») для военнослужащего
- ▶ **аккорды** для гитариста
- ▶ **инструкции процессора** («сложить два числа») для вычислительной системы

# Алгоритм

Конечная последовательность **инструкций исполнителя** направленная на решение задачи

- ▶ **рецепт для повара**
- ▶ **лабораторная работа для студента**
- ▶ **текст песни для вокалиста**
- ▶ **бизнес-план коммерческой компании**
- ▶ **шаги воспроизведения проблемы для тестировщика**
- ▶ **компьютерная программа для вычислительной системы**



## С точки зрения данных

Ввод → Обработка → Вывод

# Интерфейс

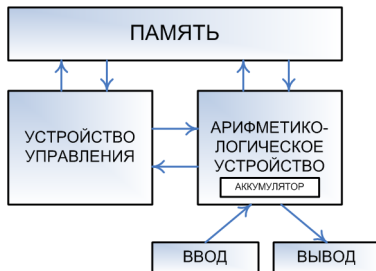
Означает «взаимодействие»

Нужно для осуществления ввода/вывода данных

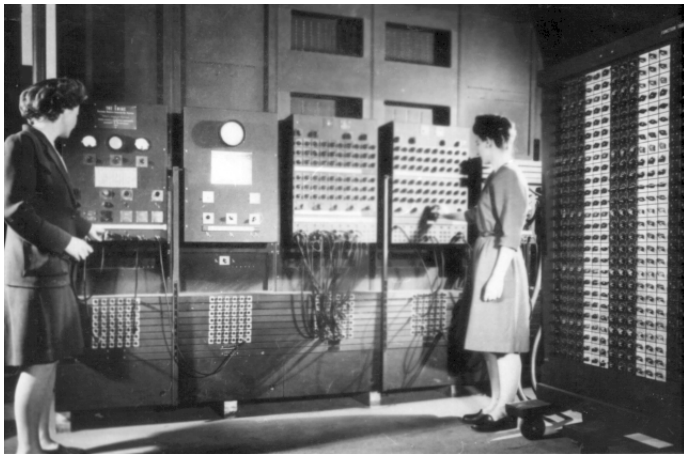
- ▶ Пользовательский (UI — User Interface)
  - ▶ Графический пользовательский интерфейс, Graphical User Interface (GUI)
  - ▶ Интерфейс командной строки, Command-Line Interface (CLI)
- ▶ Программный (API — Application Interface)
  - ▶ подпрограммы, модули, библиотеки
  - ▶ сетевые протоколы (например HTTP)
  - ▶ много других

CLI может использоваться как API

## 1945: Архитектура фон Неймана (Von Neumann architecture)



## 1946: ENIAC

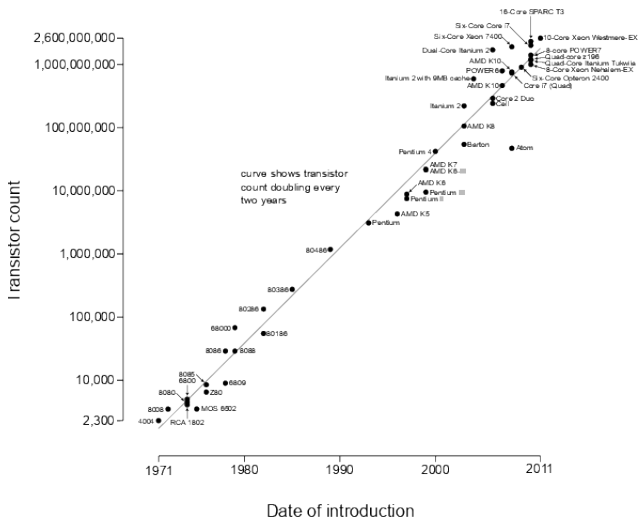


## 1964: IBM System/360



Стивен Леви — Хакеры: Герои компьютерной революции

## Microprocessor Transistor Counts 1971-2011 & Moore's Law



Закон прекратил действовать — пришло время новых идей по увеличению производительности железа

- ▶ выполнять трудоемкие операции на других устройствах (например GPU)
- ▶ объединять несколько ядер процессоров в один
- ▶ объединять несколько компьютеров в вычислительный кластер
- ▶ проектировать концептуально другие выч. системы, необязательно на основе фон Неймовской архитектуры
  - ▶ квантовые компьютеры
  - ▶ клеточные компьютеры
  - ▶ ...

С ростом производительности железа растет и сложность задач

Сложность программ тоже растет

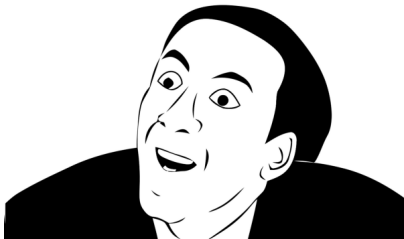
Появляется много специализаций в IT, по аналогии с врачами



## Многоуровневые системы

Системы состоят из подсистем (из слоёв / уровней)

**YOU DON'T SAY?**

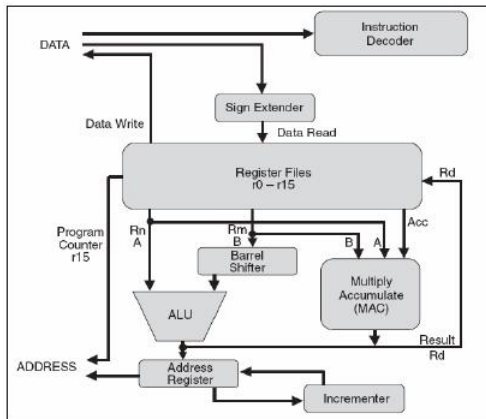


## Уровни (levels / layers) в реальной жизни

Уровни характеризуются **обязанностями** и определяют **кто над кем главнее (выше)**

- ▶ Директор конторы (высокий уровень)
  - ▶ Художник (средний уровень)
  - ▶ Программист (средний уровень)
    - ▶ Младший программист (низкий уровень)

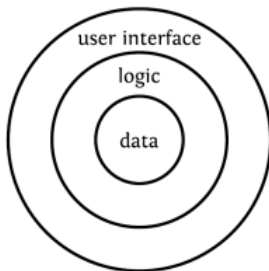
Каждая железяка состоит из множества более низкоуровневых железок



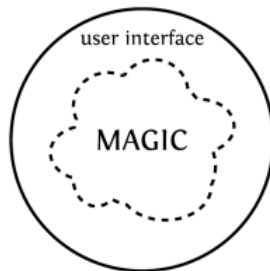
## Софт тоже многоуровневый

---

Your program



How users see it



# Введение в программирование

## └ Парадигмы программирования

Императивное

The diagram consists of two dashed rectangular boxes. The box on the left is labeled 'Императивное' (Imperative) and the box on the right is labeled 'Декларативное' (Declarative). Both boxes are empty, suggesting they are placeholders for further content or examples related to these paradigms.

Декларативное

## Императивное программирование (Imperative Programming)

Написание **алгоритмов** путём перечисления **инструкций исполнителя**



## Декларативное программирование (Declarative Programming)

**Описание желаемого результата, вместо алгоритма**  
получения этого результата.

Например запрос к базе данных:

Выбрать абитуриентов  
поступающих на специальность «Программная Инженерия»  
с сортировкой по сумме баллов по убыванию

Подробнее рассмотрим ближе к концу курса

# Введение в программирование

## └ Алгоритмическое программирование

Императивное

Алгоритмическое

Блок-схемы, словесное описание

Декларативное



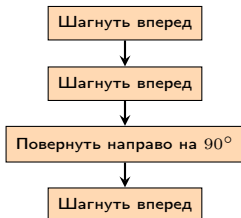
## Линейный алгоритм

1. Шагнуть вперед
2. Шагнуть вперед
3. Повернуть направо на  $90^\circ$
4. Шагнуть вперед

## Алгоритм с ветвлением

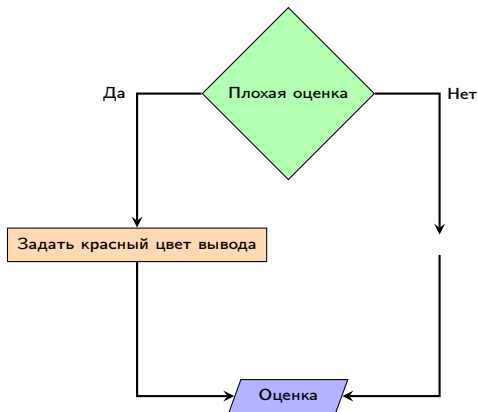
1. Если небо светлое — перейти к п. 2 иначе к п. 3
2. Напечатать «Сейчас день»
3. Перейти к п.5
4. Напечатать «Сейчас ночь»
5. Напечатать «Ваш Капитан Очевидность»

## Линейный алгоритм



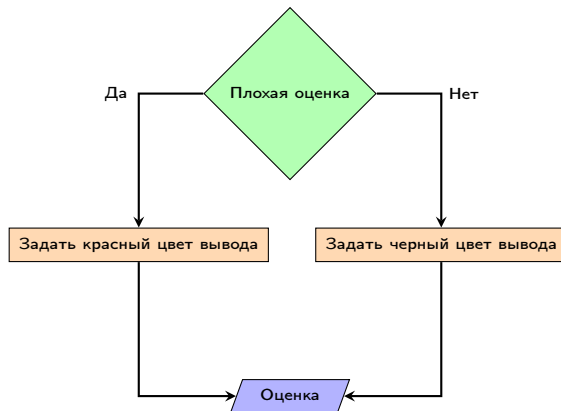
## Алгоритм с ветвлением

Бывает с одной веткой

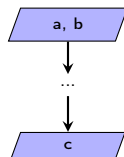


## Алгоритм с ветвлением

И с двумя ветками



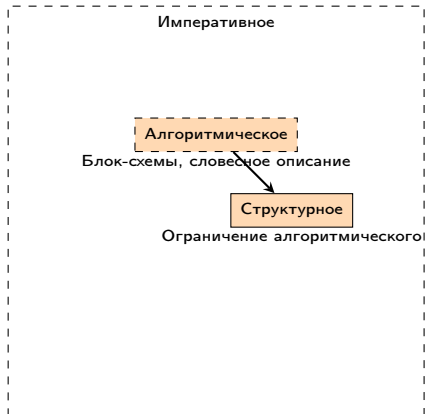
## Практика



1. Нарисовать блок-схему к программе, которая берет в качестве ввода числа  $a$  и  $b$  и решает уравнение  $a = b \cdot c$ . Вывести решение (значение  $c$ ). (Подсказка: сначала нужно вручную преобразовать уравнение)
2. Предусмотреть обработку ошибки деления на ноль к предыдущему заданию. Выводить «Ошибка: деление на ноль» вместо решения в этом случае
3. Написать словесное описание получившегося алгоритма

# Введение в программирование

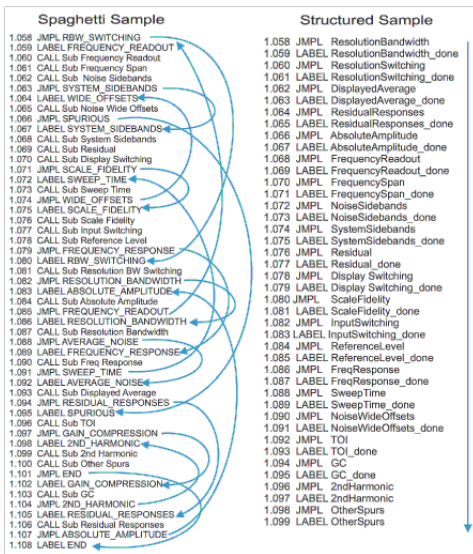
## └ Структурное программирование



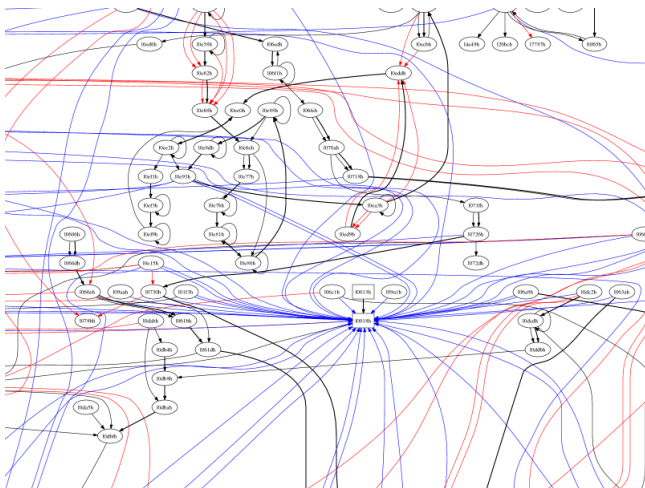
## Структурное программирование (Structured Programming)

- ▶ отказ от меток и безусловного перехода
- ▶ использование двух **управляющих структур**
  - ▶ условие
  - ▶ цикл





# Spaghetti is write-only code



## Повторения всё же нужны

Но не такие безобразные

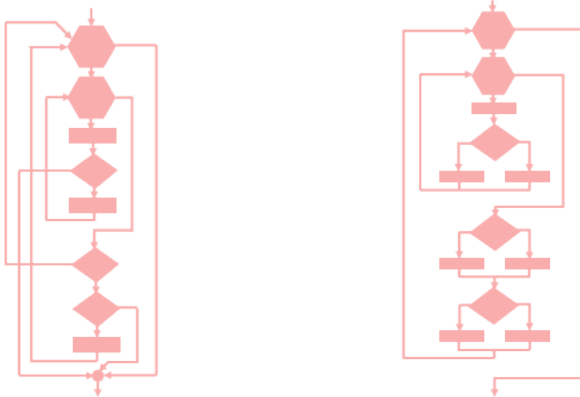
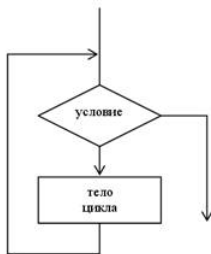
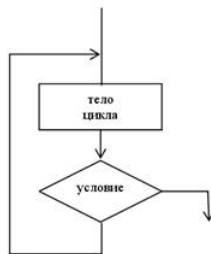


Рис.: Спагетти-код и структурный код

## Циклы



Цикл с предусловием



Цикл с постусловием

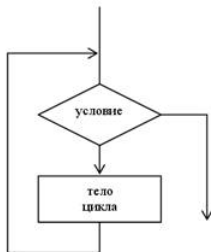
## Цикл с предусловием (while)

### 1. Пока гвоздь не забит

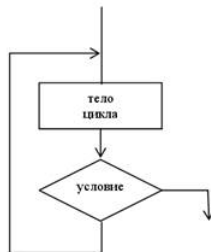
1.1 Поднять молоток

1.2 Ударить молотком по гвоздю

1.3 Перейти к п. 1



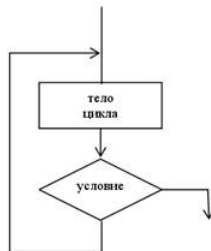
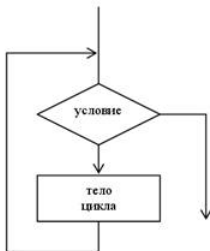
Цикл с предусловием



Цикл с постусловием

## Цикл с постусловием (do ... while)

1. (Делать)
  - 1.1 Поднять молоток
  - 1.2 Ударить молотком по гвоздю
2. Пока гвоздь не забит — перейти к п. 1



## Цикл со счетчиком (for) — частный случай цикла с предусловием

1. Инициализировать счетчик  $i$  значением 1
2. Пока  $i \leq n$ 
  - 2.1 Выполнить действие
  - 2.2 Выполнить другое действие
  - 2.3 ...
  - 2.4 Увеличить счетчик  $i$  на единицу
  - 2.5 Перейти к п. 2.

## Всё тот же цикл for

1. Для  $i$  от 1 до  $n$ 
  - 1.1 Выполнить действие
  - 1.2 Выполнить другое действие
  - 1.3 ...
  - 1.4 Перейти к п. 1.



## Циклы могут быть вложены

1.  $i = 1$
2. Пока  $i \leq \text{количество\_студентов}$
3. Напечатать имя  $i$ -го студента
  - 3.1  $j = 1$
  - 3.2 Пока  $j \leq \text{количество\_оценок\_студента}$ 
    - 3.2.1 Показать  $j$ -ю оценку  $i$ -го студента
    - 3.2.2  $j = j + 1$
    - 3.2.3 Перейти к п. 3.2
  - 3.3  $i = i + 1$
  - 3.4 Перейти к п. 2
4. Напечатать «такие дела»

## Практика

1. Нарисовать блок-схему к алгоритму забивания гвоздя  
со слайда с циклом с предусловием
2. Тоже самое для слайда с циклом с постусловием
3. Тоже самое для алгоритма с вложенными циклами  
(подсказка: можно начать с вложенного цикла)

## Домашка

Нарисовать блок-схему и написать словесное описание к алгоритму решения квадратного уравнения  $ax^2 + bx + c = 0$ .

Входные данные:  $a$ ,  $b$  и  $c$ . Выходные данные:

- ▶ либо «Корней нет»
- ▶ либо  $x_1$
- ▶ либо  $x_1$  и  $x_2$

Подсказки:

- ▶ дискриминант вычисляется по формуле  $D = b^2 - 4ac$
- ▶ корни уравнения вычисляются по формуле  $x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$
- ▶ количество корней зависит от значений дискриминанта (3 ситуации:  $D < 0$ ,  $D = 0$  и  $D > 0$ )