Машинное представление данных

Хранение информации

В ЭВМ обычно выделяют три основных вида запоминающих устройств:

сверхоперативная, оперативная и внешняя память.

Обычно сверхоперативная память строится на регистрах (в частности, регистрах процессора). Регистры используются для временного хранения и преобразования информации.

Оперативная память предназначена для запоминания более постоянной по своей природе информации, но после выключения ЭВМ она исчезает. Важнейшим свойством оперативной памяти является адресуемость. Это означает, что каждая ячейка памяти имеет свой идентификатор, однозначно идентифицирующий ее в общем массиве ячеек памяти. Этот идентификатор называется адресом. Адреса ячеек являются операндами тех машинных команд, которые обращаются к оперативной памяти. В подавляющем большинстве современных вычислительных систем единицей адресации является байт - ячейка, состоящая из 8 двоичных разрядов. Определенная ячейка оперативной памяти или множество ячеек могут быть связаны с конкретной переменной в программе. Для выполнения арифметических вычислений, в которых участвует переменная, необходимо, чтобы до начала вычислений значение переменной было перенесено из ячейки памяти в регистр. Если результат вычисления должен быть присвоен переменной, то результирующая величина снова должна быть перенесена из соответствующего регистра в связанную с этой переменной ячейку оперативной памяти.

Внешняя память служит прежде всего для долговременного хранения данных. Характерным для данных на внешней памяти является то, что они могут сохраняться там даже после завершения создавшей их программы и могут быть впоследствии многократно использованы той же программой при повторных ее запусках или другими программами. Внешняя память используется также для хранения самих программ, когда они не выполняются.

При программировании необходимо понимать машинное представление данных, т. е. знать как данные физически размещается в памяти вычислительной машины (в первую очередь, в оперативной).

Машинное представление данных

Различные структуры данных имеют разное машинное представление, особенно базовые (простые) структуры, из которых строятся более сложные конструкции данных.

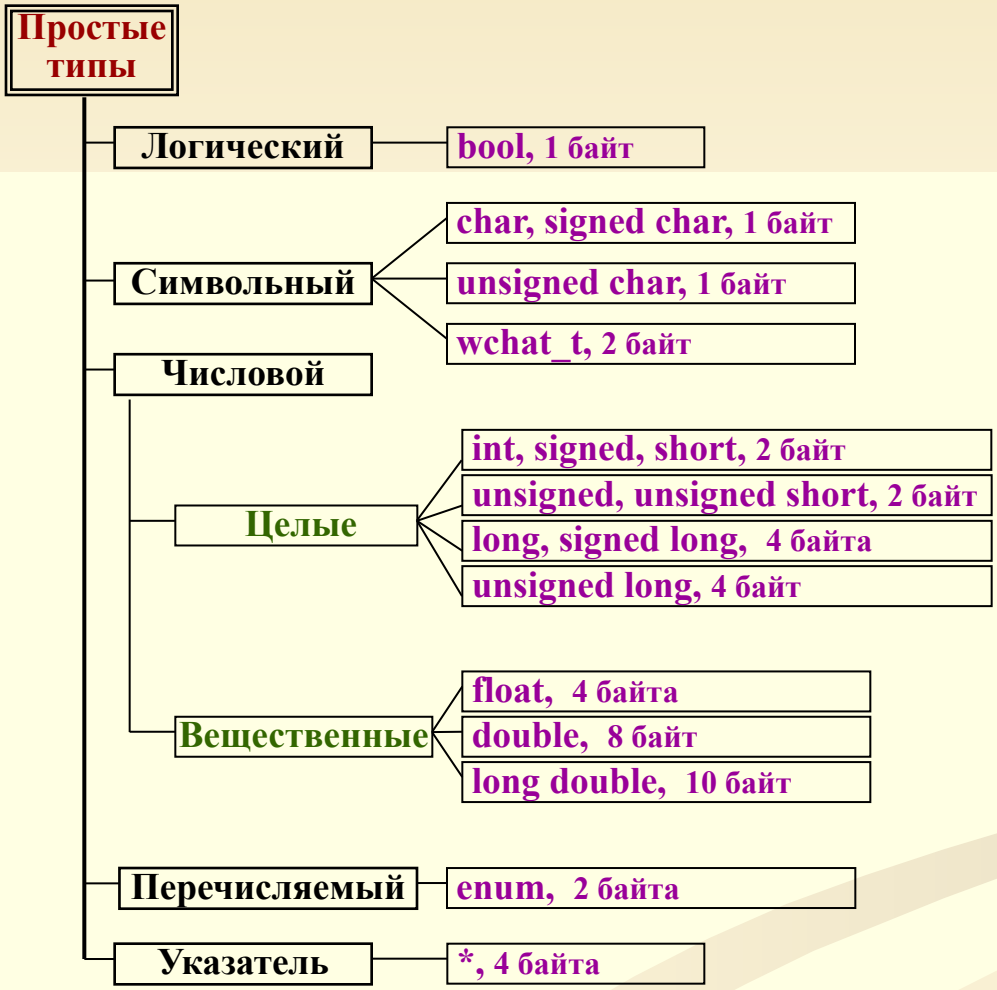
Машинное представление данных

C / C++

Типовые размеры памяти для простых типов данных

Размер памяти, необходимый для данных того или иного базового типа, может быть разным для разных процессоров и разных языков программирования, а также и в разных реализациях одного и того же языка.

Процессор Intel x86 – ЯП MS VS C/C++ **Процессор Intel x64 – ЯП MinGW C++ в Qt 5**

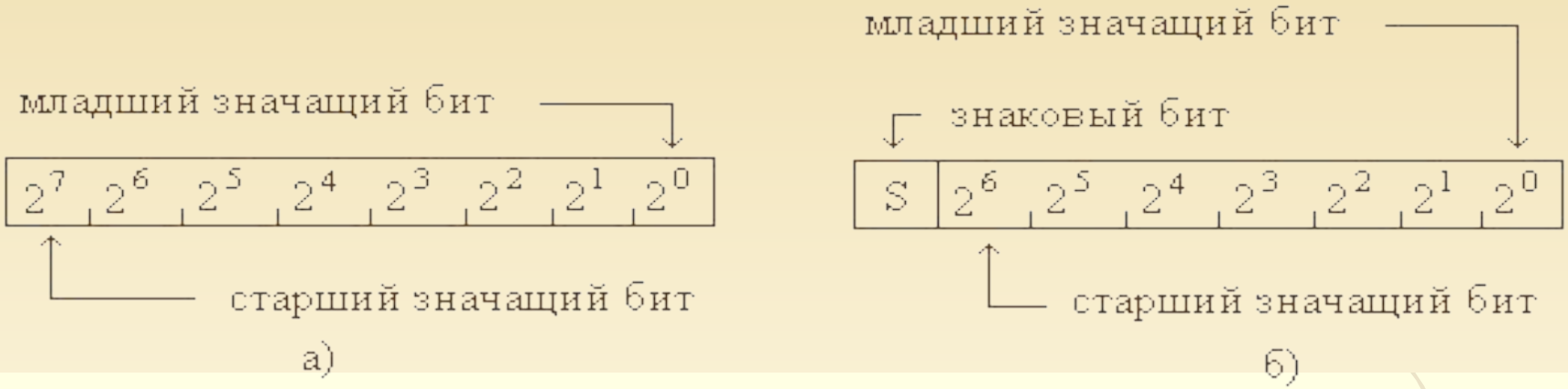


Исследовать размеры оперативной памяти для базовых типов данных C++ в Qt 5

и нарисовать схему

Машинное представление данных

Машинное представление байта



Байт

а - беззнаковый байт, б - знаковый байт

Целые и вещественные числа представляются в памяти компьютера по-разному.

В байте со знаком, для представления целого числа можно использовать не 8, а только 7 бит (разрядов). Восьмой же бит является *знаковым (S – sign)*, т. е. индицирует знак хранимого в байте числа; 1 в этом разряде, обычно, соответствует отрицательному числу, а 0 – положительному.

Операции с таким представлением целых чисел наглядны, но не экономичны. Для оперирования со знаком числа требуется выполнять специальные алгоритмы операции, а это затраты машинного времени.

Машинное представление байта

Более экономным является представление отрицательных целых чисел в дополнительном коде, который формируется следующим образом:

- **модуль отрицательного числа записать в прямом коде, в неиспользуемые старшие биты записать нули;**
- **сформировать обратный код числа, для этого нуль заменить единицей, а единицу заменить нулем;**
- **к обратному коду числа прибавить единицу;**
- **знаковый (старший) разряд числа сделать равным 1.**

Пример: для числа -33 в формате integer:

1000000000100001 - прямой код

0111111111011110 - обратный код

\+ 1

1111111111011111 - дополнительный код

Для положительных чисел прямой, обратный и дополнительный коды одинаковы, при этом знаковый (старший) разряд равен 0. Аналогично представляются целые числа других типов.

Машинное представление данных

Дополнительный код позволяет свести операцию вычитания к сложению положительного числа в прямом коде и отрицательного в обратном.

Пример: **Сложим +1 и -1:**

00000001 - в прямом двоичном коде

11111111 - в дополнительном коде

00000000 - перенос разряда дает в результате +1-1=0

(левый разряд переполнения – отбрасывается)

Тот же принцип дополнительного кода для отрицательных чисел можно использовать и в компьютерном представлении шестнадцатеричных (десятичных) чисел: для каждого разряда цифра X заменяется на 15-X (9-X), и к получившемуся числу добавляется 1.

Примеры:

- **Десятичные числа – при использовании четырёхзначных чисел -0081 заменяется на 9919 (9919+0081=0000, пятый разряд отбрасывается),**
- **16-ричные числа в дополнительном коде – 7F3C - прямой код**

80C3 - обратный код

+1

80C4 – дополнительный код

Машинное представление данных

Для отображения шестнадцатеричных чисел (в частности, в браузерах FAR, Total Commander и других используются восемь двоичных разрядов байта разбитые на две группы по четыре бита.

Четырьмя разрядами двоичной системы счисления представляется одна цифра шестнадцатеричной системы счисления (F_{16} -> 1111_2).

В шестнадцатеричной системе счисления число представляется в виде суммы степеней числа 16. Для изображения числа используется шестнадцать цифр: десять обычных десятичных цифр {0,1,2,3,4,5,6,7,8,9} и шесть латинских заглавных букв {A,B,C,D,E,F}.

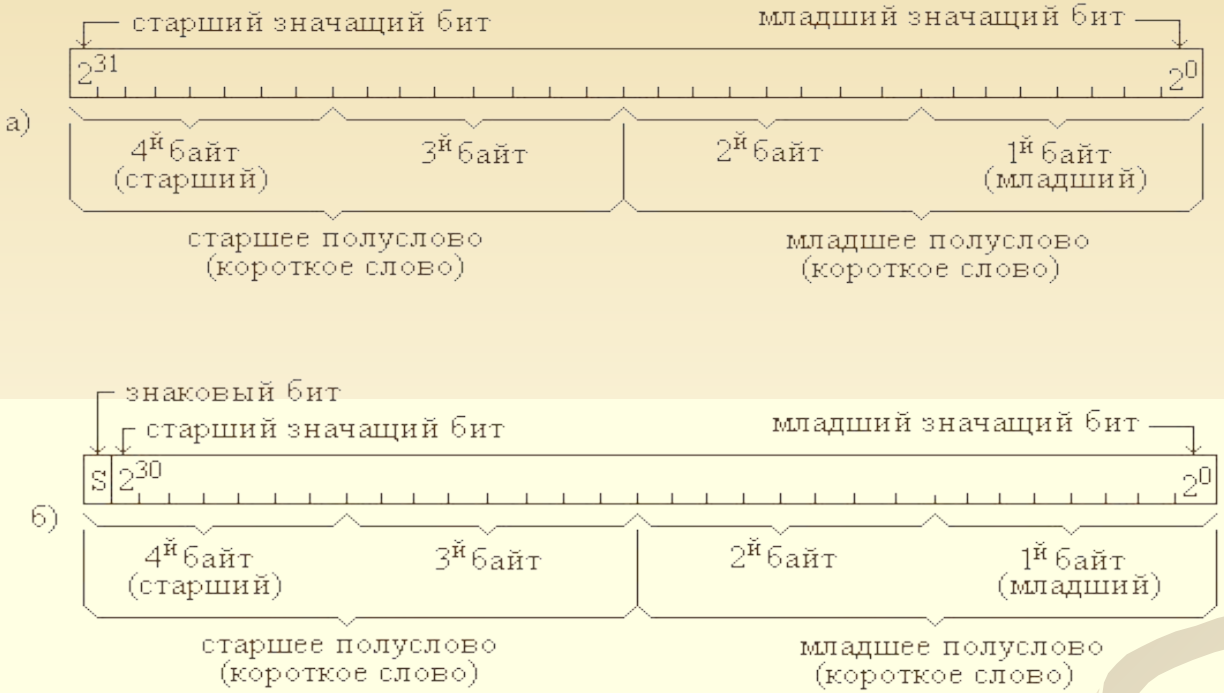
Таким образом содержимое байта отображают двузначным числом в шестнадцатеричной системе счисления (младший разряд – справа).

Например, $0110\ 1101_2 = 6D_{16}$

$1010\ 0010_2 = A2_{16}$

Машинное представление данных

Машинное представление совокупности байт



Двойное слово 32 (разряда)

а - беззнаковое, б - знаковое

В 16-разрядных процессорах слово состоит из 2-х байт. В 32- разрядных процессорах слово состоит из 4-х байт. Двойное слово, как и следует из названия, содержит ровно в два раза больше байт, чем просто слово. Как же называют набор из двух байт для 32-разрядного процессора? На больших ЭВМ - это полуслово. А на процессоре Intel 80x86 (младшие модели этих процессоров были 16-разрядными, а начиная с 80386 стали 32-разрядными) фирма Intel сохранила терминологию 16-разрядных моделей. В официальной документации на процессоры Intel словом, или коротким словом, называется набор из 2 байт, то есть *16 разрядов*. Это верно даже для Pentium. Набор из 4 байт, или *32 разряда*, называется двойным словом, или длинным словом. *Это сделано для единства терминологии, независимо от конкретной модели ПЭВМ.*

В процессорах Intel слова хранятся в памяти начиная с младшего байта, и за адрес слова принимается адрес младшего байта. То есть короткое слово 53C6 в памяти хранится так: C6, 53. А длинное слово 14AFB820 так: 20, B8, AF, 14. Т.е. младший байт числа – слева, а младший разряд в байте – справа.

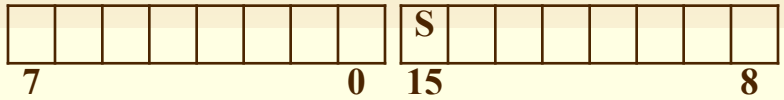
Машинное представление данных

C / C++

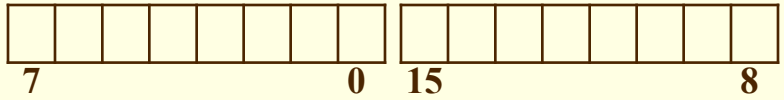
Целые числовые типы данных

Процессор Intel x86 -- ЯП MS VS C/C++ **Процессор Intel x64 -- ЯП MinGW C++ в Qt 5**

int, signed [int], short [int] – переменная хранится как слово (2 байта) со знаком



unsigned [int], unsigned short [int] – переменная хранится как слово (2 байта) без знака



long [int], signed long [int] – переменная хранится как двойное слово (4 байта) со знаком

unsigned long [int] – переменная хранится как двойное слово (4 байта) без знака

Исследовать размеры памяти для целых типов данных C++ в Qt 5

и нарисовать схему

Машинное представление данных

C / C++

Процессор Intel x86 – ЯП MS VS C/C++

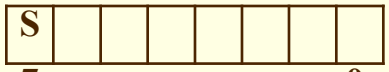
Процессор Intel x64 – ЯП MinGW C++ в Qt 5

Логические типы данных

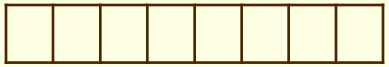
bool – переменная хранится как байт без знака

Символьные типы данных

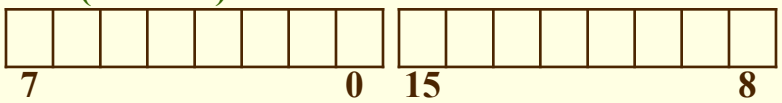
char, signed char – переменная хранится как байт со знаком



unsigned char – переменная хранится как байт без знака

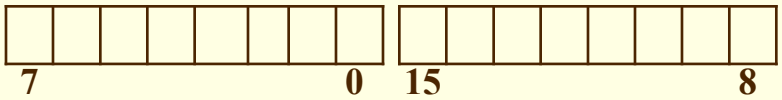


wchar_t – переменная хранится как слово (2 байта) без знака



Перечисляемый тип данных

enum – переменная хранится как тип int (2 байта) без знака



Исследовать размеры оперативной памяти для базовых типов данных C++ в Qt 5

и нарисовать схему

Указатели

Тип указателя – это адрес ячейки памяти. 4 байта.

Кодировка символьных типов данных

(ASCII CP 866 - MS DOS)

Символы с кодами 0 - 127

0 -	16 - ▶	32 -	48 - 0	64 - @	80 - P	96 - '	112 - p
1 - ☺	17 - ◀	33 - !	49 - 1	65 - A	81 - Q	97 - a	113 - q
2 - ☻	18 - ↕	34 - "	50 - 2	66 - B	82 - R	98 - b	114 - r
3 - ♥	19 - !!	35 - #	51- 3	67 - C	83 - S	99 - c	115 - s
4 - ♦	20 - ¶	36 - $	52 - 4	68 - D	84 - T	100 - d	116 - t
5 - ♣	21 - §	37 - %	53 - 5	69 - E	85 - U	101 - e	117 - u
6 - ♠	22 - ▬	38 -&	54 - 6	70 - F	86 - V	102 - f	118 - v
7 -	23 - ↕	39 - '	55 - 7	71 - G	87 - W	103 - g	119 - w
8 -	24 - ↑	40 - (	56 - 8	72 - H	88 - X	104 - h	120 - x
9 -	25 - ↓	41 -)	57 - 9	73 - I	89 - Y	105 - i	121 - y
10 -	26 - →	42 - *	58 - :	74 - J	90 - Z	106 - j	122 - z
11 -	27 - ←	43 - +	59 - ;	75 - K	91 - [	107 - k	123 - {
12 -	28 - ∟	44 - ,	60 - <	76 - L	92 - \	108 - l	124 - |
13 -	29 - ↔	45 - -	61 - =	77 - M	93 - j	109 - m	125 - }
14 - ♫	30 - ▲	46 - .	62 ->	78 - N	94 - ^	110 - n	126 - ~
15 - ☼	31 - ▼	47 - /	63 - ?	79 - O	95 - ÷	111 - o	127 - ⌂
16 - ▶	32 -	48 -0	64 - @	80 - P	96 -	112 - p	

Кодировка символьных типов данных

(ASCII CP 866 - MS DOS)

Символы с кодами 128 - 255

128 -А	144 - Р	160 - а	176 - ▒	192 - └	208 - ╨	224 -р	240 - Ё
129 - Б	145 - С	161 - б	177 - ▒	193 - ┴	209 - ╤	225 -с	241- ё
130 -В	146 - Т	162 - в	178 - ▒	194 - ┬	210 - ╥	226 -т	242 - Є
131 - Г	147 - У	163 - г	179 - │	195 - ├	211 - ╙	227 -у	243 - є
132 -Д	148 - Ф	164 - д	180 - ┤	196 - ─	212 - ╘	228 -ф	244 - Ї
133 -Е	149 - Х	165 - е	181 - ╡	197 - ┼	213 - ╒	229 - х	245 - ї
134 - Ж	150 - Ц	166 - ж	182 - ╢	198 - ╞	214 - ╓	230 - ц	246 - Ў
135 -З	151 - Ч	167 - з	183 - ╖	199 - ╟	215 - ╫	231 - ч	247 - ў
136 -И	152 -Ш	168 - и	184 - ╕	200 -	216 - ╪	232 - ш	248 - °
137 - Й	153 -Щ	169 - й	185 - ╣	201 - ╔	217 - ┘	233 - щ	249 - ●
138- К	154 - Ъ	170 - к	186 - ║	202 - ╩	218 - ┌	234 - ъ	250 - ·
139 -Л	155 - Ы	171 - л	187 - ╗	203 - ╦	219 - ▉	235 - ы	251 - √
140 -М	156 - Ь	172 - м	188 - ╝	204 - ╠	220 - ▄	236 - ь	252- №
141 -Н	157 - Э	173 - н	189 - ╜	205 - ═	221 - ▌	237 - э	253 - ¤
142 -О	158 - Ю	174 - о	190 - ╛	206 - ╬	222 - ▐	238 - ю	254 - ■
143- П	159 - Я	175 - п	191 - ┐	207 - ╧	223 - ▀	239 - я	255 -
144 - Р	160 - а	176 - ▒	192 - └	208 - ╨	224 - р	240 - Ё	

Кодировка символьных типов данных

Сводная таблица кодов ASCII

ASCII таблица кодов символов Windows (Win-1251)

Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ
0	0	спец. NOP	32	20	спец. SP (Пробел)	64	40	@	96	60	`	128	80	Ъ	160	A0		192	C0	А	224	E0	а
1	1	спец. SOH	33	21	!	65	41	A	97	61	a	129	81	Г	161	A1	Ў	193	C1	Б	225	E1	б
2	2	спец. STX	34	22	"	66	42	B	98	62	b	130	82	,	162	A2	ў	194	C2	В	226	E2	в
3	3	спец. ETX	35	23	#	67	43	C	99	63	c	131	83	ѓ	163	A3	Ј	195	C3	Г	227	E3	г
4	4	спец. EOT	36	24	$	68	44	D	100	64	d	132	84	„	164	A4	¤	196	C4	Д	228	E4	д
5	5	спец. ENQ	37	25	%	69	45	E	101	65	e	133	85	…	165	A5	Ґ	197	C5	Е	229	E5	е
6	6	спец. ACK	38	26	&	70	46	F	102	66	f	134	86	†	166	A6	¦	198	C6	Ж	230	E6	ж
7	7	спец. BEL	39	27	'	71	47	G	103	67	g	135	87	‡	167	A7	§	199	C7	З	231	E7	з
8	8	спец. BS	40	28	(	72	48	H	104	68	h	136	88	€	168	A8	Ё	200	C8	И	232	E8	и
9	9	спец. Табуляция	41	29	)	73	49	I	105	69	i	137	89	‰	169	A9	©	201	C9	Й	233	E9	й
10	0A	спец. LF (Возвр. каретки)	42	2A	*	74	4A	J	106	6A	j	138	8A	Љ	170	AA	Є	202	CA	К	234	EA	к
11	0B	спец. VT	43	2B	+	75	4B	K	107	6B	k	139	8B	‹	171	AB	«	203	CB	Л	235	EB	л
12	0C	спец. FF	44	2C	,	76	4C	L	108	6C	l	140	8C	Њ	172	AC	¬	204	CC	М	236	EC	м
13	0D	спец. CR (Новая строка)	45	2D	-	77	4D	M	109	6D	m	141	8D	Ќ	173	AD	-	205	CD	Н	237	ED	н
14	0E	спец. SO	46	2E	.	78	4E	N	110	6E	n	142	8E	Ћ	174	AE	®	206	CE	О	238	EE	о
15	0F	спец. SI	47	2F	/	79	4F	O	111	6F	o	143	8F	Џ	175	AF	Ї	207	CF	П	239	EF	п
16	10	спец. DLE	48	30	0	80	50	P	112	70	p	144	90	ђ	176	B0	°	208	D0	Р	240	F0	р
17	11	спец. DC1	49	31	1	81	51	Q	113	71	q	145	91	'	177	B1	±	209	D1	С	241	F1	с
18	12	спец. DC2	50	32	2	82	52	R	114	72	r	146	92	'	178	B2	І	210	D2	Т	242	F2	т
19	13	спец. DC3	51	33	3	83	53	S	115	73	s	147	93	"	179	B3	і	211	D3	У	243	F3	у
20	14	спец. DC4	52	34	4	84	54	T	116	74	t	148	94	"	180	B4	ґ	212	D4	Ф	244	F4	ф
21	15	спец. NAK	53	35	5	85	55	U	117	75	u	149	95	•	181	B5	µ	213	D5	Х	245	F5	х
22	16	спец. SYN	54	36	6	86	56	V	118	76	v	150	96	–	182	B6	¶	214	D6	Ц	246	F6	ц
23	17	спец. ETB	55	37	7	87	57	W	119	77	w	151	97	—	183	B7	·	215	D7	Ч	247	F7	ч
24	18	спец. CAN	56	38	8	88	58	X	120	78	x	152	98	[illegible]	184	B8	ё	216	D8	Ш	248	F8	ш
25	19	спец. EM	57	39	9	89	59	Y	121	79	y	153	99	™	185	B9	№	217	D9	Щ	249	F9	щ
26	1A	спец. SUB	58	3A	:	90	5A	Z	122	7A	z	154	9A	љ	186	BA	є	218	DA	Ъ	250	FA	ъ
27	1B	спец. ESC	59	3B	;	91	5B	[	123	7B	{	155	9B	›	187	BB	»	219	DB	Ы	251	FB	ы
28	1C	спец. FS	60	3C	<	92	5C	\	124	7C	|	156	9C	њ	188	BC	ј	220	DC	Ь	252	FC	ь
29	1D	спец. GS	61	3D	=	93	5D	]	125	7D	}	157	9D	ќ	189	BD	Ѕ	221	DD	Э	253	FD	э
30	1E	спец. RS	62	3E	>	94	5E	^	126	7E	~	158	9E	ћ	190	BE	ѕ	222	DE	Ю	254	FE	ю
31	1F	спец. US	63	3F	?	95	5F	_	127	7F		159	9F	џ	191	BF	ї	223	DF	Я	255	FF	я

Машинное представление данных

Формат машинного представления вещественных чисел

Система вещественных чисел, применяемая при ручных вычислениях, предполагается бесконечно непрерывной. Это означает, что не существует никаких ограничений на диапазон используемых чисел и точность их представления. Для любого вещественного числа имеется бесконечно много чисел, которые больше или меньше его, а между любыми двумя вещественными числами также находится бесконечно много вещественных чисел.

Реализовать такую систему в технических устройствах невозможно. Во всех компьютерах размеры ячеек памяти фиксированы, что ограничивает систему представимых чисел. Ограничения касаются как диапазона, так и точности представления чисел, т.е. система машинных чисел оказывается конечной и и дискретной, образуя подмножество системы вещественных чисел.

Отсюда, в отличии от порядковых типов (все целые, символьный, логический), значения которых всегда сопоставляются с рядом целых чисел и, следовательно, представляются в памяти машины абсолютно точно, значение вещественных типов определяет число лишь с некоторой конечной точностью, зависящей от внутреннего формата вещественного числа.

Для вещественных чисел по стандарту ANSI/IEEE 754-1985 (IEEE Standard for Binary Floating-Point Arithmetic's) используется нормализованное представление со смещенным порядком. То есть число приводится к виду:

$$A = \pm M \cdot 2^{\pm P}, \quad ½ <= M < 1,$$

где M – мантисса, P – порядок. (Нормализованная запись отличного от нуля действительного числа – это запись вида $a = m*Q^p$, где p – целое число (положительное, отрицательное или ноль), а m – правильная Q-ричная дробь, у которой первая цифра после запятой не равна нулю, то есть $1/Q <= m <1$). Так как в результате старший разряд целой части двоичного числа всегда равен единице, его обычно в памяти не хранят («скрытый бит» или «скрытая единица»). Порядок хранится в смещенном коде («модифицированный порядок» или «характеристика»): $P' = P+2^{n-1}$, где n – число разрядов машинного представления порядка. Это делается для того, чтобы характеристика всегда была положительной. Для представления нуля обнуляются все биты мантиссы и порядка.

Объясним ИНАЧЕ: Старший разряд двоичного представления вещественного числа всегда кодирует знак числа. Остальная часть разбивается на две части: мантиссу и экспоненту. Вещественное число имеет вид:

$$A = \pm S \cdot M \cdot 2^E,$$

где S - знаковый бит числа, E - экспонента, M - мантисса. Если $1/2 \le M < 1$, то такое число называется нормализованным. При хранении нормализованных чисел сопроцессор отбрасывает целую часть мантиссы (она всегда 1), сохраняя лишь дробную часть. Экспонента кодируется со сдвигом на половину разрядной сетки, таким образом, удается избежать вопроса о кодировании знака экспоненты. Т.е. при 8-битной разрядности экспоненты код 0 соответствует числу -127, 1 - числу -126, ..., 255 числу +126 (экспонента вычисляется как код 127).

Машинное представление данных

Формат машинного представления вещественных чисел

Стандарт IEEE-754 определяет три основных способа кодирования (типа) вещественных чисел:

Формат	Общая длина (байт / бит)	Знак мантиссы (бит)	Порядок (бит)	Мантисса (бит)	Смещение порядка	Диапазон представимых чисел	Точность в десятичной системе счисления	Особенность представления
Вещественное ординарной точности single precision	4 / 32	1	8	23	127_{10} $7FH_{16}$	$10^{-38}...10^{38}$	7 – 8 цифр	неявный бит F_0
Вещественное двойной точности double precision	8 / 64	1	10	53	1023_{10} $3FFH_{16}$	$10^{-308}...10^{308}$	15 – 16 цифр	неявный бит F_0
Вещественное расширенной точности extended precision	10 /80	1	15	64	16383_{10} $3FFFH_{16}$	$10^{-4932}...10^{4932}$	19 – 20 цифр	явный бит F_0

Пример: кодирование числа 178,625 в соответствии с IEEE-754.

$$178{,}625_{10} = 128 + 32 + 16 + 2 + 0{,}5 + 0{,}125 =$$
$$1\cdot2^7 + 0\cdot2^6 + 1\cdot2^5 + 1\cdot2^4 + 0\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 + + 1\cdot2^{-1} + 0\cdot2^{-2} + 1\cdot2^{-3} = 10110010{,}101_2$$

Его нужно нормализовать (привести в экспоненциальный вид):

$$1{,}78625E_{10}2 = 1{,}0110010101E_2111$$

В формате вещественного числа одинарной точности оно будет представлено так:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Знак | Смещенная экспонента: 111+01111111 | Дробная часть мантиссы 1,0110010101

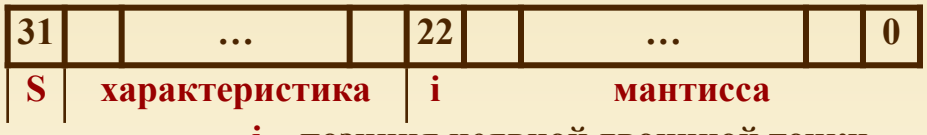
Машинное представление данных

C / C++

Вещественные числовые типы данных

Процессор Intel x86 – ЯП MS VS C/C++ **Процессор Intel x64 – ЯП MinGW C++ в Qt 5**

float



i – позиция неявной двоичной точки

Исследовать размеры памяти для вещественных типов данных C++ в Qt 5

и нарисовать схему

double

i – позиция неявной двоичной точки

long double

1 –целочисленный бит мантиссы

(явный в long и неявный во float и double)

Машинное представление данных

C / C++

Вещественные числовые типы данных

Алгоритм формирования машинного представления вещественного числа в памяти ЭВМ

1) Число представляется в двоичном коде.
2) Двоичное число нормализуется. При этом для чисел, больших единицы, плавающая точка переносится влево, определяя положительный порядок. Для чисел, меньших единицы, точка переносится вправо, определяя отрицательный порядок.
3) Затем с учетом типа вещественного числа определяется характеристика.
4) В отведенное поле памяти (в соответствии с типом числа) записываются мантисса, характеристика и знак числа. При этом :
 (а) для чисел типа float, double, long double характеристика хранится в старших байтах памяти;
 (б) знак числа находится всегда в старшем бите старшего байта;
 (в) мантисса всегда хранится в прямом коде;
 (г) целая часть мантиссы (*у нормализованного числа всегда равна* 1) для чисел типа float, double не хранится (является скрытой).

В числах типа long double все разряды мантиссы хранятся в памяти ЭВМ.

Машинное представление данных

C \ C++

Практическое занятие:

Анализ машинных форматов

Пример программы для анализа машинного представления данных смотреть на сетевом диске commonkof, папка Презентации – ПРГ\C в файле

Data-in-Mem-struct.txt

Задание:

1. **Исследовать размеры памяти для всех типов данных C++ в Qt 5 (целые, символьные, логические, перечисляемые, вещественные, строки) и нарисовать схемы машинного представления данных.**
2. **Вывести в файл datatype.dtc заданные преподавателем данные и просматривая их на экране в 16-ричном виде определить где какое число и символ (смотреть 16-ричное значение в браузере FAR: F3-Просмотр, F4-Код).**
3. **Вывести в программе адреса вводимых переменных и определить сколько памяти отводится под каждое значение.**