

**Кафедра вычислительных систем**

Сибирский государственный университет телекоммуникаций и информатики

# **Лекция 1. Введение в язык программирования СИ**

**Перышкова Евгения Николаевна**

E-mail: [e.peryshkova@gmail.com](mailto:e.peryshkova@gmail.com)

Сайт кафедры: <http://csc.sibsutis.ru>

Курс «Программирование»

Осенний семестр, 2017

# Многообразие языков программирования

На сегодняшний день существует несколько тысяч языков программирования.

Наиболее полный перечень – Bill Kinnersley – около 2500 языков

<http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm>

Около 50 наиболее популярных языков в виде исторического графа с 1954 года по наши дни:

<http://www.levenez.com/lang/>

# Прежде, чем начать программировать

1. Выбрать язык программирования
2. Выбрать среду разработки

# Виды языков программирования

1. Компиляторы, компилируемые языки программирования
2. Интерпритаторы, интерпретируемые языки программирования

# Компиляция

Процесс преобразования программы (трансляция), составленной на исходном языке высокого уровня (Си, Pascal и др.), в эквивалентную программу на низкоуровневом языке, близком к машинному коду.

# Интерпретация

Пооператорный анализ, обработка и тут же выполнение исходной программы или запроса.

# Классификация языков программирования

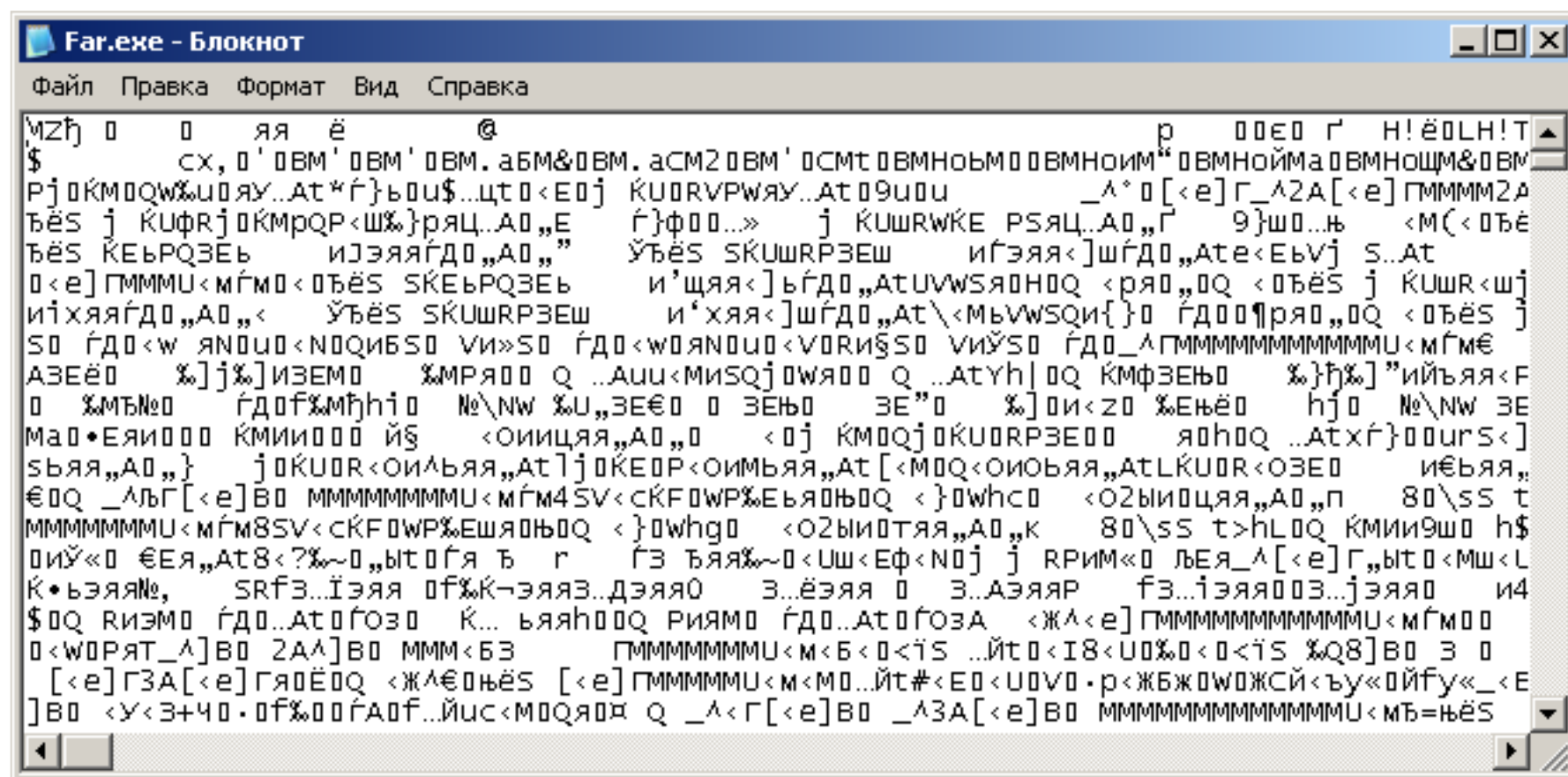
Язык программирования – система синтаксических правил для написания команд, из которых состоит программа, непосредственно исполняемая на компьютере (язык низкого уровня) или исполняемая на компьютере после преобразования (трансляции) в исполняемую программу (язык высокого уровня)

# Язык низкого уровня

C:\Program Files (x86)\Far2\Far.exe								1251	1380352	Col 0
0000000000:	4D	5A	90	00	03	00	00	04 00 00 00 FF FF 00 00	MZh ♥ ♦	яя
0000000010:	B8	00	00	00	00	00	00	40 00 00 00 00 00 00 00	ë	@
0000000020:	00	00	00	00	00	00	00	00 00 00 00 00 00 00 00		
0000000030:	00	00	00	00	00	00	00	00 00 00 00 F0 00 00 00		p
0000000040:	0E	1F	BA	0E	00	B4	09 CD	21 B8 01 4C CD 21 54 68	♪▼♫ ♪○H!ë☺LH!	Th
0000000050:	69	73	20	70	72	6F	67 72	61 6D 20 63 61 6E 6E 6F	is program canno	
0000000060:	74	20	62	65	20	72	75 6E	20 69 6E 20 44 4F 53 20	t be run in DOS	
0000000070:	6D	6F	64	65	2E	0D	0D 0A	24 00 00 00 00 00 00 00	mode.♪♪☐\$	
0000000080:	63	78	2C	1E	27	19	42 4D	27 19 42 4D 27 19 42 4D	cx,▲'↓BM'↓BM'↓BM	
0000000090:	2E	61	C1	4D	26	19	42 4D	2E 61 D1 4D 32 19 42 4D	.aBM&↓BM.aCM2↓BM	
00000000A0:	27	19	43	4D	74	18	42 4D	48 6F DC 4D 05 19 42 4D	'↓CMt↑BMНoBM♣↓BM	
00000000B0:	48	6F	E8	4D	93	19	42 4D	48 6F E9 4D E0 19 42 4D	НоиМ"↓BMНойМа↓BM	
00000000C0:	48	6F	D9	4D	26	19	42 4D	48 6F D8 4D 26 19 42 4D	НоШМ&↓BMНоШМ&↓BM	
00000000D0:	48	6F	DF	4D	26	19	42 4D	52 69 63 68 27 19 42 4D	НоЯМ&↓BMRich'↓BM	
00000000E0:	00	00	00	00	00	00	00	00 00 00 00 00 00 00 00		
00000000F0:	50	45	00	00	4C	01	06 00	2B E2 49 4D 00 00 00 00	PE L☺♠ +BIM	
0000000100:	00	00	00	00	E0	00	22 01	0B 01 0A 00 00 F0 10 00	a "☺♂☺☐	p▶



# Язык низкого уровня



```
Far.exe - Блокнот
Файл  Правка  Формат  Вид  Справка
MZj 0 0  яя ё @ р 00Е0 Г Н!ё0LН!Т
$ cx, 0'0BM'0BM'0BM.аBM&0BM.асМ20BM'0СМт0BMноьМ00BMноим"0BMнойМа0BMнощМ&0BM
Pj0KMQW%u0ЯУ..At*г}ь0u$...цт0<Е0j KUORVPWЯУ..At09u0u _^0[<e]Г_Λ2A[<e]ГММММ2A
ЪёS j KUфRj0KMPQP<ш%}ряц..А0,,Е ф}ф00...» j KUWRWKE PСЯЦ..А0,,Г 9}ш0...ъ <М(<0Ъё
ЪёS KEЪPQЗЕЪ иЗэяягдо,,А0,, УЪёS SKUWRPЗЕШ иГэяя<]шгдо,,Аte<ЕьVj S..At
0<e]ГМММУ<мгм0<0ЪёS SKЕЪPQЗЕЪ и'щяя<]ьгдо,,AtUVWSЯ0H0Q <ря0,,0Q <0ЪёS j KUWR<шj
иixяягдо,,А0,,< УЪёS SKUWRPЗЕШ и'хяя<]шгдо,,At\<мьVWSQi{ }0 гдо00ря0,,0Q <0ЪёS j
S0 гдо<w ян0u0<N0QИБS0 Vi»S0 гдо<wоян0u0<V0RиS0 ViУS0 гдо_ΛГМММММММММММУ<мгмЕ
АЗЕё0 %}j%]ИЗЕМО %МРя00 Q ..Аu0<MиSQj0Wя00 Q ..AtYh|0Q КМфЗЕЬ0 %}h%]"ийъяя<F
0 %мЪ№0 гдоf%мhhi0 №\NW %У,,ЗЕё0 0 ЗЕЬ0 ЗЕ"0 %]и0<Z0 %Еньё0 hj0 №\NW ЗЕ
Ма0•Еяи000 КМИи000 йS <Оиицяя,,А0,,0 <0j KMQj0KUORPЗЕ00 я0h0Q ..Atxf}00urs<]
сьяя,,А0,,} j0KUOR<ОиАьяя,,Atj0KEOP<Оимьяя,,At[<MQ<Оиобьяя,,AtLKUOR<ОЗЕ0 иЕьяя,,
Е0Q _ΛьГ[<e]B0 ММММММММУ<мгм4SV<СКFOWP%Еья0Ь0Q <}0whc0 <O2Ыи0цяя,,А0,,п 80\ss t
ММММММММУ<мгм8SV<СКFOWP%Ешя0Ь0Q <}0whg0 <O2Ыи0тяя,,А0,,к 80\ss t>hL0Q КМИи9ш0 h$
пиУ<0 €Ея,,At8<?%~0,,Ыт0Гя Ъ г ГЗ Ъяя%~0<Uш<Еф<N0j j PРим<0 ъЕя_Λ[<e]Г,,Ыт0<Mш<L
К•ьяя№, SRfЗ...Iэяя 0f%К~эяяЗ..дэяя0 З...ёэяя 0 З..АэяяР fЗ...iэяя00З...jэяя0 и4
$0Q РиэМО гдо..At0fоз0 К...ьяяh00Q РияМО гдо..At0fозА <жΛ<e]ГМММММММММММУ<мгм00
0<wOPЯT_Λ]B0 2AΛ]B0 МММ<БЗ ГМММММММУ<м<Б<0<is ...йт0<I8<U0%0<0<is %Q8]B0 З 0
[<e]ГЗА[<e]Гя0Е0Q <жΛё0ньёS [<e]ГМММММУ<м<МО...йт#<Е0<U0V0•p<жБж0W0ЖСй<ьУ«0йfy«_<E
]B0 <У<З+Ч0•0f%00ГA0f...йус<МОQЯ0Я Q _Λ<Г[<e]B0 _ΛЗА[<e]B0 МММММММММММММУ<мЪ=ньёS
```

# Язык низкого уровня

**\$ od -t x1 Far.exe |head**

```
00000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00
00000020 b8 00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00 00
00001000 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68
00001200 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f
00001400 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20
00001600 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00
00002000 63 78 2c 1e 27 19 42 4d 27 19 42 4d 27 19 42 4d
00002200 2e 61 c1 4d 26 19 42 4d 2e 61 d1 4d 32 19 42 4d
```

# Язык низкого уровня

Ни одну сложную программу в наше время никто не будет писать на **языке машинного кода**.

**Ассемблер**, который представляет собой мнемоническую запись тех же машинных команд, используется системными программистами достаточно редко.

Чаще используют небольшие **ассемблерные вставки** в код на Си или подключаются (линкуются при сборке) библиотеки функций, скомпилированных на Ассемблере.

# Язык низкого уровня

Процесс трансляции программы с языка Ассемблера в машинный код называется ассемблированием.

Обратный процесс – дизассемблированием.

Машинный код	Ассемблер
0E	PUSH CS
1F	POP DS
BA0E00	MOV DX,000E
B409	MOV AH,09
CD21	INT 21
B8014C	MOV AX,4C01
CD21	INT 21
54	PUSH SP
68	DB 68
69	DB 69
7320	JNB 0033

# Язык низкого уровня

**\$ cat hello.asm**

BITS32 ; Говорим компилятору, что код 32-битный  
; В исполняемом файле может быть различное количество секций.  
; Секции обычно выделяются по содержимому:  
; .text – для кода,  
; .data – для данных,  
; .bss – для неинициализированных данных.  
section .text ; Начало секции кода  
global \_start ; Метка \_start должна быть глобальной,  
; чтобы линкер смог её найти и сделать точкой входа в программу.

\_start:

mov eax,4 ; системный вызов <<write>>

mov ebx,1 ; стандартный вывод

mov ecx,msg ; адрес сообщения

mov edx,[msg\_size] ; длина

int 0x80 ; вызов прерывания

mov eax,1 ; системный вызов <<exit>>

xor ebx,ebx ; код выхода

int 0x80 ; вызов прерывания

section .data ; Начало секции данных

; Объявление переменной с сообщением

msg db 'Привет',0xA

; Объявление переменной с длиной сообщения

msg\_size dd \$-msg

**\$ nasm -f elf hello.asm**

hello.asm:1: warning: label alone on a line without a colon might be in  
error

**\$ ld -m elf\_i386 -o hello hello.o**

**\$ ./hello**

Привет

## Язык высокого уровня

Разрабатывались с целью повышения производительности труда программистов за счет использования команд (операторов), использующих слова английского языка (в основном), соответствующих последовательности из многих машинных инструкций

# Язык высокого уровня

```
$ cat hello.c
```

```
#include <stdio.h>
```

```
void main(void)
```

```
{
```

```
printf("Привет\n");
```

```
}
```

```
$ gcc -c hello.c
```

```
$ gcc -o hello hello.o
```

```
$ ./hello
```

```
Привет
```

# Виды языков высокого уровня

## 1. Универсальные

Fortran, Cobol, Algol, C

C++, C#, Pascal, Objective-C

## 2. Проблемно-ориентированные

PHP, Perl, JavaScript

Lisp, Prolog, Multilisp



# Виды языков высокого уровня

**Процедурные языки** – языки высокого уровня, в которых используется метод декомпозиции программы на отдельные связанные друг с другом модули – подпрограммы (процедуры и функции)

# Виды языков высокого уровня

**Объектно-ориентированные языки** – дальнейший уровень развития процедурных языков с основной концепцией организации программы как совокупности программных объектов. Алгоритм решения задачи представляет собой последовательность создания экземпляров описанных предварительно пользователем библиотечных объектов и использования их методов

# Виды языков высокого уровня

**Функциональные языки** – языки искусственного интеллекта.

Последовательность функций и выражения, которые нужно вычислить.

Основная структура данных – связный список.

# Виды языков высокого уровня

**Логические языки** – ориентированы на решение проблем без описания алгоритмов, языки искусственного интеллекта.

PROGOL – экспертные системы

# Виды языков высокого уровня

**Языки сценариев или скрипты** – функциональные и/или ОО языки для создания программ, исполняемых в определенной программной среде

# Виды языков высокого уровня

```
$ cat while.sh
```

```
#!/bin/bash
```

```
date -R
```

```
echo "Начинаем отсчёт."
```

```
date
```

```
i=5
```

```
while [ $i -ge 1 ]
```

```
do
```

```
echo $i
```

```
let i=i-1
```

```
done
```

```
echo "Конец."
```

```
$ ./while.sh
```

```
Wed, 09 Jan 2013 18:44:33 +0400
```

```
Начинаем отсчёт.
```

```
5
```

```
4
```

```
3
```

```
2
```

```
1
```

```
Конец.
```

# Виды языков высокого уровня

**Языки, ориентированные на данные**, созданы специально для работы с одним определенным типом данных

APL – матрицы и векторы без циклов

Snobol, Icon – обработка строк

SETL – описание операций над множествами

Языки работы с базами данных

Автоматизация работы с офисными приложениями

# Алгоритмизация

**Алгоритм** – описание последовательности действий для решения поставленной задачи.

Интуитивное определение:

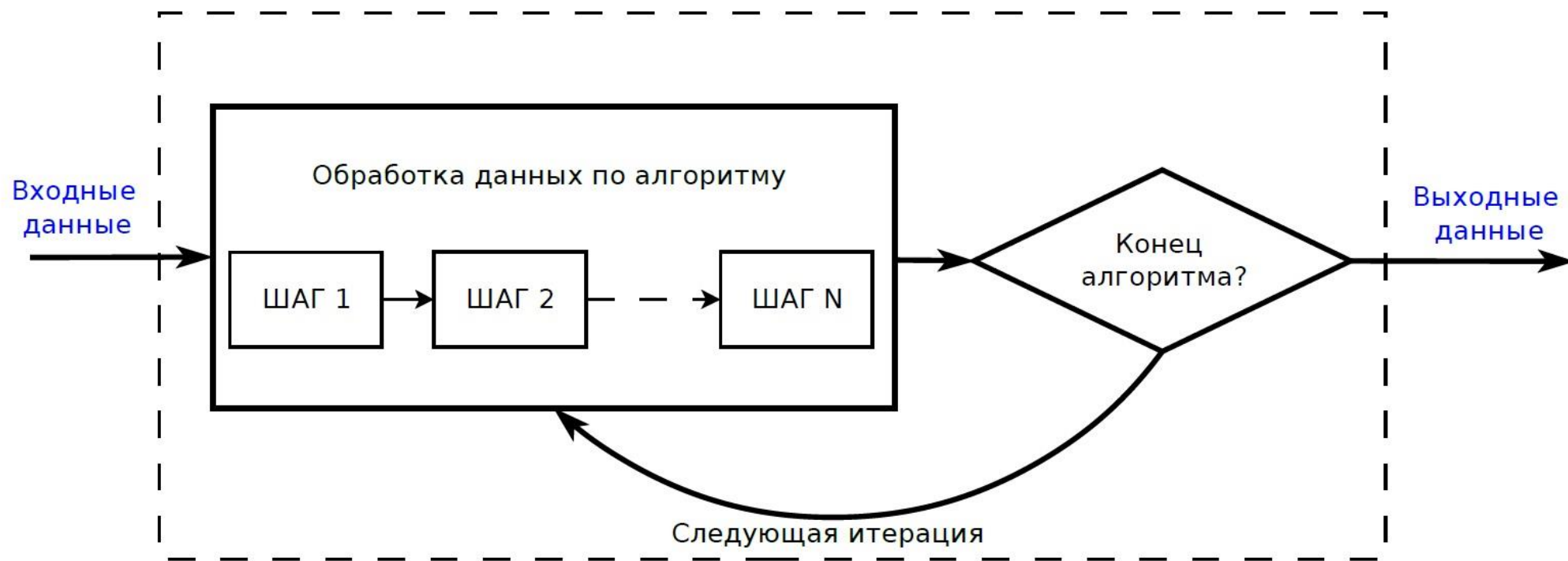
**Алгоритм** – точное предписание, которое задает вычислительный процесс, начинающийся с произвольного исходного данного и направленный на получение полностью определяемого этим исходным данным результата



# Особенности алгоритма

- 1. Конечность**
- 2. Определенность**
- 3. Ввод**
- 4. Вывод**
- 5. Эффективность**

# Схема алгоритмического процесса



# Математические определения алгоритма

Были разработаны 3 типа математических моделей алгоритма:

1. Основан на понятии рекурсивной функции
2. На основе описания детерминированного устройства, работающего по шагам и выполняющего на каждом шаге заранее определенные операции с элементами устройства и данными (машина Тьюринга)
3. Связан с работой со словами в некотором фиксированном алфавите, которые с помощью подстановок переходят в другие слова (нормальный алгоритм Маркова)

# Свойства и формы описания алгоритмов

Описание алгоритма может быть представлено в виде:

1. Текстовой инструкции;
2. Графической схемы;
3. Программы на одном из языков программирования

# Свойства и формы описания алгоритмов

Каждый алгоритм должен иметь:

1. Название, отражающее суть решаемой задачи,
2. Описание исходной информации,
3. Описание последовательности действий,
4. Описание выходной информации.

# Свойства алгоритмов

1. Универсальность
2. Дискретность
3. Однозначность
4. Результативность

# Основные типовые алгоритмы

1. **Линейный** – неизменная последовательность операций от его начало до конца без повторов действий;
2. **Разветвляющийся** – последовательность выполняемых действий может изменяться в зависимости от каких-либо условий;
3. **Циклический** – группа операций, которые могут повторяться многократно, кратность повтора определяется некоторым условием.

# Принципы фон Неймана

1. Использование **двоичной системы счисления** в вычислительных машинах. **Цель:** технически реализовать устройства хранения информации в двоичной системе счисления существенно проще, чем в устройства, основанные на десятичной системе. Также проще реализовать выполнение арифметических и логических операций.
2. **Программное управление ЭВМ.** Работа ЭВМ контролируется программой, состоящей из набора команд. **Команды выполняются последовательно друг за другом.** **Цель:** вычислительное устройство становится универсальным и может решать широкий круг задач, так как их программа может быть изменена.



# Принципы фон Неймана

## 3. Память компьютера используется не только для хранения данных, но и программ.

Команды и данные кодируются в двоичной системе счисления, поэтому для их хранения может использоваться одно устройство. **Цель:** в определенных ситуациях над командами можно выполнять те же действия, что и над данными (изменять их!).

## 4. Ячейки памяти ЭВМ имеют адреса, которые последовательно пронумерованы.

**Цель:** возможность обращения к произвольным ячейкам памяти в любой момент времени. Данный принцип открыл возможность использовать переменные в программировании.

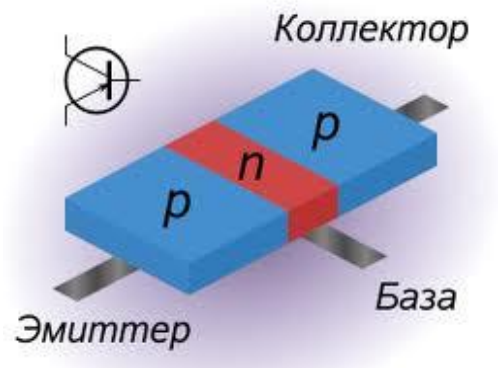
# Принципы фон Неймана

## 5. Возможность **условного перехода** в процессе выполнения программы.

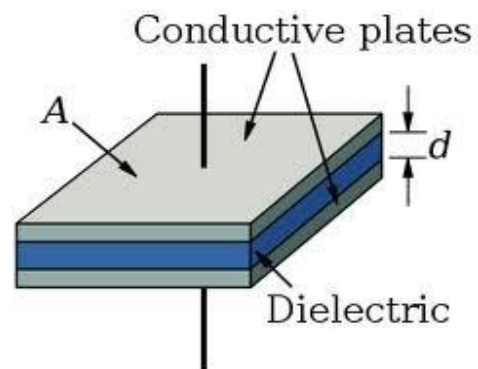
**Цель:** обеспечить управление процессом вычислений. Не смотря на то, что команды выполняются последовательно, в программах можно реализовать возможность перехода к любому участку кода. Таким образом, входные данные могут влиять на ход выполнения программы. Данный принцип используется при организации *ветвлений* и *циклов* в программировании.

# Ячейка памяти (элементная база)

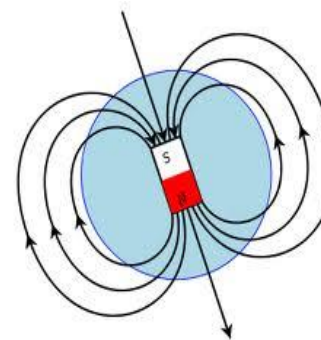
## Элементная база



**Транзистор**



**Конденсатор**

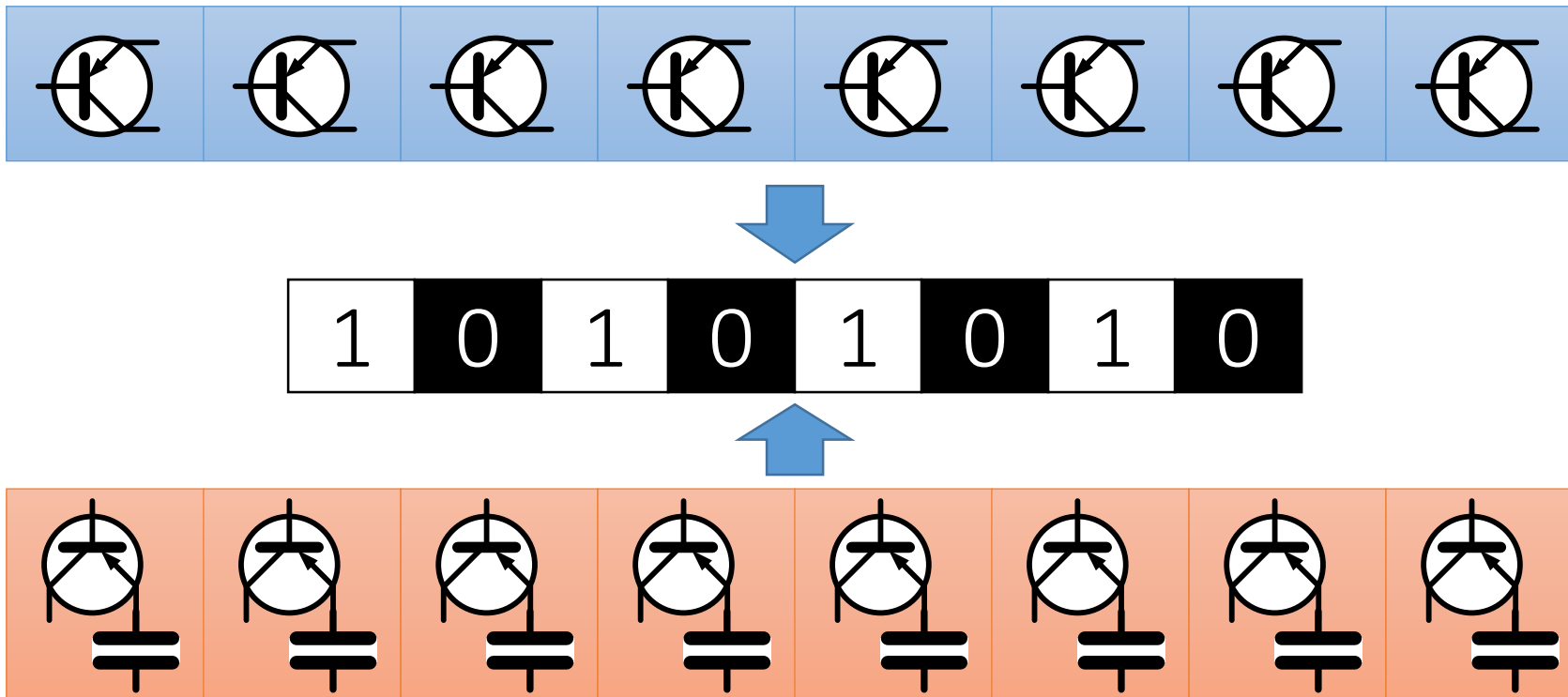


**Ферромагнетики**

**Два устойчивых состояния!**

# Ячейка памяти (логическое представление)

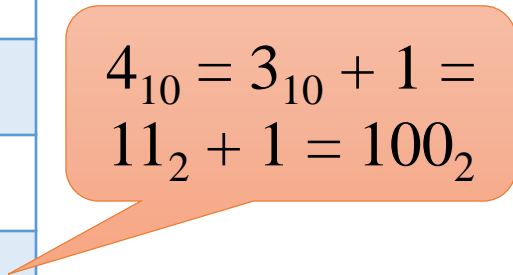
- Логическое представление ячейки не зависит от физических компонентов, использованных для ее изготовления.
- Ячейка рассматривается как набор двоичных разрядов, которые объединяются в группы из 8 штук, образующие **байт**.



# Двоичная система счисления

Цифры двоичной СС: **0, 1**

$x_{10}$	$x_2$
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111


$$\begin{aligned}4_{10} &= 3_{10} + 1 = \\ &11_2 + 1 = 100_2\end{aligned}$$

$x_{10}$	$x_2$
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

# Шестнадцатеричная система счисления

Шестнадцатеричная система счисления (шестнадцатеричные числа) – позиционная система счисления по целочисленному основанию 16.

В качестве шестнадцатеричных цифр используются десятичные цифры от 0 до 9 и латинские буквы от A до F для обозначения цифр от  $10_{10}$  до  $15_{10}$ : **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**.

**Широко используется в низкоуровневом программировании и компьютерной документации.** Минимальная единица памяти (8-битный байт) можно записать двумя шестнадцатеричными цифрами. Такое использование началось с системы IBM/360.

- В математике основание системы счисления принято указывать в десятичной системе в нижнем индексе. Например, десятичное число 1443 можно записать как  $1443_{10}$  или как  $5A3_{16}$ .
- В Си и языках схожего синтаксиса, например, в Java, используют префикс «0x». Например, «0x5A3».

# Шестнадцатеричная система счисления

Двоичная и шестнадцатеричная СС являются родственными, т.к.  $16 = 2^4$  (основание одной является степенью основания другой)

$x_2$	$x_{10}$	$x_{16}$
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

$x_2$	$x_{10}$	$x_{16}$
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

# Связь двоичной и шестнадцатеричной СС

$x_2$	$x_{10}$	$x_{16}$
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

$x_2$	$x_{10}$	$x_{16}$
10000	16	10
10001	17	11
10010	18	12
10011	19	13
10100	20	14
10101	21	15
10110	22	16
10111	23	17



# Перевод $x_2 \rightarrow x_{16}$

$x_2$	$x_{16}$	$x_2$	$x_{16}$
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
<b>0101</b>	<b>5</b>	<b>1101</b>	<b>D</b>
<b>0110</b>	<b>6</b>	1110	E
0111	7	1111	F

Для перевода из двоичной СС в шестнадцатеричную достаточно разбить  $x_2$  на 4-хразрядные блоки и перевести каждый из них по отдельности:

$$\begin{aligned} &11011010101_2 = \\ &\mathbf{0110\ 1101\ 0101}_2 = \\ &\mathbf{6D5}_{16} \end{aligned}$$

# Перевод $x_2 \rightarrow x_{16}$ T01.1

$$10011_2 = x_{16}$$

$$1000000_2 = x_{16}$$

$$01010101_2 = x_{16}$$

$$11111111_2 = x_{16}$$

$$10010010_2 = x_{16}$$

$$1010010010_2 = x_{16}$$

$x_2$	$x_{16}$
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

$x_2$	$x_{16}$
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

# Перевод $x_{16} \rightarrow x_2$

$x_2$	$x_{16}$	$x_2$	$x_{16}$
0000	0	<b>1000</b>	<b>8</b>
0001	1	1001	9
<b>0010</b>	<b>2</b>	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	<b>1111</b>	<b>F</b>

Для перевода из шестнадцатеричной СС в двоичную необходимо каждый разряд шестнадцатеричного числа представить 4-хразрядным двоичным числом:

$$2F8_{16} = 0010\ 1111\ 1000_2$$

# Перевод $x_{16} \rightarrow x_2$ T01.2

$$1A_{16} = x_2$$

$$10_{16} = x_2$$

$$211_{16} = x_2$$

$$BEEF_{16} = x_2$$

$$ABC_{16} = x_2$$

$$2A3B_{16} = x_2$$

$x_2$	$x_{16}$
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

$x_2$	$x_{16}$
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

# Восьмеричная СС T01.3

По аналогии с шестнадцатеричной СС предложите алгоритм перевода  $x_8 \rightarrow x_2$  и  $x_2 \rightarrow x_8$ . С помощью предложенного алгоритма проведите преобразования:

$$10011_2 = x_8$$

$$1000000_2 = x_8$$

$$01010101_2 = x_8$$

$$11111111_2 = x_8$$

$$10010010_2 = x_8$$

$$1010010010_2 = x_8$$

$$14_8 = x_2$$

$$181_8 = x_2$$

$$547_8 = x_2$$

$$100_8 = x_2$$

$$1234_8 = x_2$$

$$756_8 = x_2$$

# Системы счисления в языке Си

Для обозначения констант в десятичной системе счисления используется привычная запись:

26

Для обозначения восьмеричных к константе добавляется префикс "0":

26 = 032 (32<sub>8</sub>)

Для обозначения шестнадцатеричных чисел к константе добавляется префикс "0x", **x** – **heXadecimal**:

26 = 0x1A (1A<sub>16</sub>)

# Цель ВТ – обработка информации

