



НИУ ВШЭ - Нижний Новгород

November 4, 2024

Алгоритмы и структуры данных

Лекция 0. Алгоритмы, вычислительные машины и программы

Илья Сергеевич Бычков
ibychkov@hse.ru



О курсе



О курсе

Организационная информация

Продолжительность курса: октябрь - июнь / 2-4 модули /9 месяцев

Состав курса: примерно 30 лекций + 30 практик

Результаты курса: 2 оценки - оценка за модуль 2 и итоговая оценка

Оценка модуль 2 = Накопленная оценка (2) * 0.6 + Экзамен (2)* 0.4

Оценка модули 3-4 = Накопленная оценка (3-4) * 0.6 + Экзамен (3-4) * 0.4

Финальная оценка = Оценка (2) * 0.4 + Оценка (3-4) * 0.6

Из чего складывается накопленная оценка:

- Контесты - **вес 0.3**
- Контрольные - **вес 0.6, блокирующий элемент**
- Активность/доп задания - **вес 0.1**



О курсе

Команда курса

Лекции:

- Илья Сергеевич Бычков - HSE PhD, Senior Algorithm Developer (Huawei), старший научный сотрудник ЛАТАС

Семинары:

- Максим Алексеевич Захаров - Intel, Yadro
- Михаил Михайлович Железин - Yandex
- Никита Александрович Наумов - Sber
- Кирилл Сергеевич Тараканов - Huawei, ШАД



О курсе

План курса

Контексты

Письменные контрольные

II модуль												III модуль												IV модуль																							
Ноябрь						Декабрь						Январь						Февраль						Март						Апрель						Май						Июнь					
Пн	28	4	11	18	25	25	2	9	16	23	30	30	6	13	20	27	27	3	10	17	24	24	3	10	17	24	31	31	7	14	21	28	28	5	12	19	26	26	2	9	16	23	30				
Вт	29	5	12	19	26	26	3	10	17	24	31	31	7	14	21	28	28	4	11	18	25	25	4	11	18	25	1	1	8	15	22	29	29	6	13	20	27	27	3	10	17	24	1				
Ср	30	6	13	20	27	27	4	11	18	25	1	1	8	15	22	29	29	5	12	19	26	26	5	12	19	26	2	2	9	16	23	30	30	7	14	21	28	28	4	11	18	25	2				
Чт	31	7	14	21	28	28	5	12	19	26	2	2	9	16	23	30	30	6	13	20	27	27	6	13	20	27	3	3	10	17	24	1	1	8	15	22	29	29	5	12	19	26	3				
Пт	1	8	15	22	29	29	6	13	20	27	3	3	10	17	24	31	31	7	14	21	28	28	7	14	21	28	4	4	11	18	25	2	2	9	16	23	30	30	6	13	20	27	4				
Сб	2	9	16	23	30	30	7	14	21	28	4	4	11	18	25	1	1	8	15	22	29	5	5	12	19	26	3	3	10	17	24	31	31	7	14	21	28	5									
Вс	3	10	17	24	1	1	8	15	22	29	5	5	12	19	26	2	2	9	16	23	30	6	6	13	20	27	4	4	11	18	25	1	1	8	15	22	29	6									

Часть 1: Введение, битовые операции, массивы
Длительность: 2 недели
Контест: 16-17 ноября

Часть 3: Алгоритмы поиска
Длительность: 2 недели
Контест: 11-12 января

Часть 5: Сложные списки
Длительность: 2 недели
Контест: 15-16 февраля

Часть 7: Хэш-таблицы
Длительность: 2 недели
Контест: 15-16 марта

Часть 9: Графы 1
Длительность: 2 недели
Контест: 19-20 апреля

Часть 11: Жадные алгоритмы, д&с, динамическое программирование
Длительность: 3 недели
Контест: 31 мая - 1 июня

Часть 2: Сортировки, сложность алгоритмов, тестирование
Длительность: 3 недели
Контест: 7-8 декабря

Часть 4: Алгоритмы теории чисел, шифрование
Длительность: 3 недели
Контест: 1-2 февраля

Часть 6: Стеки и очереди
Длительность: 2 недели
Контест: 1-2 марта

Часть 8: Кучи
Длительность: 2 недели
Контест: 5-6 апреля

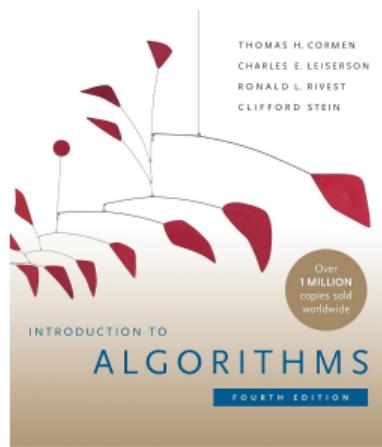
Часть 10: Графы 2
Длительность: 2 недели
Контест: 10-11 мая

Часть 12: Два указателя, строки
Длительность: 2 недели
Контест: 14-15 июня

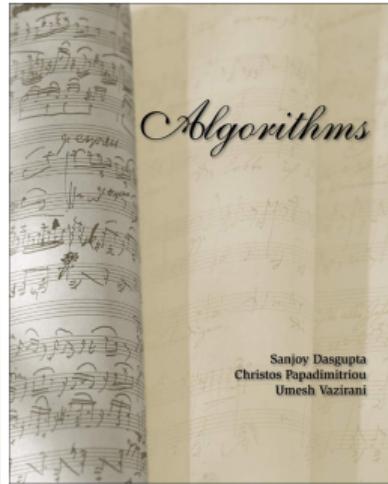


О курсе

Литература



(1)
**Introduction to
Algorithms, 4th Edition**
Thomas H Cormen
Charles E Leiserson
Ronald L Rivest
Clifford Stein



(2)
Algorithms
Sanjoy Dasgupta
Christos Papadimitriou
Umesh Vazirani



(3)
Algorithms, 4th Edition
Robert Sedgewick
Kevin Wayne

algs4.cs.princeton.edu



Лекция 0.

Алгоритмы, вычислительные машины и программы



Алгоритмы, вычислительные машины и программы

План лекции

0. План лекции
1. Понятие алгоритма
2. Вычислительные машины
3. Алгоритмы и программы
4. Алгоритмы и данные
5. Битовые операции



Понятие алгоритма

Неформальное понятие алгоритма

В повседневной жизни мы можем рассматривать **алгоритм** как последовательность действий, направленную на достижение желаемого результата.

Алгоритмы для приготовления пищи называются рецепты. Они включают заданный набор ингридиентов (**входных данных**), чётко определенную последовательность действий (**инструкции/команды**) и результат - блюдо с желаемым внешним видом и вкусовыми качествами.

Для составления алгоритмов воспроизведения музыки используются ноты. Таким образом, человек может пошагово воспроизвести записанное и получить в качестве результата желаемую музыку.



Понятие алгоритма

Более формальное понятие алгоритма

Алгоритм - понятная и точная последовательность шагов (вычислительная процедура), которая получает некоторое значение или набор значений (входные данные) и за конечное время выдает некоторое значение или набор значений в качестве результата (выходные данные).

Таким образом, алгоритм трансформирует входные данные в ответ.

Алгоритм - фундаментальная концепция всей области компьютерных наук.

В сфере информационных технологий нам будут интересны алгоритмы, которые способны решать **вычислительные задачи**, а исполнителями в этом случае будут являться **вычислительные машины**.



0. План лекции
1. Понятие алгоритма
2. Вычислительные машины
3. Алгоритмы и программы
4. Алгоритмы и данные
5. Битовые операции



Вычислительные машины

Алгоритмы и вычислительные машины

Создание алгоритма для решения некоторой вычислительной задачи - важная работа.

Как только правильный (с точки зрения выдаваемых результатов) и эффективный (с точки зрения затрачиваемых ресурсов) алгоритм придуман, решение рассматриваемой задачи больше не требует понимания принципов, на которых построен алгоритм. Далее все сводится лишь к последовательному исполнению шагов алгоритма.

Чтобы автоматически выполнять уже имеющиеся эффективные алгоритмы, люди в течение нескольких столетий изобретали и улучшали **вычислительные машины**.



Вычислительные машины

Ранние вычислительные машины

Абак - одно из самых ранних устройств для выполнения арифметических операций. В большей степени являлось средством хранения и представления чисел, процесс вычислений проводился человеком.

Первое запатентованное счетное устройство - механический калькулятор, созданный в 1640х годах Блезом Паскалем. Состояло из спиц и колёсиков, умело складывать и вычитать.



(4) Abacus (Абак/Счёты)



(5) Калькулятор Паскаля



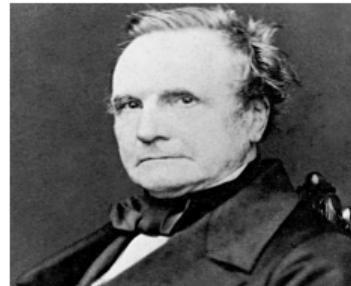
Вычислительные машины

Изобретения Чарльза Бэббиджа

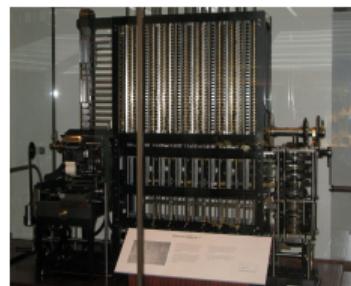
Позже английский математик и инженер Чарльз Бэббидж (1791 - 1871) создал **разностную машину (differential engine)**, способную вычислять синусы, косинусы, тангенсы и логарифмы. Вычисление этих функций разбивалось на элементарные арифметические операции.

Далее в 1834 Бэббидж разработал проект счётной машины **общего назначения**, которая могла выполнять множество операций по инструкциям, задаваемых ей программным образом.

Изобретение было названо **аналитической машиной (analytical engine)**. В проекте были собраны все инновации из других областей того времени.



Чарльз Бэббидж (Cambridge, UK), Источник - Wikipedia



Разностная машина, Источник - Wikipedia



Вычислительные машины

Ада Августа Лавлейс (Байрон)

Бэббидж выступил на съезде учёных в Турине. Луиджи Менабреа законспектировал его доклад, и опубликовал описание аналитической машины на французском языке.

Ада Байрон перевела эту статью на английский. Бэббидж был удивлён тем, как она разобралась в материале и предложил ей сделать примечания к переводу.

Примечания Ады Байрон по объему вдвое привысили оригинальную статью, эта работа стала первой опубликованной компьютерной программой, а Ада Байрон вошла в историю как первый программист.



Ада Августа Лавлейс (Байрон),
Источник - Wikipedia



Вычислительные машины

Ада Августа Лавлейс (Байрон)

Ада Лавлейс предложила несколько концепций, которые на многие годы опередили развитие вычислительное техники:

- **Машина общего назначения** - запрограммирована для выполнения бесконечного числа и неограниченного круга задач
- Функции машины не должны быть ограничены математикой и числами (например, работа со словами и символами)
- Идея **подпрограмм**
- **Искусственный интеллект.** "Аналитическая машина не претендует на создание чего-то своего. Она может выполнить любую команду, которую мы сумеем задать, но от нее нельзя ожидать вывода каких-либо аналитических соотношений или законов"



Вычислительные машины

Электронные машины

Технологии 19 века не позволяли создать многие из вышеперечисленных вариантов вычислительных машин относительно дешево. Но, с развитием электроники в первой половине 20 века эта проблема была решена. Повсеместно, во всем мире стали появляться более совершенные вычислительные электронно-механические устройства.

- **Модель K, 1940 год, Р.Штиблиц, Bell Labs** - электромеханические реле, не была полностью электронной. Не была универсальной, программируемой, предназначалась для определенной задачи.
- **Z3, Г.Цузе, 1941** - первая автоматически контролируемое, программируемое электрическое двоичное устройство. Но не была полностью электронной, также были переключатели - щёлкающие реле. Не пошла в производство, была разрушена после бомбардировок Берлина в 1943.
- **Компьютер Д.В.Атанасова** - Был первым электронным компьютером в мире (использовал электронные лампы). Но блоки памяти содержали механические врачающиеся барабаны. Не был программируемым и универсальным - был создан для решения систем линейных уравнений.
- **Машина Colossus I** - создана Максом Ньюманом, Томми Флауэрсом и Аланом Тьюрингом. Электронный, программируемый и работающий. Но не являлся компьютером общего назначения и тьюринг-полной машиной и также был узкоспециализированным - для взлома военных кодов Германии.
- **Mark I Говарда Айкена** - Построенный с участием IBM и введенный в эксплуатацию в мае 1944 года был программируемым, но электро-механическим, а не электронным.



Вычислительные машины

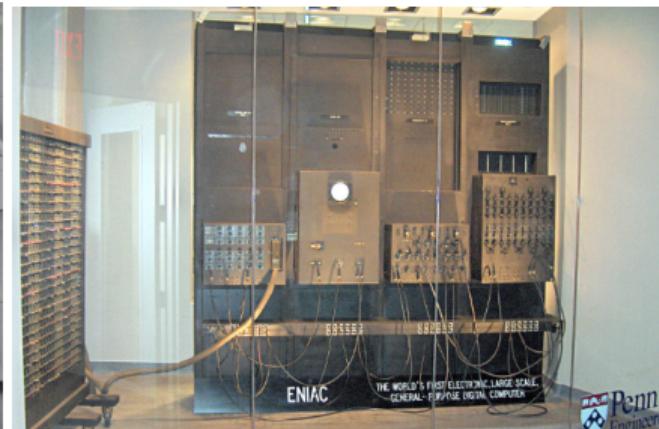
ENIAC - первый компьютер

ENIAC, построенный Джоном Мокли и Преспером Эккертом в ноябре 1945 года был первой машиной, включающей все черты современного компьютера.

- Полностью электронный (основанный на электронных лампах), быстрый
- Программируемый
- Компьютер общего назначения(теоретически мог решить любую задачу)



(6) Команда программистов ENIAC



(7) ENIAC



Вычислительные машины

Транзистор

Электронные лампы были дорогими, громоздкими, поглощали много энергии. Для компьютеров того времени их было необходимо очень много.

Наиболее значимым в этот период являлось изобретение транзисторов. За это, команда Bell Labs в лице **Уиллиама Шокли, Джон Бардин и Уолтера Браттейна** была удостоена Нобелевской премии по физике.



(8) Д.Бардин, У.Шокли, У.Браттейн



Вычислительные машины

Микрочип

В Texas Instruments (Джон Килби) и Fairchild Semiconductors (Роберт Нойс и Гордон Мур) был изобретен **микрочип**. Идея - поместить транзисторы и другие элементы на единую кремниевую пластину. Вместо связующих их проволочек было предложено пропечатать небольшие медные линии. В 2000 году за это открытие присуждена Нобелевская премия.

Производство таких схем стало надежным, массовым и дешевым. В результате, машины размером с комнату 1940-е годы были уменьшены за десятилетия до размеров одиночных шкафов. Вычислительная мощность машин стала удваиваться каждые два года.



0. План лекции
1. Понятие алгоритма
2. Вычислительные машины
3. Алгоритмы и программы
4. Алгоритмы и данные
5. Битовые операции



Алгоритмы и программы

Программы и двоичный код

Алгоритмы являются подробными и точными инструкциями по решению какой-либо задачи. Каждый алгоритм предназначается для некоторого исполнителя.

В реальной жизни исполнителями как правило являются люди или вычислительные машины. Например, инструкция по сборке мебели - это алгоритм предназначаемый для исполнителя-человека.

Однако, самые сложные задачи, связанные с обработкой, хранением и передачей информации человек умышленно поручил вычислительным машинам. Вычислительные машины оперируют лишь двумя состояниями связанными с наличием тока или его отсутствием.

Эта физическая сущность компьютеров заставила людей использовать двоичную систему счисления или **двоичный код**.



Алгоритмы и программы

Программы и двоичный код

Люди формулируют алгоритмы на понятном им языке, а компьютеры на языке двоичной системы счисления - с помощью нулей и единиц.

Для формализации и описания алгоритмов в более понятной для вычислительных машин манере были созданы **языки программирования**.

Алгоритм решения задачи, записанный на конкретном языке программирования называется **программой**. То есть, программа это более узкое понятие, а алгоритм более широкое. Программа - это один из способов записи алгоритма.

Программа – набор инструкций на специальном языке, которые могут быть выполнены компьютером.



Алгоритмы и программы

Машинный язык

Языки программирования можно условно разделить на несколько категорий.

Машинный язык

Программа на машинном языке читается компьютером напрямую, без промежуточных шагов.

Программа в машинном коде состоит из последовательности машинных инструкций в двоичном коде, которые содержат коды операций и ячейки памяти, участвующие в операции.

```
03 F3 0D 0A 14 01 3C 57 63 00 00 00 00 00 00 00 00 00  
00 01 00 00 00 40 00 00 00 73 09 00 00 00 00 64 B0  
00 47 48 64 01 00 53 28 02 00 00 00 73 0B 00 00  
00 48 65 6C 6C 6F 20 57 6F 72 6C 64 4E 28 00 00  
00 00 28 00 00 00 28 00 00 00 00 00 28 00 00 00  
00 73 0D 00 00 00 68 65 6C 6C 6F 77 6F 72 6C 64  
2E 70 79 74 08 00 00 00 00 3C 6D 6F 64 75 6C 65 3E  
01 00 00 00 73 00 00 00 00
```



(9) Пример двоичного кода

(10) Пример hex кода

(11) Пример программиста на машинном языке



Алгоритмы и программы

Ассемблер

Ассемблер (assembler)

Символьное представление машинного языка конкретного процессора.

Обычно каждая строка кода сборки производит одну машинную инструкцию.

Программирование на ассемблере медленное и подвержено ошибкам, но более эффективно с точки зрения производительности оборудования.

```
mov AH, 09h  
mov DX, offset message  
int 21h  
int 20h  
message DB 'Hello World',13,10,'$'
```

(12) Пример кода на ASM



(13) Пример программиста на ASM



Алгоритмы и программы

Языки программирования высокого уровня

Языки программирования высокого уровня (high-level programming languages)

Языки программирования, в котором используются операторы, состоящие из похожих на английские ключевых слов, таких как «FOR», «PRINT» или «IF».

Каждое утверждение соответствует нескольким инструкциям машинного языка.

```
program HelloWorld;  
begin  
    write('Hello, World!');  
end.
```

(14) Пример кода на Паскале (ЯВУ)



(15) Пример программиста на Паскале

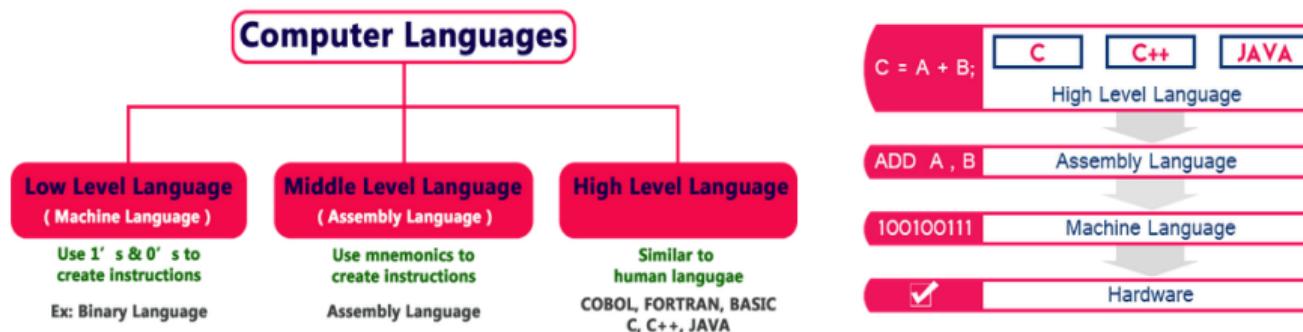


Алгоритмы и программы

Трансляция

Транслятор – программа, обеспечивающая перевод написанного нами исходного кода на внутренний язык компьютера.

Обычно выделяют трансляторы двух типов: **компилятор** и **интерпретатор**



(16) Трансляция, Источник



Алгоритмы и программы

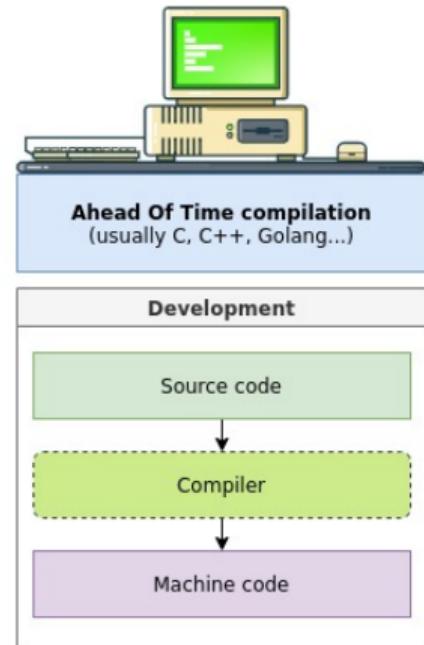
Компиляция

Компилятор — это программа, которая преобразует исходный код, написанный на одном из языков программирования, в исполняемый машинный код или в промежуточный код (например, байт-код).

Компиляторы выполняют анализ и оптимизацию кода, чтобы создать эффективный и быстрый исполняемый файл.

Основные функции компилятора:

- проверка синтаксиса исходного кода
- генерация машинного кода или байт-кода
- оптимизация кода для повышения производительности
- создание файлов объектного кода для компоновки с другими модулями



Компиляция, [источник](#)



Алгоритмы и программы

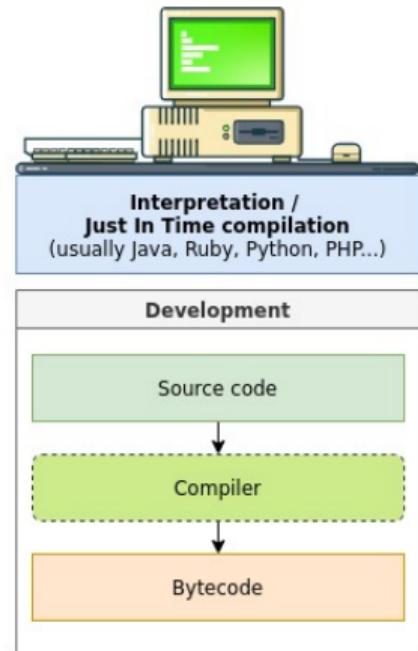
Интерпретация

Интерпретатор — это программа, которая выполняет исходный код программы построчно. В отличие от компилятора, интерпретатор не переводит программу в машинный код, а сразу выполняет инструкции.

Основные функции интерпретатора:

- чтение исходного кода программы
- последовательное выполнение каждой строки кода
- обработка ошибок и исключений во время выполнения программы

Известные интерпретируемые языки программирования - Python, Java.



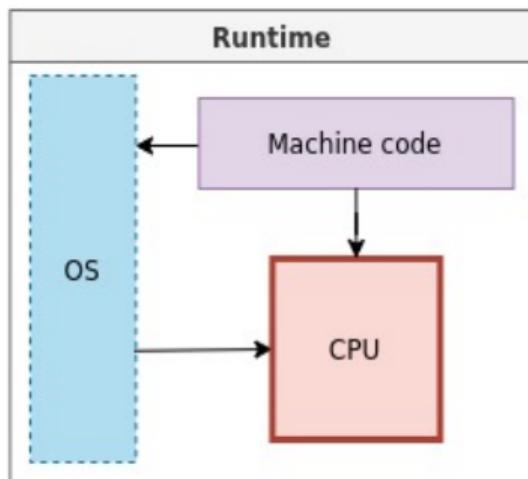
Интерпретация, [источник](#)



Алгоритмы и программы

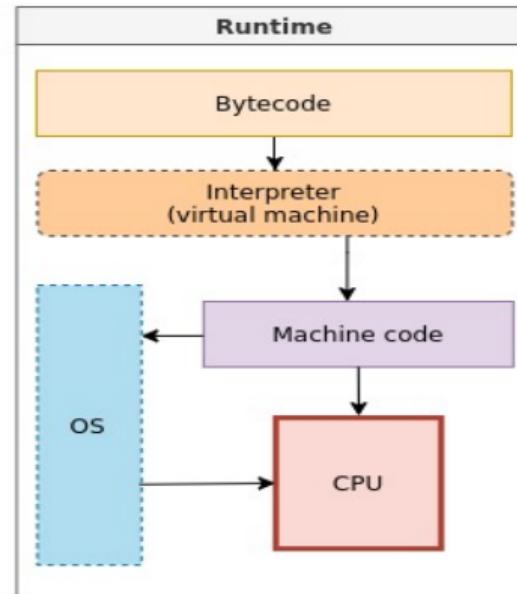
Компиляция vs Интерпретация

Компиляция vs Интерпретация в процессе выполнения кода



→ Input / Output

(17) После компиляции



(18) После интерпретации



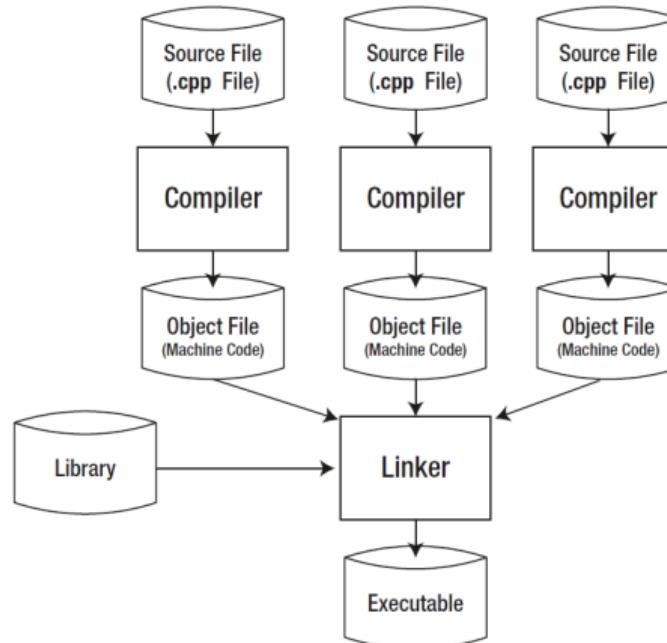
Алгоритмы и программы

Трансляция исходного кода в машинный в C/C++

The contents of header files will be included before compilation.

Each .cpp file will result in one object file.

The linker will combine all the object files plus necessary library routines to produce the executable file.



Источник - C++20 from novice to professional



Алгоритмы, вычислительные машины и программы

План лекции

0. План лекции
1. Понятие алгоритма
2. Вычислительные машины
3. Алгоритмы и программы
4. Алгоритмы и данные
5. Битовые операции



Алгоритмы и данные

Представление данных

Информация, с которой работает алгоритм, представлена в памяти компьютера в виде последовательностей нулей и единиц.

Наименьшая единица хранения - **бит/bit/binary digit** (0 или 1).

Наименьшая адресуемая единица хранения - **байт/byte** (последовательность из 8 бит).

Процессор обрабатывает информацию манипулируя блоками из фиксированного количества бит. Такой блок называется **машинное слово (word)**. В ранних компьютерах размеры машинного слова составляли 12 или 16 бит. В современных процессорах это значение обычно составляет 64 бита.

Двоичные последовательности достаточно громоздки для восприятия человеком. Для упрощения работы с данными один байт обычно делится на две **двоичные тетрады (tetrade, half-byte, nibble)**.



Алгоритмы и данные

Представление данных



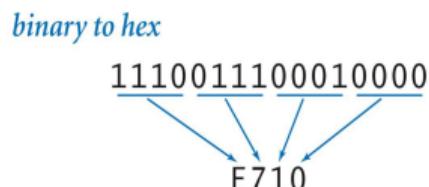
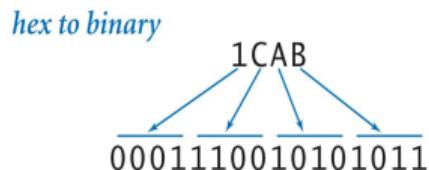
Источник - этот курс



Алгоритмы и данные

Представление данных

Каждую тетраду/полубайт удобно представлять в виде одной цифры в шестнадцатиричной системе счисления, потому что для ее представления нужно как раз 4 бита. Двоичные данные, записанные в таком виде обычно называют hex кодами (hexadecimal)



Hex-binary conversion examples

<i>decimal</i>	<i>binary</i>	<i>hex</i>
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Representations of integers from 0 to 15



Алгоритмы и данные

Память

Память компьютера можно рассматривать как очень большую ленту из пронумерованных байтов. Каждый ячейка содержит данные – значение от 0 до FF (две тетрады).

Эти значения могут интерпретироваться по-разному в зависимости от того, данные какой природы мы ожидаем в этой памяти.

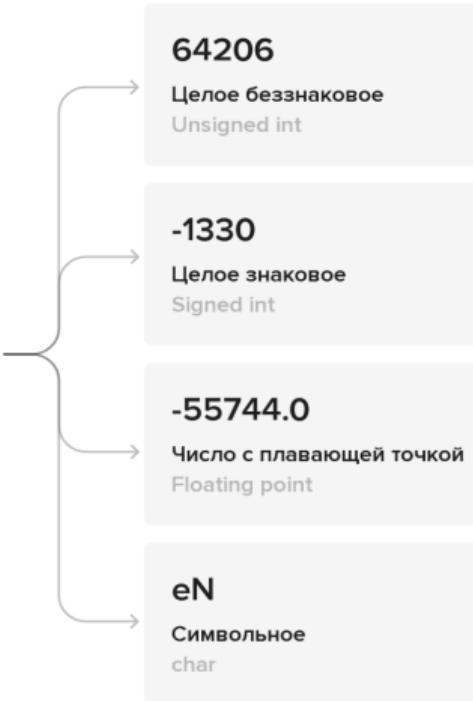
Номер байта	Адрес байта	Содержимое
Байт 0	0x00000000	E5
Байт 1	0x00000001	10
Байт 2	0x00000002	FF
Байт 3	0x00000003	15
Байт 4	0x00000004	7B
Байт 5	0x00000005	AC
Байт 6	0x00000006	0D
...		
Байт 10	0x0000000A	A0
Байт 11	0x0000000B	EC
Байт 12	0x0000000C	52
...		
Байт ??	0xfffffffffd	00
Байт ??	0xfffffffffe	3D
Байт ??	0xffffffffff	4D



Алгоритмы и данные

Интерпретация двоичных данных

1111101011001110
Двоичное представление
Binary



Источник - Sedgewick & Wayne + этот курс



Алгоритмы и данные

Переменная и тип данных

Работая с данными в языках программирования необходимо знать начальный адрес данных в памяти, размер объекта данных и природу данных, записанных в этой памяти.

Манипулировать этими компонентами напрямую - неудобно и непрозрачно для человека. Именно поэтому работа с данными осуществляется с помощью **имён переменных и типов данных**.

Имя переменной – символьное обозначение адреса ячейки памяти, по которому хранится некоторое значение. Символьное имя позволяет понимать, о каких данных идет речь и упростить работу с ними.

Каждой переменной обычно соответствует некоторый **тип данных**.



Алгоритмы и данные

Тип данных

Тип данных определяет:

- какого размера будет объект данного типа
- как будет интерпретироваться двоичная информация внутри объекта этого типа
- какие операции можно будет делать с объектами этого типа

Тип данных	Размер (в байтах)	Возможные значения
int	4	-2147483648 to 2147483647
unsigned int	4	0 to 4294967295

Источник - не помню, надо сменить картинку



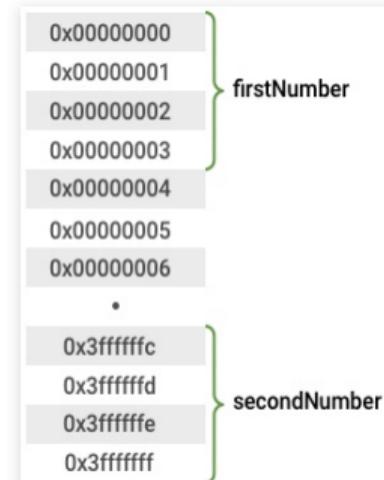
Алгоритмы и данные

Переменные в памяти

```
#include <stdio.h>

int main()
{
    int firstNumber = 1;
    int secondNumber = 2;
    int sum = firstNumber + secondNumber;

    printf("%d + %d = %d", firstNumber, secondNumber, sum);
    return 0;
}
```





Алгоритмы и данные

Типы данных

Существуют различные категории типов данных:

Целые числа (int/integer)

Беззнаковые целые числа (unsigned)

Числа со знаком (signed)

Вещественные числа/числа с плавающей точкой (floating point):

числа с обычной точностью (float)

числа с двойной точностью (double)

Символы (character):

Символы (char)

Строки



Алгоритмы и данные

Целочисленные типы данных

Беззнаковые целые числа (`unsigned`)

Стратегия хранения беззнаковых целых чисел достаточно тривиальна – хранится само число в двоичном виде без всяких модификаций.

Пример: `short c = 10` —> 00000000 00001010

<i>bits</i>	<i>smallest</i>	<i>largest</i>
4	0	15
16	0	65,535
32	0	4,294,967,295
64	0	18,446,744,073,709,551,615

Representable unsigned integers

Источник - Sedgewick & Wayne



Алгоритмы и данные

Целочисленные типы данных

В самом простом случае для представления отрицательных чисел в памяти компьютера можно использовать отдельных бит для знака (**sign bit**). На практике знаковым битом всегда выбирается самый левый (старший) бит.

Пример: представим в 1 байте числа 6 и -6

+ 6 = 00000110

- 6 = 100000110

Такой способ представления называется **signed magnitude**.

Преимущество: простота в понимании и использовании для человека

Недостаток: высокая сложность в использовании компьютером: сломанная арифметика, два нуля и тд

12 in binary is

00001100

-8 in binary (you suppose) is

10001000

If you now “add” these together, you get 10010100



Алгоритмы и данные

Целочисленные типы данных

Современные компьютеры используют другой подход: **дополнительный код (two's complement)** представление. В данном представлении мы получаем любое отрицательное число по следующему алгоритму:

0. Переводим рассматриваемое положительное число в двоичную форму
1. Инвертируем все биты числа, получаем так называемую **one's complement form**
2. Добавляем к полученному числу 1, получаем **two's complement form**

Пример

Переводим +8 в двоичную систему —> 00001000

Инвертируем биты 00001000 —> 11110111

Добавляем единицу 11110111 + 1 = 11111000



Алгоритмы и данные

Целочисленные типы данных

Signed Binary Numbers

Decimal	Signed Magnitude	Signed-1's Complement	Signed-2's Complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	—
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	—	—	1000

decimal	binary
0	0000000000000000
1	0000000000000001
2	0000000000000010
3	0000000000000011
...	...
32765	0111111111111101
32766	0111111111111110
32767	0111111111111111
-32768	1000000000000000
-32767	1000000000000001
-32766	1000000000000010
-32765	1000000000000011
...	...
-3	1111111111111101
-2	1111111111111110
-1	1111111111111111
16-bit integers (two's complement)	



Алгоритмы и данные

Возможные значения знаковых целочисленных переменных

16-bit	<i>smallest</i>	– 32,768
	<i>largest</i>	32,767
32-bit	<i>smallest</i>	– 2,147,483,648
	<i>largest</i>	2,147,483,647
64-bit	<i>smallest</i>	– 9,223,372,036,854,775,808
	<i>largest</i>	9,223,372,036,854,775,807

Representable two's complement integers

Источник - Sedgewick & Wayne



Алгоритмы и данные

Числа с плавающей точкой

Для представления дробных чисел компьютер имеет в своем распоряжении всё те же нули и единицы, однако задача усложняется. вещественные числа представляются в памяти компьютера как **числа с плавающей точкой/floating point**.

Число с плавающей точкой состоит из трёх частей:

- **Знак (sign)**. Ничего необычного, знак представляется единственным битом, где 0 - положительное число, 1 - отрицательное. Быстрая проверка на отрицательность.
- **Двоичная экспонента (binary exponent)**. Хранит знаковое целое число.
 - 5 бит (для 16 битных чисел)
 - 8 бит (для 32 битных чисел)
 - 11 бит (для 64 битных чисел)
- **Мантисса (significand, mantissa, binary fraction)**. Хранит знаковое целое число.
 - 10 бит (для 16 битных чисел)
 - 23 бита (для 32 битных чисел)
 - 53 бита (для 64 битных чисел)

Все эти компоненты представлены в двоичном формате.



Алгоритмы и данные

Числа с плавающей точкой

IEEE 754 to decimal

1010011010000000

$$\begin{array}{c} \downarrow \\ -2^{9-15} \times 1.\underline{101}_2 = -2^{-6} (1 + 2^{-1} + 2^{-3}) = -0.0253906250_{10} \end{array}$$

Decimal to IEEE 754

$$100.25_{10} = 2^6 \times (1 + 2^{-1} + 2^{-4} + 2^{-8})$$

*largest power of 2
less than or equal to 100.25*

*binary representation of
100.25 / 2⁶ = 1.56640625*

$$+ 2^{21-15} \times 1.\underline{10010001}_2$$

0101011001000100

Floating point–decimal conversion examples

Источник - Sedgewick & Wayne



Алгоритмы и данные

Символы

Символы не могут быть напрямую представлены в памяти компьютера, поэтому они кодируются числами и создаются таблицы соответствия между кодами и самими символами. Такие таблицы называются **кодировками**.

В 1960х годах для представления символов была создана таблица ASCII (American Standard Code for Information Interchange).

Каждый код в этой таблице - 7 битное число, всего 128 разных варианта кодов.

0-31 - символы, которые нельзя напечатать на клавиатуре

65-90 - латинские символы в верхнем регистре

97-122 - латинские символы в нижнем регистре



Алгоритмы и данные

Символы

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	Ø	96	60	'
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	{	72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	Ø	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	Ø	127	7F	□

Источник - ?



Алгоритмы, вычислительные машины и программы

План лекции

0. План лекции
1. Понятие алгоритма
2. Вычислительные машины
3. Алгоритмы и программы
4. Алгоритмы и данные
5. Битовые операции



Битовые операции

Основные битовые операторы

Из школьных курсов все мы помним о существовании булевых операций, таких как конъюнкция (AND), дизъюнкция (OR), отрицание (NOT) и исключающее или (XOR).

Вход (Input)		Результат (Output)
A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

(19) Конъюнкция (И/AND)

Вход (Input)		Результат (Output)
A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

(20) Дизъюнкция (ИЛИ/OR)



Битовые операции

Основные битовые операторы

Из школьных курсов все мы помним о существовании булевых операций, таких как конъюнкция (AND), дизъюнкция (OR), отрицание (NOT) и исключающее или (XOR).

Вход (Input)		Результат (Output)
A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

(21) Исключающее ИЛИ/XOR

Вход (Input)	Результат (Output)
A	not A
0	1
1	0

(22) Отрицание (НЕ/NOT)



Битовые операции

Числовые литералы

Для удобства работы с числами языки программирования обычно содержат разные варианты задания числовых констант ([литералов/literal](#)).

```
#include <stdio.h>

int main()
{
    unsigned int aBin = 0b1010;
    unsigned int aOct = 0101;
    unsigned int aHex = 0xFF;

    printf("Binary      0b1010 = %d\n", aBin);
    printf("Oct         0101 = %d\n", aOct);
    printf("Hex         0xFF = %d\n", aHex);

    return 0;
}
```

(23) Пример числовых литералов C/C++



Битовые операции

Числовые литералы

Для удобства работы с числами языки программирования обычно содержат разные варианты задания числовых констант ([литералов/literal](#)).

```
#include <stdio.h>

int main()
{
    unsigned int aBin = 0b1010;
    unsigned int aOct = 0101;
    unsigned int aHex = 0xFF;

    printf("Binary      0b1010 = %d\n", aBin);
    printf("Oct        0101 = %d\n", aOct);
    printf("Hex        0xFF = %d\n", aHex);

    return 0;
}
```

(24) Пример числовых литералов C/C++

```
Binary      0b1010 = 10
Oct        0101 = 65
Hex        0xFF = 255
```

(25) Результат примера



Битовые операции

Пример битовых операций

Побитовые операторы применяют упомянутые выше логические операции не к одному биту, а сразу к целой последовательности бит.

```
— □ ×

#include <stdio.h>

int main()
{
    unsigned int a = 9; // 0b1001
    unsigned int b = 0b111;

    printf("Bitwise AND result = %d\n", a & b);
    printf("Bitwise OR  result = %d\n", a | b);
    printf("Bitwise XOR result = %d\n", a ^ b);
    printf("Bitwise NOT result = %d\n", ~ a);

    return 0;
}
```

(26) Пример числовых литералов C/C++



Битовые операции

Пример битовых операции

Побитовые операторы применяют упомянутые выше логические операции не к одному биту, а сразу к целой последовательности бит.

```
#include <stdio.h>

int main()
{
    unsigned int a = 9; // 0b1001
    unsigned int b = 0b111;

    printf("Bitwise AND result = %d\n", a & b);
    printf("Bitwise OR  result = %d\n", a | b);
    printf("Bitwise XOR result = %d\n", a ^ b);
    printf("Bitwise NOT result = %d\n", ~ a);

    return 0;
}
```

(27) Пример битовых операции C/C++

```
Bitwise AND result = 1
Bitwise OR  result = 15
Bitwise XOR result = 14
Bitwise NOT result = -10
```

(28) Результат примера



Битовые операции

Битовые маски

Битовая маска - это специальное число, которое используется для работы с конкретными битами рассматриваемых данных. Такой способ работы приносит компактность и элегантность некоторых операций.



```
#include <stdio.h>
int main()
{
    const unsigned int MASK1 = 0x0000000F;
    const unsigned int MASK2 = 0x000000FF;
    const unsigned int MASK3 = 0x000F0000;

    unsigned int a = 0x0ACE;

    printf("%d & %d = %d\n", a, MASK1, a & MASK1);
    printf("%d & %d = %d\n", a, MASK2, a & MASK2);
    printf("%d & %d = %d\n", a, MASK3, a & MASK3);

    return 0;
}
```

(29) Пример битовых масок



Битовые операции

Битовые маски

Битовая маска - это специальное число, которое используется для работы с конкретными битами рассматриваемых данных. Такой способ работы приносит компактность и элегантность некоторых операций.

```
#include <stdio.h>
int main()
{
    const unsigned int MASK1 = 0x0000000F;
    const unsigned int MASK2 = 0x000000FF;
    const unsigned int MASK3 = 0x000F0000;

    unsigned int a = 0x0ACE;

    printf("%d & %d = %d\n", a, MASK1, a & MASK1);
    printf("%d & %d = %d\n", a, MASK2, a & MASK2);
    printf("%d & %d = %d\n", a, MASK3, a & MASK3);

    return 0;
}
```

(30) Пример битовых масок

```
2766 & 15 = 14
2766 & 255 = 206
2766 & 983040 = 0
```

(31) Результат примера



Битовые операции

Битовые сдвиги

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int a = 3;
6     int b = -3;
7
8     printf("Unsigned left shifts\n");
9     printf("%d << %d = %d \n", a, 1, a << 1);
10    printf("%d << %d = %d \n", a, 2, a << 2);
11
12    printf("Signed left shifts\n");
13    printf("%d << %d = %d \n", b, 1, b << 1);
14    printf("%d << %d = %d \n", b, 2, b << 2);
15
16    printf("Unsigned right shifts\n");
17    printf("%d >> %d = %d \n", a, 1, a >> 1);
18    printf("%d >> %d = %d \n", a, 2, a >> 2);
19
20    printf("Signed right shifts\n");
21    printf("%d >> %d = %d \n", b, 1, b >> 1);
22    printf("%d >> %d = %d \n", b, 2, b >> 2);
23
24    return 0;
25 }
```

- □ ×



Битовые операции

Битовые сдвиги

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int a = 3;
6     int b = -3;
7
8     printf("Unsigned left shifts\n");
9     printf("%d << %d = %d \n", a, 1, a << 1);
10    printf("%d << %d = %d \n", a, 2, a << 2);
11
12    printf("Signed left shifts\n");
13    printf("%d << %d = %d \n", b, 1, b << 1);
14    printf("%d << %d = %d \n", b, 2, b << 2);
15
16    printf("Unsigned right shifts\n");
17    printf("%d >> %d = %d \n", a, 1, a >> 1);
18    printf("%d >> %d = %d \n", a, 2, a >> 2);
19
20    printf("Signed right shifts\n");
21    printf("%d >> %d = %d \n", b, 1, b >> 1);
22    printf("%d >> %d = %d \n", b, 2, b >> 2);
23
24    return 0;
25 }
```

- □ ×

Unsigned left shifts

$3 << 1 = 6$

$3 << 2 = 12$

Signed left shifts

$-3 << 1 = -6$

$-3 << 2 = -12$

Unsigned right shifts

$3 >> 1 = 1$

$3 >> 2 = 0$

Signed right shifts

$-3 >> 1 = -2$

$-3 >> 2 = -1$

- □ ×



Битовые операции

Задачи

Задача про пары чисел

Дана последовательность A из n различных целых чисел.

Про последовательность известно, что каждое число кроме одного встречается в ней дважды. Найдите число, у которого нет пары.

Ограничения на входные данные:

$$n \in \mathbb{N}, \quad 1 \leq n \leq 10^9$$

$$A = a_1, a_2, \dots, a_n$$

$$10^{-9} \leq a_i \leq 10^9 \quad \forall i \in \mathbb{N}, \quad 1 \leq i \leq n$$

Количество дополнительной памяти, используемой алгоритмом, не должно превышать 1МВ, время работы 1 секунда.



Слайд 63

Слайд 63

Дополнительный слайд, чтобы общее их количество стало равно 64.



Слайд 64

Слайд 64

Дополнительный слайд, чтобы общее их количество стало равно 64.