

# Технологии и языки программирования

## Введение

Юдинцев В. В.

Кафедра теоретической механики  
Самарский университет

14 сентября 2016 г.

# Содержание

- 1 Введение
- 2 Алгоритм
- 3 Этапы программирования
- 4 Язык программирования
- 5 Машинно-зависимые языки
- 6 Машинно-независимые языки
- 7 Поколения языков высокого уровня
- 8 Список источников

# **Введение**

# Задачи курса

- Знакомство с основными структурами данных и типовыми методами обработки этих структур
- Изучение методов структурного и объектно-ориентированного программирования
- Закрепление навыков алгоритмизации и программирования на основе изучения языка программирования Python
- Создание практической базы для изучения других учебных дисциплин (“Методы вычислений”, “Компьютерный практикум по механике”, ...)

# Содержание курса

## I семестр (зачёт)

- Развитие языков программирования, классификация
- Обзор основных технологий программирования
- Основы программирования на языке Python
- Технологии программирования на языке Python (процедурное, функциональное, ООП)

## II семестр (экзамен)

- Использование библиотек Python NumPy, SciPy для проведения научных вычислений
- Визуализация результатов: использование библиотеки matplotlib и языка gnuplot
- Построение интерфейсов

# Что такое программирование?

Программирование

Часть прикладной математики и вычислительной техники,  
разрабатывающая методы составления **программ**

Программирование

Научная и практическая деятельность по созданию **программ**<sup>1</sup>

# Программирование

Процесс преобразования сформулированной задачи в набор инструкций, включающий:

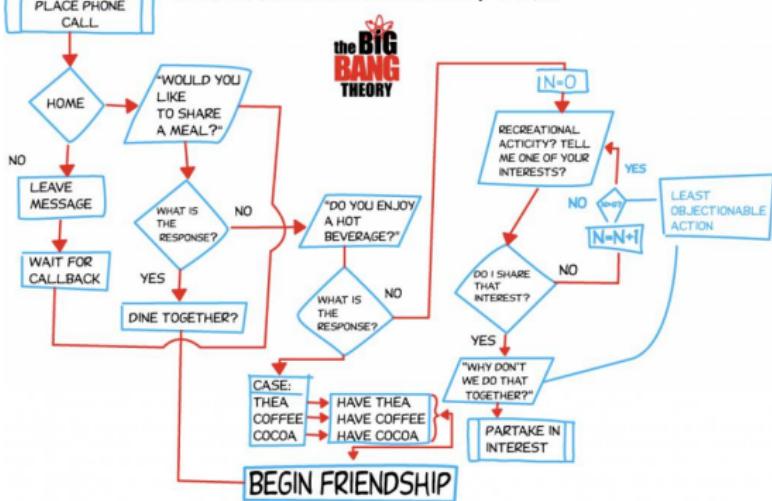
- анализ и разработку **алгоритма** решения задачи,
- оценку **корректности** и **эффективности** алгоритма,
- непосредственную **реализацию** алгоритма на языке программирования, который выбирается в зависимости от задачи, типа ЭВМ – написание программы.

# Алгоритм

# Алгоритм

## THE FRIENDSHIP ALGORITHM

DR. SHELDON COOPER , PH.D



Алгоритм – это набор инструкций, описывающих порядок действий исполнителя, для достижения некоторого результата

# Свойства алгоритма

- **Определенность и результативность**

За **конечное** число шагов должен быть получен **результат**, или доказано его отсутствие

- **Массовость**

Возможность получения результата при различных исходных данных для некоторого класса сходных **задач**

- **Формальность**

Отвлечение от содержания поставленной задачи и строгое выполнение некоторого правила, инструкции

- **Дискретность**

Возможность разбиения алгоритма на **отдельные элементарные действия**

# Формы представления алгоритма

- Словесная на неформальном языке
- Графическая (блок-схема)
- На **языках программирования**: алгоритм преобразуется в **программу**

## Блок-схема: решение квадратного уравнения

Найти действительные решения квадратного уравнения ( $a \neq 0$ ):

$$ax^2 + bx + c = 0$$

### Решение

- 1 Определить дискриминант

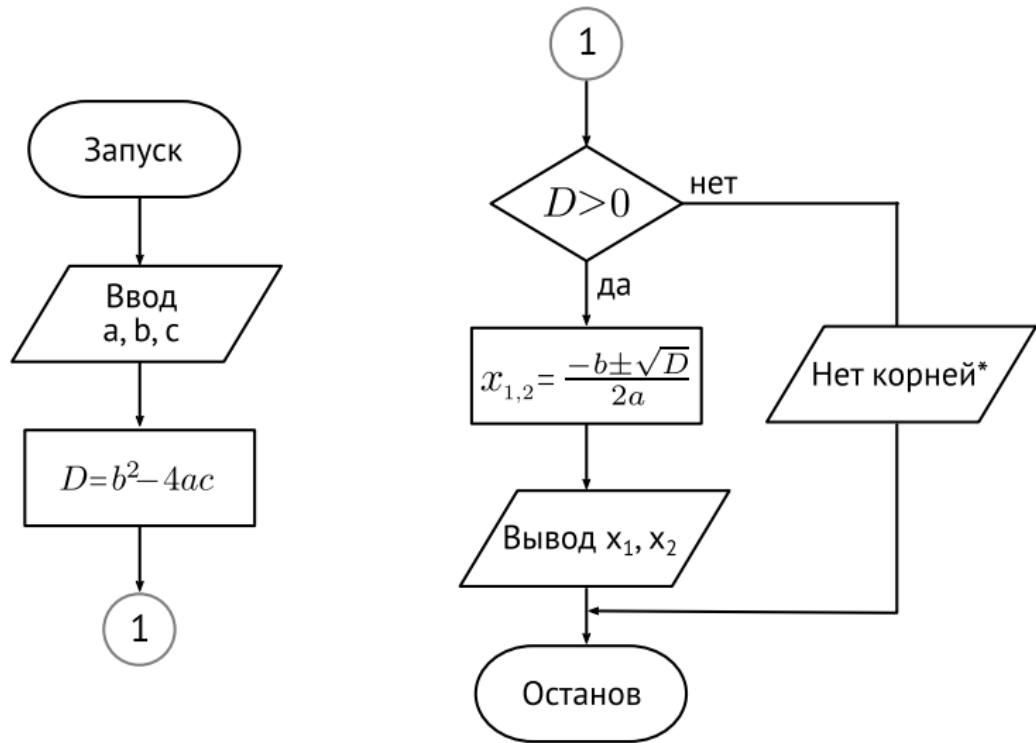
$$D = b^2 - 4ac$$

- 2 Если дискриминант меньше нуля, то действительных корней нет
- 3 Если дискриминант больше или равен нулю, то есть два разных корня ( $D > 0$ ) или два одинаковых ( $D = 0$ )

$$x_1 = \frac{-b + \sqrt{D}}{2a}, \quad x_2 = \frac{-b - \sqrt{D}}{2a}$$

# Блок-схема: решение квадратного уравнения

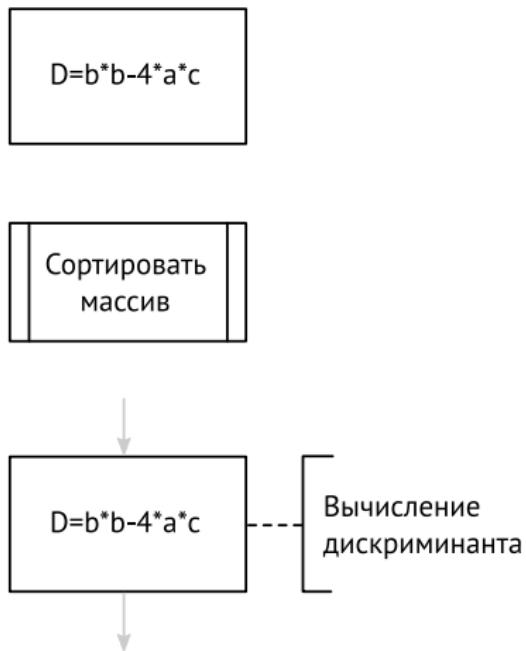
$$ax^2 + bx + c = 0, (a \neq 0)$$



## Блок-схема

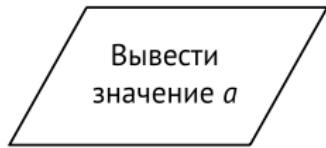
- Блок-схема – схема (графическая модель), описывающая алгоритм или процесс
- Отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими направление последовательности<sup>8</sup>
- Правила выполнения регламентируются ГОСТ 19.701-90 “Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения”

# Процесс<sup>8</sup>



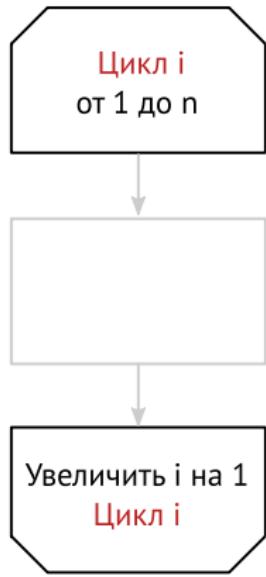
- Функция обработки данных любого вида
- Предопределённый процесс (вызов подпрограммы)
- Комментарии к процессу

# Данные<sup>8</sup>



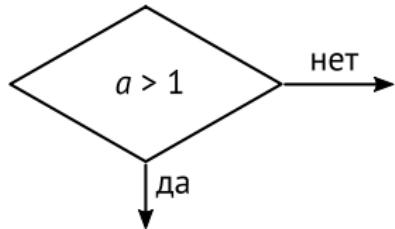
- Ввод данных
- Отображение результата

# Цикл<sup>8</sup>



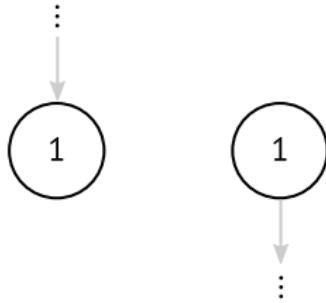
- Начало и конец цикла
- Обе части символа имеют одинаковый идентификатор

# Решение<sup>8</sup>



- Решение или функция “переключательного” типа, имеющая **один вход** и ряд альтернативных **выходов**
- Только один выход может быть активизирован после вычисления условий

## Соединитель<sup>8</sup>



- Символ отображает выход в часть схемы и вход из другой части этой схемы
- Используется для обрыва линии и продолжения ее в другом месте
- Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение

# Программа на языке Python

```
1 import math
2
3 print("Введите коэффициенты для уравнения (ax^2 + bx + c = 0):")
4
5 a = float(input("a="))
6 b = float(input("b="))
7 c = float(input("c="))
8
9 discr=b**2-4*a*c
10
11 if discr>0:
12     x1 = (-b+math.sqrt(discr))/(2*a)
13     x2 = (-b-math.sqrt(discr))/(2*a)
14     print("x1 = %.2f \nx2 = %.2f" % (x1, x2))
15 else:
16     print("Корней нет")
17
```

# Этапы программирования

# Этапы программирования<sup>3</sup>

- Постановка задачи на естественном языке
- Анализ задачи и моделирование
- Разработка алгоритма
- Программирование (кодирование)
- Тестирование и отладка
- Сопровождение

# Постановка задачи на естественном языке



- сбор информации о задаче;
- формулировка условия задачи;
- определение конечных целей решения задачи;
- определение формы выдачи результатов;
- описание данных (типы, диапазоны, ...).

# Анализ задачи и моделирование



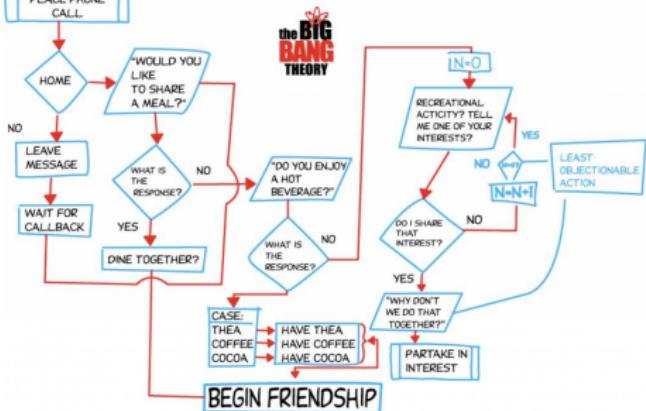
- анализ существующих аналогов;
- анализ технических и программных средств;
- разработка математической модели;
- разработка структур данных.

Источник: [www.wikihow.com](http://www.wikihow.com)

# Разработка алгоритма

## THE FRIENDSHIP ALGORITHM

DR. SHELDON COOPER , PH.D



- выбор метода проектирования алгоритма;
- выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
- выбор тестов и метода тестирования;
- проектирование алгоритма.

# Программирование



- выбор языка программирования;
- уточнение способов организации данных;
- запись алгоритма на выбранном языке программирования.

Источник: <https://commons.wikimedia.org>

# Тестирование и отладка



- синтаксическая отладка;
- отладка семантики и логической структуры;
- тестовые расчеты и анализ результатов тестирования;
- анализ результатов решения задачи, уточнение математической модели, алгоритма, программы.

Источник: <https://www.pixafy.com>

# Сопровождение программы



- составление документации к решенной задаче, математической модели, алгоритму, программе, набору тестов, использованию;
- доработка программы для решения конкретных задач.

Источник: <https://github.com/jtleek/rpackages>

# Программа

## Программа

Данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма<sup>1</sup>

## Программное обеспечение

Совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ

# Программирование

Программирование требует знания

- предметной области для которой пишется программа,
- формальной логики,
- специализированных **алгоритмов**.

# **Язык программирования**

# Языки программирования

## Язык программирования

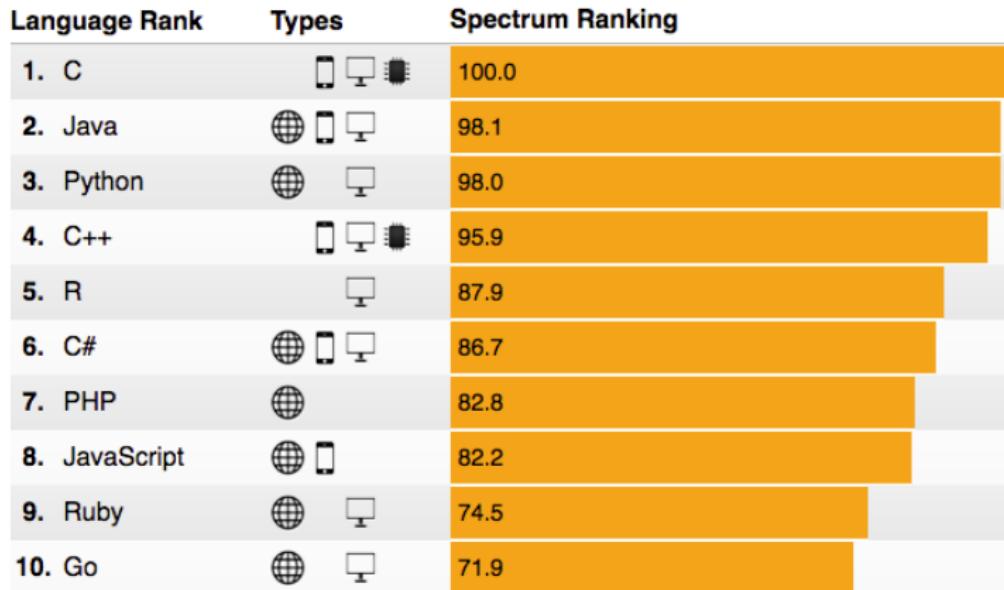
формальная знаковая система, при помощи которой записываются компьютерные программы<sup>2</sup>

- Для языков программирования однозначно определены **синтаксис и семантика**.
- Описание синтаксиса языка включает определение алфавита и правил построения различных конструкций языка из символов алфавита.

# Литералы. Идентификаторы. Ключевые слова

- Символы алфавита языка образуют **лексемы** – минимальные единицы языка, имеющие самостоятельный смысл<sup>10</sup>
- Лексемы** формируют базовый словарь языка, понятный транслятору
  - ключевые слова: `False, class, return, is, else, ...`
  - идентификаторы – имена переменных, констант, массивов, функций
  - литералы – неизменяемые величины (строки текста, числа)
  - операции: `+ - * ** & < > <= ...`
  - знаки пунктуации (разделители): `( ) [ ] , : .`

# Языки программирования



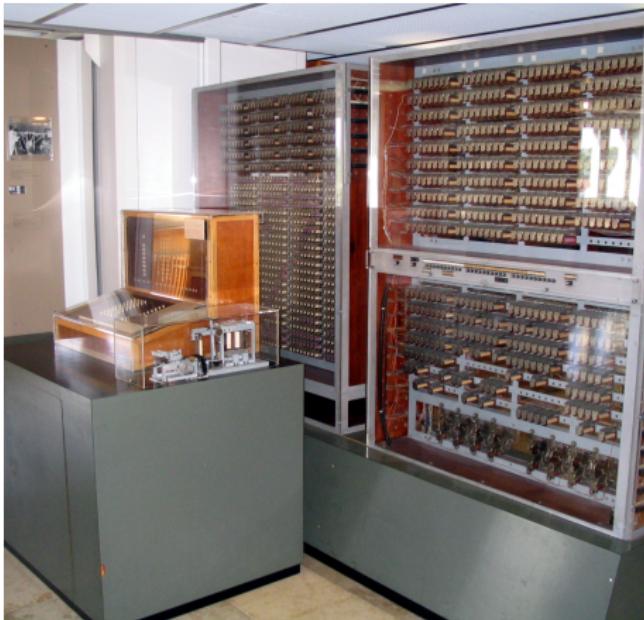
Источник: <http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016>

# Классификация языков программирования

- Машино-зависимые языки
- Машино-независимые языки

# **Машинно-зависимые языки**

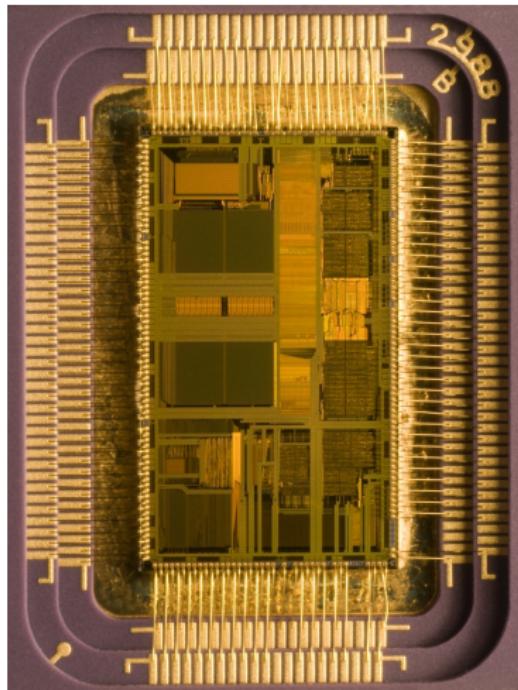
# Z3 – Первая вычислительная машина



Источник: Wikimedia.org

- Конрад Цузе, 1941 г, Германия
- Первая программируемая в двоичном коде с плавающей точкой рабочая вычислительная машина
- Машина построена на 2600 телефонных реле
- Для хранения программ использовалась перфорированная лента
- Язык программирования **Plankalkul** (Исчисление планов)

# Машинные коды



- Память ЭВМ может хранить лишь **двоичные представления**
- Программы для первых ЭВМ вводились в память с пульта, перемычками или тумблерами
- Программы писали в **машинных кодах**
- **Машинные коды – первое поколение языков программирования, машинно-зависимых языков**

Источник: Wikimedia.org

# Машинный код

```
BB 11 01 B9 0D 00 B4  
0E 8A 07 43 CD 10 E2  
F9 CD 20 48 65 6C 6C  
6F 2C 20 57 6F 72 6C  
64 21
```

Код программы  
“Hello World!”

- Машинный код – система команд (набор кодов операций) конкретной вычислительной машины, которая интерпретируется непосредственно процессором этой вычислительной машины
- Машинный код может исполняться ещё более низкоуровневым слоем программ (процессора), называемых микропрограммами

# Машинно-зависимые языки программирования

- Языки программирования, учитывающий структуру и характеристики ЭВМ определенного типа
- Имеют непосредственный доступ к аппаратным средствам ЭВМ

# Машинно-ориентированные языки<sup>6</sup>

Мнемокод (язык символьического кодирования, язык ассемблера)

- Коды операций, адреса ячеек памяти заменяются буквами или буквенно-цифровыми обозначениями
- **Программа-транслятор (ассемблер)** переводит программу на языке ассемблера в исполняимый машинный код
- Применяется для написания программ, явно использующих специфику конкретной аппаратуры

# Язык ассемблера

```
1 .MODEL SMALL
2 .DATA
3     msg DB 'Hello World',13,10,'$'
4 .CODE
5 START:
6     mov ax, @DATA
7     mov ds, ax
8     mov ax, 0900h
9     lea dx, msg
10    int 21h
11    mov ax, 4C00h
12    int 21h
13 END START
```

# Машинно-ориентированные языки<sup>6</sup>

## Макроязык

- Содержит макрокоманды, не имеющие аналогов в машинном языке
- При переводе на машинный язык каждая макрокоманда заменяется группой команд машинного языка
- Ассемблер относится к языкам **второго поколения**

# Язык программируемого калькулятора МК-61



## Программа “Лунолёт-1”

	0	1	2	3	4	5	6	7	8	9
0	Пх1	С/П	Пх3	С/П	Пх2	С/П	хП4	1	0	х
10	Пх5	÷	Пх6	-	хП7	Пх8	х	Пх3	+	хП9
20	Fx <sup>2</sup>	В↑	Пх3	Fx <sup>2</sup>	-	Пх7	÷	2	÷	Пх1
30	+	хП11	Пх9	хП13	Пх2	Пх4	Пх8	-	хП12	K x
40	2	-	Fx<0	0	Пх3	K x	5	-	Fx≥0	54
50	6	6	6	С/П	7	7	7	С/П		

Цель игры: посадить корабль на поверхность Луны со скоростью не больше 5 м/с. Управление осуществляется расходом топлива в единицу времени, определяющим силу тяги двигателя. Количество топлива ограничено.

# **Машино-независимые языки**

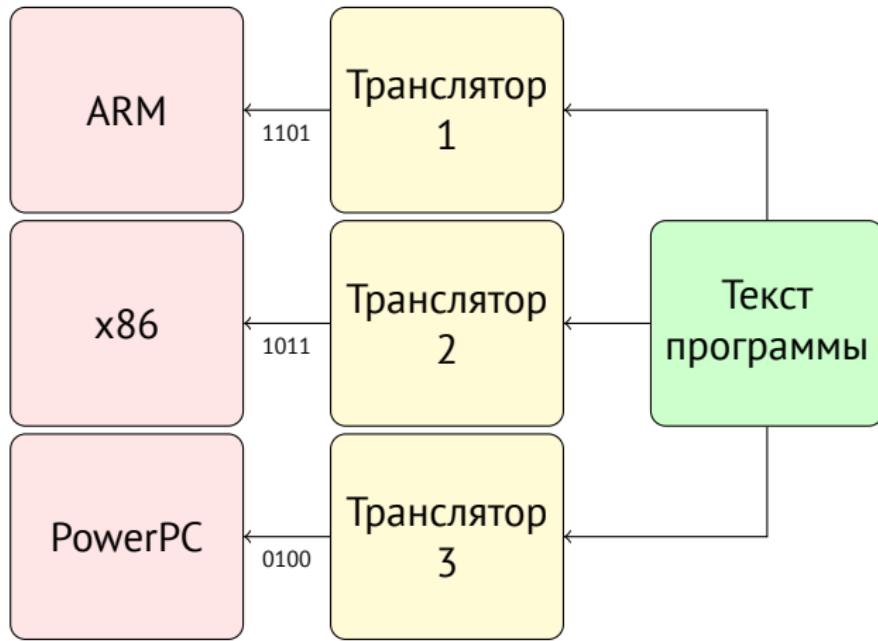
# Машинно-независимые языки программирования

Языки программирования, структура которых не связана с конкретной ЭВМ и позволяют выполнять составленные на нем программы на любой ЭВМ, снабженной **трансляторами** с этого языка.

# Трансляция

- Программа, написанная на языке высокого уровня преобразуется (транслируется) в машинный код при помощи специальной программы – **транслятора**
- **Транслятор** – это специальная программа, преобразующая текстовый файл программы в исполняемый код

# Трансляция для разных платформ



# Компиляторы и интерпретаторы

Для преобразования команд языка программирования в машинный код используются трансляторы, которые бывают двух типов

- Интерпретаторы
- Компиляторы

# Компиляторы

- **Компилятор** – программа или техническое средство, выполняющее **компиляцию**.
- **Компиляция** – трансляция (перевод) программы, составленной на языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду.
- Результат компиляции – **исполнимый модуль** обладает максимальной возможной производительностью и **привязан к определённой операционной системе и процессору**.

# Преимущества и недостатки компиляции

- ✓ Высокая скорость (исполняется машинный код)
- ✓ Не требуется дополнительного ПО для запуска\* программы
- ✗ Время отладки программы увеличивается (после исправления ошибки необходима перекомпиляция всей программы)
- ✗ Программу нельзя запустить на ЭВМ другого типа

# Интерпретатор

Интерпретатор выполняет построчный анализ, обработка и выполнение программы, без обработки всего текста программы

# Преимущества и недостатки интерпретации

## Преимущества и недостатки

- ✓ Программа может исполняться на разных платформах
- ✓ Меньший размер кода (нет машинного кода, есть только текст)
- ✓ Простая отладка программы
- ✗ Невозможно выполнение программы без интерпретатора
- ✗ Медленное исполнение (затраты времени на анализ выражений и невозможность оптимизации)

# Трансляция Си в код ассемблера

```

1 #include <stdio.h>
2
3 int main (void)
4 {
5     puts ("Hello , World !");
6     return 0;
7 }
```

gcc -O2 -S -c hworld.c

```

1          .file   "hworld.c"
2          .section      .rodata.str1.1, "aMS", @progbits,1
3 .LC0:           .string  "Hello , World !"
4          .section      .text.unlikely, "ax", @progbits
5 .LCOLDB1:        .section      .text.startup, "ax", @progbits
6 .LHOTB1:         .p2align 4,,15
7          .globl   main
8          .type    main, @function
9
10         main:
11         .LFB23:
12             .cfi_startproc
13             subq   $8, %rsp
14             .cfi_def_cfa_offset 16
15             movl   $.LC0, %edi
16             call    puts
17             xorl   %eax, %eax
18             addq   $8, %rsp
19             .cfi_def_cfa_offset 8
20             ret
21             .cfi_endproc
22 .LFE23:          .size    main, .-main
23             .section      .text.unlikely
24 .LCOLDE1:        .section      .text.startup
25 .LHOTE1:         .ident   "GCC: (Ubuntu 5.4.0) 5.4.0 20160609"
26             .section      .note.GNU-stack, "", @progbits
27
28
29
30
31
```

# Трансляция Си в машинный код

## 10% машинного кода

```

1 | #include <stdio.h>
2 |
3 | int main (void)
4 | {
5 |     puts ("Hello , World!");
6 |     return 0;
7 | }
```

gcc -Os hworld.c -o hworld

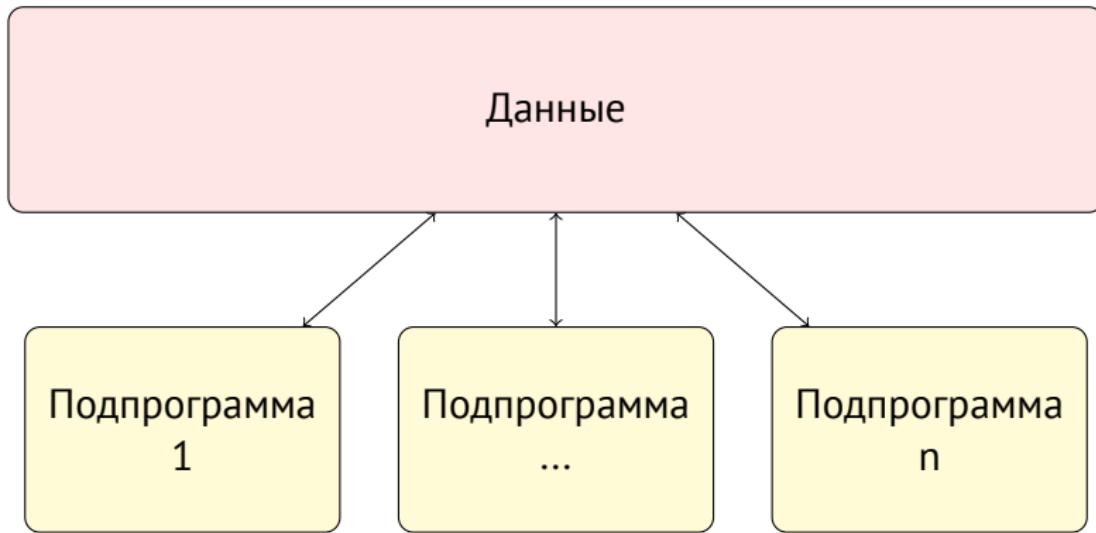
00000000	457f	464c	0102	0001	0000	0000	0000	0000	0000	0000
00000010	0002	003e	0001	0000	0440	0040	0000	0000	0000	0000
00000020	0040	0000	0000	0000	19e0	0000	0000	0000	0000	0000
00000030	0000	0000	0040	0038	0009	0040	001f	001c	0000	0000
00000040	0006	0000	0005	0000	0040	0000	0000	0000	0000	0000
00000050	0040	0040	0000	0000	0040	0040	0000	0000	0000	0000
00000060	01f8	0000	0000	0000	01f8	0000	0000	0000	0000	0000
00000070	0008	0000	0000	0000	0003	0000	0004	0000	0000	0000
00000080	0238	0000	0000	0000	0238	0040	0000	0000	0000	0000
00000090	0238	0040	0000	0000	001c	0000	0000	0000	0000	0000
000000a0	001c	0000	0000	0000	0001	0000	0000	0000	0000	0000
000000b0	0001	0000	0005	0000	0000	0000	0000	0000	0000	0000
000000c0	0000	0040	0000	0000	0040	0000	0000	0000	0000	0000
000000d0	06f4	0000	0000	0000	06f4	0000	0000	0000	0000	0000
000000e0	0000	0020	0000	0000	0001	0000	0006	0000	0000	0000
000000f0	0e10	0000	0000	0000	0e10	0060	0000	0000	0000	0000
00001000	0e10	0060	0000	0000	0228	0000	0000	0000	0000	0000
00001100	0230	0000	0000	0000	0000	0020	0000	0000	0000	0000
00001200	0002	0000	0006	0000	0e28	0000	0000	0000	0000	0000
00001300	0e28	0060	0000	0000	0e28	0060	0000	0000	0000	0000
00001400	01d0	0000	0000	0000	01d0	0000	0000	0000	0000	0000
00001500	0008	0000	0000	0000	0004	0000	0004	0000	0000	0000
00001600	0254	0000	0000	0000	0254	0040	0000	0000	0000	0000
00001700	0254	0040	0000	0000	0044	0000	0000	0000	0000	0000
00001800	0044	0000	0000	0000	0004	0000	0000	0000	0000	0000
00001900	e550	6474	0004	0000	05d4	0000	0000	0000	0000	0000
00001a00	05d4	0040	0000	0000	05d4	0040	0000	0000	0000	0000
00001b00	0034	0000	0000	0000	0034	0000	0000	0000	0000	0000
00001c00	0004	0000	0000	0000	e551	6474	0006	0000	0000	0000
00001d00	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

# Языки, используемые при создании программ



## **Поколения языков высокого уровня**

# Языки первого поколения<sup>11</sup>



# Языки первого поколения

- Программы состоят из подпрограмм
- Используются глобальные переменные – данные открытые для всех подпрограмм
- ✗ Ошибка, допущенная в одной части программы, может оказать разрушительное влияние на остальную часть системы
- ✗ Низкая надежность, запутанность большой программы

# ФОРТРАН (FORTRAN – FORmula TRANslator)

- Разработан в 1954-1957 годах программистами под руководством Джона Бэкуса (John W. Backus) в IBM
- Разработан и используется по настоящее время для научных и инженерных вычислений (BLAS, LINPACK, EISPACK, ODEPACK, IMSL)
- Для эффективности вычислений многие конструкции языка первоначально разрабатывались с учетом архитектуры машины IBM 407
- Существуют стандарты языка: FORTRAN 66, FORTRAN 90, FORTRAN 95, FORTRAN 2003, FORTRAN 2008

# Алгол

- Разработан в 1958-1960 годах как универсальный язык программирования
- Авторы языка хотели создать язык **независимый от конкретной архитектуры** вычислительной системы, **удобный** для описания алгоритмов и применяющий систему обозначений, близкую к той, что принята в математике.
- Широко использовался в Европе и СССР для научных вычислений
- Появилась блочная структура организации программ: каждый блок (главная программа, подпрограммы) ограничивался ключевыми словами **begin** и **end**
- Появились рекурсивные процедуры

# Программа на языке Алгол-60

Дана исходная система семи дифференциальных уравнений первого порядка:

- 1).  $\frac{dt}{dt} = 1;$
- 2).  $\frac{dx}{dt} = \dot{x};$
- 3).  $\frac{dy}{dt} = \dot{y};$
- 4).  $\frac{d\dot{x}}{dt} = \dot{\dot{x}};$
- 5).  $\frac{d\dot{x}}{dt} = \frac{(P_1 - C_1 \frac{\rho U^2}{2} S_m) \cos \varphi + (P_2 + C_2 \frac{\rho U^2}{2} S_m) \sin \varphi}{m};$
- 6).  $\frac{d\dot{y}}{dt} = \frac{(P_1 - C_1 \frac{\rho U^2}{2} S_m) \sin \varphi - (P_2 + C_2 \frac{\rho U^2}{2} S_m) \cos \varphi}{m} - g;$
- 7).  $\frac{d\dot{\psi}}{dt} = \frac{-P_2(x_p - x_r) + C_2 \frac{\rho U^2}{2} S_m (x_r - x_d)}{J};$

где:  $P_1, P_2, \rho, S_m, m, g, x_p, x_r, x_d, J$  – постоянные;  
 $C_1, C_2, U, \varphi$  – переменные величины.

Требуется определить:  $x, y, \varphi, \dot{x}, \dot{y}, \dot{\psi}$ .

Программа составлена на языке АЛГОЛ-60.

Исходные данные заданы массивом  $isx[1:21]$ , величины  $C_1$  и  $C_2$  заданы массивами  $mct, mct[1:12]$ . Исходная система дифференциальных уравнений решается методом Рунге-Кутта с постоянным шагом интегрирования.

Выход из решения осуществляется по одной функции:  $f = \dot{y}$ . На печать выдается на первом шаге и, затем, через каждые 10 шагов:  $t, x, y, \varphi, \dot{x}, \dot{y}, \dot{\psi}$ .

В программе используются следующие стандартные программы:

0042  
1045  
1041  
1071

Необходимо расширение рабочего поля ИС-2. В данной программе ПИ с 6000 ячеек. Программный останов: 077 в ячейке I242. Время счета – 5 минут.

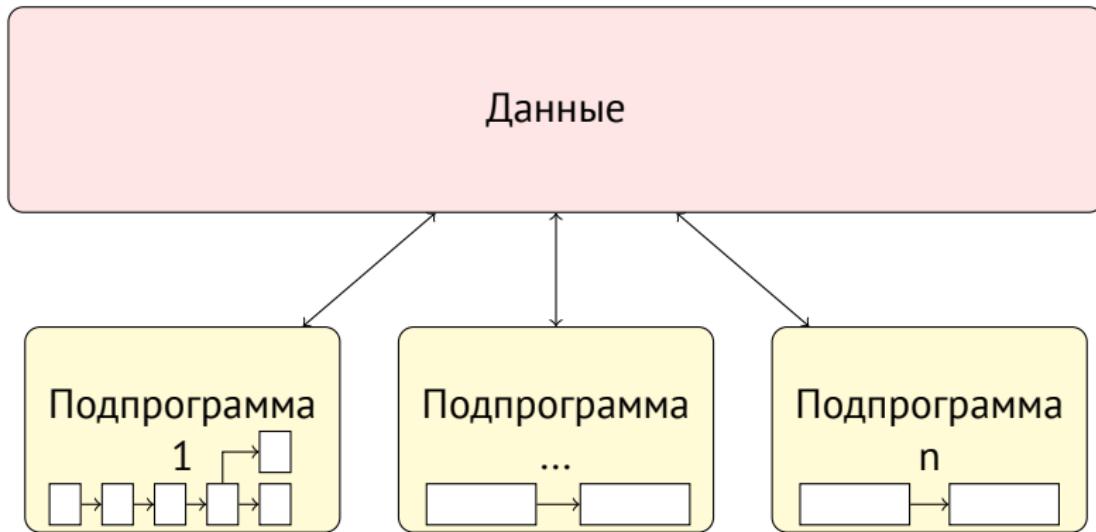
Дата составления программы: 17.11.67.

```

begin array isx[1:21], k[1:28], e[1:3], t[1:4],
mct, mcn[1:12], ct, cn[1:2], f[1:7];
Boolean array pi[1:3];
real uk, s, u, a, p1, p2, p1x, p1y, p2x, p2y, n, mn;
integer i, j, h;
P0042(isx, mct, mcn);
for i:=1 step 1 until 7 do k[i]:=isx[i+14];
e[1]:=isx[2]; e[2]:=0; j:=isx[14]-1;
P1045(isx[1], 7, i, k, o, e, true, AN, AK, AK, BK);
go to BK;
AN: ok:=k[12]+k[13]*t2; s:=(isx[6]*uk*isx[5])/2;
u:=arctg(k[13]/k[12])*57.3; a:=k[11]-u;
ct[1]:=abs(a); P1071(6, i, met, ct, r, t, 1, false);
if k[8]<isx[13] then h:=1 else h:=0;
P1:=isx[8]*h-ct[2]*s; P1x:=P1*cos(k[11]/57.3);
P1y:=P1*sin(k[11]/57.3); cn[1]:=abs(a);
P1071(6, i, mcn, cn, r, t, 1, false); n:=cn[2]*s;
if a>0 then begin mn:=-n; P2:=isx[9]*h-n;
go to C end else mn:=n; P2:=isx[9]*h+n;
C: P2x:=P2*x.sin(k[11]/57.3); P2y:=P2*x.cos(k[11]/57.3);
k[12]:=1; for i:=1 step 1 until 3 do k[i+8]:=k[i+11];
k[12]:=(P1x+P2x)/(isx[3]); k[13]:=(P1y-P2y)/(isx[3]-isx[7]);
k[14]:=57.3*(isx[9]*h*(isx[10]-isx[12])+mn*x
(isx[12]-isx[11]))/isx[4];
AK: e[3]:=k[6]; j:=j+1; if j=isx[14] then begin
for i:=1 step 1 until 7 do f[i]:=k[i]; j:=0;
P1041(f) end; BK: stop end

```

# Структурное программирование



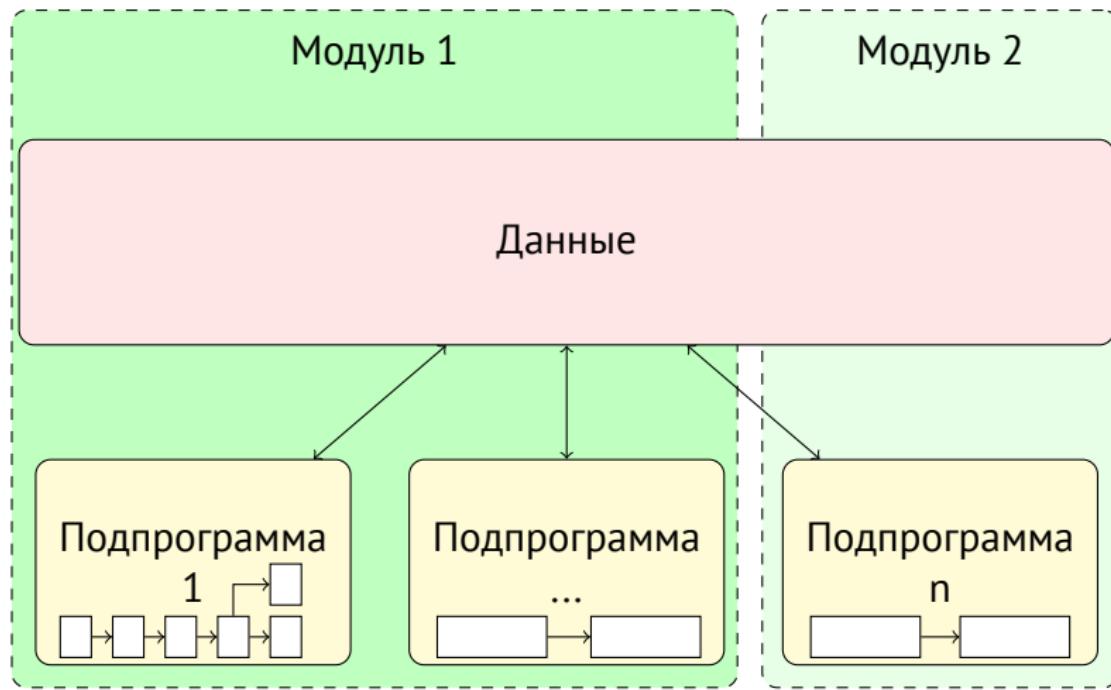
# Структурное программирование

- Программа разбивается на составляющие элементы
- В программе четко обозначены управляемые структуры, программные блоки, автономные **подпрограммы**.
- Базовые управляемые структуры<sup>4</sup>
  - последовательность
  - ветвление
  - цикл
- Используются **локальные переменные**<sup>9</sup>
- Разработка программы ведётся пошагово, методом “сверху вниз”

# Язык Паскаль

```
1 program QuadraticEquation;
2 var
3   a,b,c,d,x1,x2: real;
4 begin
5   write('введите коэффициенты уравнения a,b,c');
6   readln(a,b,c);
7   d:= b*b-4*a*c;
8   if d >= 0 then begin
9     x1:= (-b+sqrt(d))/(2*a);
10    x2:= (-b-sqrt(d))/(2*a);
11    writeln('x1=',x1:4:2, ', x2=',x2:4:2);
12  end
13  else begin
14    writeln('Действительных корней нет');
15  end
16 end.
```

# Топология языков третьего поколения



# Модульное программирование

- Программа разбивается на модули изолированные программные сегменты с четко определённым **интерфейсом**, описывающим как подготовить данные и как вызвать функции модуля преобразующие эти данные.
- Отладка модулей может проводиться независимо
- Модули одной программы могут разрабатываться на различных языках программирования
- Модули могут использоваться повторно

# Объектно-ориентированное программирование

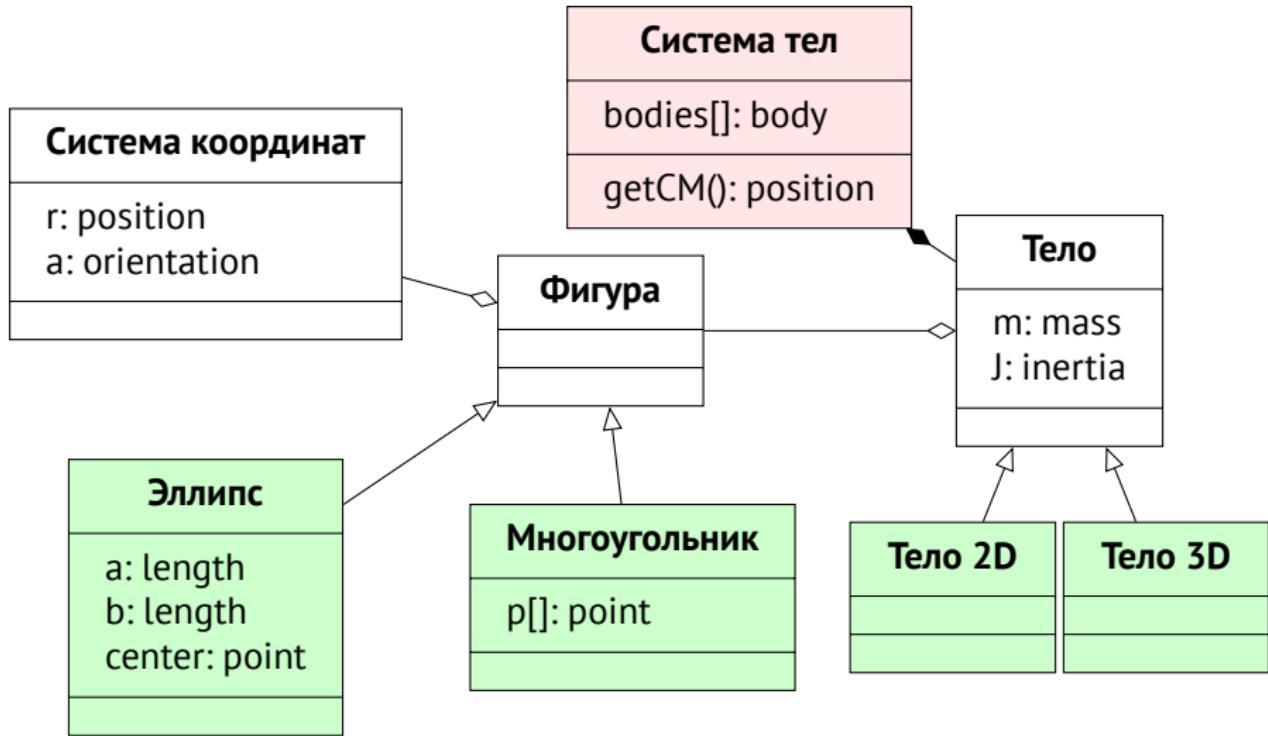
## Основные принципы

- Основной элемент конструкции – модуль
- Модуль состоит из связанных классов и объектов
- Классы образуют иерархию (наследование, включение)

# Объектные и объектно-ориентированные языки

- Первый язык с поддержкой объектно-ориентированного программирования *Simula-67*
- *Smalltalk* (70е годы) – первый объектно-ориентированный язык программирования

# Иерархия классов



# Выбор языка программирования

- Выбор языка зависит от решаемой задачи
- Разные языки требуют от программиста различного уровня внимания к деталям при реализации алгоритма

## Области применения

Научные вычисления	C++, FORTRAN, Java
Системное программирование	C, C++, Java
Искусственный интеллект	LISP, Prolog
Издательская деятельность	Postscript, TeX, LaTeX
Описание документов	HTML, XML

## **Список источников**

# Список источников

- [1] ГОСТ 19781–90 Обеспечение систем обработки информации программное.
- [2] Progopedia, “Энциклопедия языков программирования.” [Online]. Available: <http://progopedia.ru/>. [Accessed: 04-Sep-2016].
- [3] Бородина Т. А. Turbo Pascal для школьников [Online]. Available: <http://tat67183862.narod.ru/> [Accessed: 04-Sep-2016].
- [4] Wikipedia. Structured programming [Online]. Available: [https://en.wikipedia.org/wiki/Structured\\_programming](https://en.wikipedia.org/wiki/Structured_programming) [Accessed: 04-Sep-2016].
- [5] J. Daintith, “A dictionary of computing,” Oxford dictionary of computing, vol. 6. p. 567, 2008.
- [6] А.Д.Липенков Информатика [Online]. Available: [http://uchu2008.narod.ru/razdely/informatika/inform\\_lectures/soderganie/temy.html](http://uchu2008.narod.ru/razdely/informatika/inform_lectures/soderganie/temy.html) [Accessed: 04-Sep-2016].
- [7] Wikipedia. LaTeX [Online]. Available: <https://ru.wikipedia.org/wiki/LaTeX> [Accessed: 04-Sep-2016].
- [8] Wikipedia. Блок-схема [Online]. Available: <https://ru.wikipedia.org/wiki/Блок-схема> [Accessed: 04-Sep-2016].
- [9] Игорь Дидковский, Перспективные ОС и языки программирования. Available: <http://e-skin.hut.ru/langs/> [Accessed: 04-Sep-2016].
- [10] INF-W Available: [http://inf-w.ru/?page\\_id=4367](http://inf-w.ru/?page_id=4367) [Accessed: 04-Sep-2016].
- [11] Г. Буч, Объектно-ориентированный анализ и проектирование с примерами приложений на С++. Второе издание. М.: “Издательство Бином”, 1998.