Представление информации в памяти компьютера

Единицы измерения информации

- Единица информации называется **битом**. Термин **"бит"** предложен как аббревиатура от английского словосочетания **"Binary digit"**, которое переводится как **"двоичная цифра"**. 1 бит информации - количество информации, посредством которого выделяется одно из двух равновероятных состояний объекта.
- В компьютерной технике бит соответствует физическому состоянию носителя информации: намагничено - не намагничено, есть отверстие - нет отверстия. При этом одно состояние принято обозначать цифрой **0**, а другое - цифрой **1**. Выбор одного из двух возможных вариантов позволяет также различать логические истину и ложь. Последовательностью битов можно закодировать текст, изображение, звук или какую-либо другую информацию. Такой метод представления информации называется двоичным кодированием (**binary encoding**).

Единицы измерения информации

- В информатике часто используется величина, называемая **байтом** (*byte*) и равная **8 битам**. И если бит позволяет выбрать один вариант из двух возможных, то байт, соответственно, **1** из **256** (2^8).
- В большинстве современных **ЭВМ** при кодировании **каждому символу соответствует своя последовательность из восьми нулей и единиц, т. е. байт**
- Наряду с байтами для измерения количества информации используются более крупные единицы:
- 1 Кбайт (один килобайт) = **2^{10} байт** = **1024 байта;**
  1 Мбайт (один мегабайт) = **2^{10} Кбайт** = **1024 Кбайта** = **2^{20} байт;**
  1 Гбайт (один гигабайт) = **2^{10} Мбайт** = **1024 Мбайта** = **2^{30} байт.**

  1 Тбайт (один терабайт) = **2^{10} Гбайт** = **1024 Гбайт** = **2^{40} байт**,
  1 Пбайт(один петабайт) = **2^{10} Тбайт** = **1024 Тбайт** = **2^{50} байт**.

Представление числовой информации

- Числовая информация была первым видом информации, который начали обрабатывать ЭВМ, и долгое время она оставалась единственным видом. Поэтому не удивительно, что в современном компьютере существует большое разнообразие типов и представлений чисел. Прежде всего, это **целые** и **вещественные числа**, которые по своей сути и по представлению в машине различаются очень существенно. Целые числа, в свою очередь, делятся на числа **со знаком** и **без знака**, имеющие уже не столь существенные различия. Наконец, вещественные числа имеют два способа представления – **с фиксированной** и **с плавающей запятой**, правда, первый способ сейчас представляет в основном исторический интерес.

Беззнаковые целые числа

- Беззнаковые целые числа представляются в машине наиболее просто. Для этого достаточно перевести требуемое число в двоичную форму и дополнить полученный результат слева нулями до стандартной разрядности.
- Например, **восьмиразрядное** (1 байт) число **14** будет иметь вид **0000 1110**. Это же самое число в **16** - разрядном представлении будет иметь слева еще **8** нулей.
- Просто определить *минимальное* и *максимальное* значение чисел для **N**-разрядного беззнакового целого: минимальное состоит из одних нулей, а значит, при любом **N** равняется нулю; максимальное, напротив, образовано одними единицами и, разумеется, для разных **N** различно. Для вычисления **максимально допустимого значения** обычно используют формулу: **Max = 2^N - 1.**
- Если выполнить следующие операции: **255** + 1 и **0** - **1**.
- Мысленно представим себе, что при осуществлении операции слева существует еще один дополнительный (девятый) разряд. И , отбросив несуществующий дополнительный разряд, получаем несколько странный, но действительно имеющий место на практике результат: **255 + 1 = 0. А 0-1=255.**

Переполнение

- Если теперь внимательно посмотреть на полученные результаты, то можно заметить, что при последовательном увеличении на единицу мы доходим до максимального значения и возвращаемся к минимальному. При вычитании единицы получается обратная картина. Подобные свойства поведения чисел можно отобразить вместо традиционного отрезка математической числовой оси замкнутой окружностью.
- Обсуждаемая проблема выхода за отведенную разрядную сетку машины занимает важное место в реализации компьютерной арифметики и называется **переполнением**. Ситуация эта не совсем нормальная и для получения достоверных результатов ее следует избегать. Положение осложняется тем, что для процессора описанные результаты не являются чем-то "угрожающим", и он "спокойно" продолжает вычисления. Единственная тонкость заключается в том, что сам факт переполнения всегда фиксируется путем установки в единицу специального управляющего бита, который последующая программа имеет возможность проанализировать. Образно говоря, процессор "замечет" переполнение, но предоставляет программному обеспечению право принять решение реагировать на него или проигнорировать.

Целые числа со знаком

- Для того, чтобы различать положительные и отрицательные числа, в двоичном представлении чисел выделяется **знаковый разряд**. По традиции для кодирования знака используется самый старший бит, причем нулевое значение в нем соответствует знаку "+", а единичное – минусу. Подчеркнем, что с точки зрения описываемой системы кодирования число ноль является положительным, т.к. все его разряды, включая и знаковый, нулевые.
- Представление положительных чисел при переходе от беззнаковых чисел к целым со знаком сохраняется, за исключением того, что теперь для собственно числа остается на один разряд меньше.
- Первое, что приходит в голову, это кодировать отрицательные значения точно так же, как и положительные, только добавлять в старший бит единицу. **Подобный способ кодирования называется прямым кодом**.

Дополнительный код числа

- В его основе лежит запись отрицательных чисел в виде $2^N - |m|$, где **N**, как обычно, количество двоичных разрядов, а **m** – значение числа. Поскольку фактически вместо числа теперь записывается его дополнение до некоторой характерной величины 2^N, то такой **код** назвали **дополнительным**.
- Однако способ расчета, вытекающий непосредственно из определения, не слишком хорош, поскольку требует от конструкции процессора дополнительного разряда. Поэтому для практического получения кода отрицательных чисел используется следующий эквивалентный алгоритм. Для преобразования отрицательного числа в дополнительный код необходимо:
- Модуль числа перевести в двоичную форму.
- Проинвертировать каждый разряд получившегося кода, т.е. заменит единицы нулями, а нули – единицами (полученный **код** называется **обратным**).
- К обратному коду прибавить единицу.
- Пример 1: Перевести число **– 8** в двоичный **8 - разрядный код**.
- Возьмем модуль числа ($8_{10} = 1000_2$) и дополним его до необходимого числа разрядов нулями слева: 0000 1000.
- Теперь проинвертируем: 1111 0111.
- Прибавим единицу. Получим окончательный ответ: 11111000
- Для проверки правильности перевода можно сложить последнее число с исходным и убедиться в том, что результат будет нулевым (единицей переноса из старшего разряда, как обычно, пренебрегаем).
- Проведем сопоставление целых чисел без знака и со знаком. Результат сравнения чисел со знаком и без него состоит в том, что общее количество их значений одинаково, но их диапазоны сдвинуты вдоль числовой оси.

Представление вещественных чисел

- Принципиальное отличие между вещественными и целыми числами: целые числа **дискретны**, и отсюда (если не брать во внимание эффект переполнения) каждому целому числу соответствует уникальный двоичный код; вещественные числа, напротив, непрерывны, а значит, не могут быть полностью корректно перенесены в дискретную по своей природе вычислительную машину. Это означает, что некоторые вещественные числа, незначительно отличающиеся друг от друга, могут иметь одинаковый код.
- Существует два способа представления вещественных чисел: с фиксированной и с плавающей запятой.
- В старых машинах, использовавших фиксированное размещение запятой, положение последней в разрядной сетке **ЭВМ** было заранее обусловлено – раз и навсегда для всех чисел и для всех технических устройств. Поэтому отпадала необходимость в каком-либо способе ее указания во внутреннем представлении чисел. Все вычислительные алгоритмы были заранее "настроены" на это фиксированное размещение.

Представление с плавающей запятой

- любое число **A** в системе счисления с основанием **Q** можно записать в виде:
- $A = (\pm M) \times Q^{\pm P}$.
- где **M** называют **мантиссой**, а показатель степени **P** - **порядком числа**.
- Например, $0{,}03 = 3 \times 10^{-2} = 30 \times 10^{-3} = 0{,}3 \times 10^{-1} = 0{,}03 \times 10^{0} = \ldots$ То есть, представление числа с плавающей запятой не является единственным. Поэтому договорились для выделения единственного варианта записи числа считать, что **мантисса всегда меньше единицы, а ее первый разряд содержит отличную от нуля цифру** – в нашем примере обоим требованиям удовлетворит только число $0{,}3 \times 10^{-1}$.
- Описанное представление чисел называется **нормализованным** и является единственным. *Любое число может быть нормализовано.* Особо подчеркнем, что требования к нормализации чисел вводятся исходя из соображений обеспечения максимальной точности их представления.

Представление с плавающей запятой

- Все сказанное о нормализации можно применять и к двоичной системе:
- $A = (\pm M) \times 2^{\pm P}$.
- Например: $\mathbf{-3_{10}} = \mathbf{-0{,}11} \times \mathbf{2^{10}}$, **M** = **0,11** и **P** = **10**.
- Существенно, что двоичная мантисса всегда начинается с единицы. Поэтому во многих ЭВМ эта единица не записывается в ОЗУ, что позволяет сохранить еще один дополнительный разряд мантиссы (так называемая **скрытая единица**).
- Арифметика чисел с плавающей запятой оказывается заметно сложнее, чем с фиксированной. Например, чтобы сложить два числа с плавающей запятой, требуется предварительно привести их к представлению, когда оба порядка равны; такую процедуру принято называть **выравниванием порядков**. Кроме того, в результате вычислений нормализация часто нарушается, а значит необходимо ее восстанавливать. Тем не менее, вычислительные машины со всем этим великолепно умеют автоматически справляться, и именно такой способ вычислений лежит в основе работы современных компьютеров.

Представление с плавающей запятой

- при использовании метода представления вещественных чисел с плавающей запятой фактически хранится два числа: **мантисса и порядок.** Разрядность первой части определяет **точность вычислений**, а второй – **диапазон представления чисел**.
- Для того чтобы сохранить максимальную точность, вычислительные машины почти всегда хранят мантиссу в нормализованном виде, что означает, что мантисса в данном случае есть число, лежащее между $1_{(10)}$ и $2_{(10)}$ ($1 < M < 2$). Способ хранения мантиссы с плавающей точкой подразумевает, что двоичная запятая находится на фиксированном месте. Фактически подразумевается, что двоичная запятая следует после первой двоичной цифры, т.е. нормализация мантиссы делает единичным первый бит, помещая тем самым значение между единицей и двойкой. Место, отводимое для числа с плавающей точкой, делится на два поля. Одно поле содержит знак и значение мантиссы, а другое содержит знак и значение порядка.

Вещественные типы данных

Тип	Диапазон	Байты
float	3.4e–38 … 3.4e+38	4
double	1.7e–308 … 1.7e+308	8
long double	3.4e–4932 … 3.4e+4932	10

Для типа double:

S	Смещенный порядок	Мантисса
63	62..52	51..0

Представление чисел с плавающей запятой

- Для упрощения вычислений и сравнения действительных чисел значение порядка в ЭВМ хранится в виде **смещенного числа**, т.е. к настоящему значению порядка перед записью его в память прибавляется смещение. Смещение выбирается так, чтобы минимальному значению порядка соответствовал нуль. Например, для типа Double порядок занимает 11 бит и имеет диапазон от 2^{-1023} до 2^{1023}, поэтому смещение равно $1023_{(10)} = 1111111111_{(2)}$. Наконец, бит с номером 63 указывает на знак числа.

  Алгоритм для получения представления действительного числа в памяти ЭВМ:

- Перевести модуль данного числа в двоичную систему счисления;
- нормализовать двоичное число, т.е. записать в виде $M \times 2^p$, где M — мантисса (ее целая часть равна $1_{(2)}$) и p — порядок, записанный в десятичной системе счисления;
- прибавить к порядку смещение и перевести смещенный порядок в двоичную систему счисления;
- учитывая знак заданного числа (0 — положительное; 1 — отрицательное), выписать его представление в памяти ЭВМ.

Пример:

Запишем код числа -312,3125.

- Двоичная запись модуля этого числа имеет вид 100111000,0101.
- Проведем нормализацию, получим $100111000{,}0101 = 1{,}001110000101 \times 2^{8}$.
- Находим смещенный порядок: 8 + 1023 = 1031. Далее имеем $1031_{(10)} = 10000000111_{(2)}$.
-

1	10000000111	0011100001010000000000000000000000000000000000000000
63	62 .. 52	51 ..0

- Очевидно, что более компактно полученный код стоит записать следующим образом: $C07385000000000_{(16)}$.

Обратный переход от кода действительного числа к самому числу

- Пусть дан код $3FEC6000000000000_{(16)}$
- Прежде всего замечаем, что это код положительного числа, поскольку в разряде с номером 63 записан нуль. Получим порядок этого числа: $01111111110_{(2)} = 1022_{(10)}$; 1022 - 1023 = -1.
- Число имеет вид $1{,}1100011 \times 2^{-1}$ или 0,11100011.
- Переводом в десятичную систему счисления получаем 0,88671875.

Представление текстовой информации

- Текстовая информация, как и любая другая, хранится в памяти компьютера в двоичном виде. Для этого каждому символу ставится в соответствие некоторое неотрицательное число, называемое **кодом символа**, и это число записывается в память ЭВМ в двоичном виде. Конкретное соответствие между символами и их кодами называется **системой кодировки**.
- В персональных компьютерах обычно используется система кодировки **ASCII (American Standard Code for Information Interchange** - американский стандартный код для обмена информации). Он введен в 1963 г. и ставит в соответствие каждому символу семиразрядный двоичный код. Легко определить, что в коде ASCII можно представить 128 символов.
- В системе ASCII закреплены две таблицы кодирования **базовая** и **расширенная**. Базовая таблица закрепляет значения кодов от 0 до 127, а расширенная относится к символам с номерами от 128 до 255. Первые 32 кода базовой таблицы, начиная с нулевого, отданы производителям аппаратных средств. В этой области размещаются управляющие коды, которым не соответствуют ни какие символы языков. Начиная с 32 по 127 код размещены коды символов английского алфавита, знаков препинания, арифметических действий и вспомогательных символов.

Таблица ASCII

[	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺ 1	☻ 2	♥ 3	♦ 4	♣ 5	♠ 6	• 7	8	○ 9	10	11	12	13	14	15
1	► 16	◄ 17	18	! 19	20	21	‗ 22	23	↑ 24	↓ 25	→ 26	← 27	28	↔ 29	▲ 30	▼ 31
2	32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	}] 125	~ 126	⌂ 127
8	Ç 128	ü 129	é 130	â 131	ä 132	à 133	å 134	ç 135	ê 136	ë 137	è 138	ï 139	î 140	ì 141	Ä 142	Ă 143
9	É 144	æ 145	Æ 146	ô 147	ö 148	ò 149	û 150	ù 151	ÿ 152	Ö 153	Ü 154	¢ 155	£ 156	¥ 157	₧ 158	ƒ 159
A	160	í 161	ó 162	ú 163	ñ 164	Ñ 165	ª 166	º 167	¿ 168	⌐ 169	¬ 170	½ 171	¼ 172	¡ 173	« 174	» 175
B	░ 176	▒ 177	▓ 178	| 179	┤ 180	╡ 181	╢ 182	╖ 183	╕ 184	╣ 185	║ 186	╗ 187	╝ 188	╜ 189	╛ 190	┐ 191
C	└ 192	┴ 193	┬ 194	├ 195	─ 196	┼ 197	╞ 198	╟ 199	╚ 200	╔ 201	╩ 202	╦ 203	╠ 204	═ 205	╬ 206	╧ 207
D	╨ 208	╤ 209	╥ 210	╙ 211	╘ 212	╒ 213	╓ 214	╫ 215	216	┘ 217	┌ 218	▉ 219	▄ 220	▌ 221	▐ 222	▀ 223
E	α 224	ß 225	Γ 226	π 227	Σ 228	σ 229	µ 230	τ 231	Φ 232	Θ 233	Ω 234	δ 235	∞ 236	φ 237	ε 238	∩ 239
F	≡ 240	± 241	≥ 242	≤ 243	⌠ 244	⌡ 245	÷ 246	≈ 247	° 248	∙ 249	· 250	√ 251	ⁿ 252	² 253	■ 254	□ 255

Кодировка символов, предложенная IBM (соответствует ASCII - кодировке)

Универсальная система кодирования текстовых данных

- Если проанализировать организационные трудности, связанные с созданием единой системы кодирования текстовых данных, то можно прийти к выводу, что они вызваны ограниченным набором кодов (256). В то же время, очевидно, что если, кодировать символы не восьмиразрядными двоичными числами, а числами с большим разрядом то и диапазон возможных значений кодов станет на много больше. Такая система, основанная на 16-разрядном кодировании символов, получила название **универсальной - UNICODE**. Шестнадцать разрядов позволяют обеспечить уникальные коды для 2^{16}=65 536 различных символов - этого поля вполне достаточно для размещения в одной таблице символов большинства языков планеты.