лекция 3

ПРОСТЫЕ ТИПЫ ДАННЫХ ЯЗЫКА С

План лекции

- Простые типы данных
  - Ограничения на простые типы данных
- Машинное представление простых типов данных
- Системы счисления (представление целых чисел чисел)

Простые типы данных

- *Тип данных* – это пара, состоящая из множества значений и набора операций над ними
- Языки программирования позволяют строить одни типы данных из других типов данных
- *Простые* типы данных – это типы данных, которые нельзя построить из других типов данных
- *Составные* типы данных – это типы данных, которые строятся из других типов данных

Простые типы данных Си

- Символы, 8-битовые целые
- Целые
- Числа с плавающей точкой
- Перечислимые типы

Простые типы данных -- символы

- C89
  - спецификатор-символьного-типа ::= [signed|unsigned] char
  - Символы и 8-битовые целые со знаком (signed) или без знака (unsigned)
- CHAR_MIN, CHAR_MAX, UCHAR_MAX и др. в limits.h
- Стандарт не определяет, есть ли знак у значений типа char

Простые типы данных -- целые

- C89
  - спецификатор-целого-типа ::=
    [signed|unsigned] [short|long] int
- C99, C11 (поддержка есть в gcc 4.6)
  - спецификатор-целого-типа ::=
    [signed|unsigned] [short|long [long]] int
- C89/C99/C11 не определяют, есть ли знак у int
  - Все известные компиляторы считают int = singed int
- Нестандартные целые типы
  - __int16, __int32, __int64, __int128
  - Наличие и смысл зависят от компилятора

Простые типы данных -- целые

Варианты имени	Диапазон значений в limits.h
[signed] short [int]	SHRT_MIN … SHRT_MAX
unsigned short [int]	0 … USHRT_MAX
int|signed [int]	INT_MIN … INT_MAX
unsigned [int]	0 … UINT_MAX
[signed] long [int]	LONG_MIN … LONG_MAX
unsigned long [int]	0 … ULONG_MAX
[signed] long long [int]	LLONG_MIN … LLONG_MAX
unsigned long long [int]	0 … ULLONG_MAX

sizeof(char) == sizeof(unsigned char) <=

<= sizeof(short) == sizeof(unsigned short) <=

<= sizeof(int) == sizeof(unsigned) <=

<= sizeof(long) == sizeof(unsigned long) <=

<= sizeof(long long) == sizeof(unsigned long long)

Простые типы данных – числа с плавающей точкой

- C89/C99/C11
  - спецификатор-типа-с-плавающей ::=
    float | [long] double
- sizeof(float) <= sizeof(double) <= sizeof(long double)
- FLT_MIN, FLT_MAX, DBL_MIN, DBL_MAX, LDBL_MIN, LDBL_MAX и др. в файле float.h

Простые типы данных – перечислимые типы

- C89/C99/C11
  - enum-спецификатор ::=
    'enum' [имя] '{' список-перечислителей '}'
    | 'enum' [имя] '{' список-перечислителей ',' '}'
    | 'enum' имя
  - список-перечислителей ::= перечислитель
    | список-перечислителей ',' перечислитель
  - перечислитель ::= перечислимая-константа
    | перечислимая-константа '='
    константное-выражение
  - перечислимая-константа ::= имя
  - *константное-выражение на след. лекции*
- Тип, диапазон значений и размер в памяти такие же, как у int

Простые типы данных – перечислимые типы

- Примеры
  - enum my_boolean_t { my_false = 0, my_true = 1 }
  - enum my_boolean_t { my_false, my_true }
    - my_false = 0
    - my_true = my_false+1 = 1
  - enum my_boolean_t { my_false = 0, my_true = 0 }
    - my_false = my_true = 0
  - enum my_day_t { mon, tue, wed, thu, fri, sat, sun }

Машинное представление данных простых типов

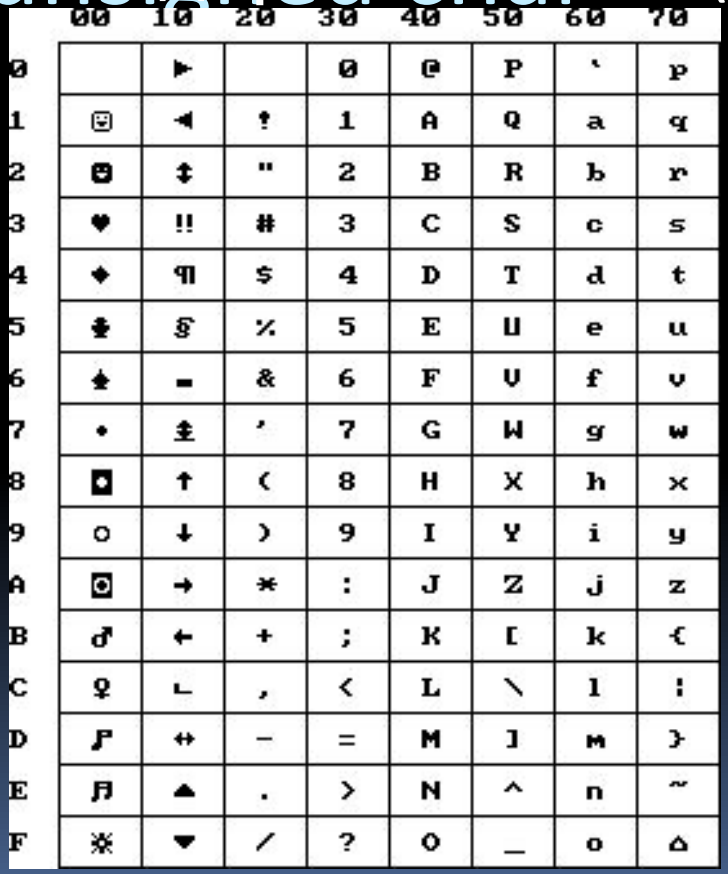
- Символы, 8-битовые целые
- Целые
- Числа с плавающей точкой

Машинное представление значений типа char, signed char, unsigned char

- 1 байт памяти,
  - signed char целые числа от -128 до 127
  - unsigned char целые числа от 0 до 255
- Программы на Си используют значения типов char, signed char, unsigned char для печати текстовых сообщений на экране, бумаге и т.п.
- Соответствие значений и символов определяется кодировкой ОС

Машинное представление значений типа char, signed char, unsigned char

Кодировка CP866 (MS DOS)



	00	10	20	30	40	50	60	70
0		►		0	@	P	`	p
1	☺	◄	!	1	A	Q	a	q
2	☻	↕	"	2	B	R	b	r
3	♥	‼	#	3	C	S	c	s
4	♦	¶	$	4	D	T	d	t
5	♣	§	%	5	E	U	e	u
6	♠	▬	&	6	F	V	f	v
7	•	↨	'	7	G	W	g	w
8	◘	↑	(	8	H	X	h	x
9	○	↓	)	9	I	Y	i	y
A	◙	→	*	:	J	Z	j	z
B	♂	←	+	;	K	[	k	{
C	♀	∟	,	<	L	\	l	|
D	♪	↔	-	=	M	]	m	}
E	♫	▲	.	>	N	^	n	~
F	☼	▼	/	?	O	_	o	⌂

	80	90	A0	B0	C0	D0	E0	F0
0	А	Р	а	░	└	╨	р	≡
1	Б	С	б	▒	┴	╤	с	±
2	В	Т	в	▓	┬	╥	т	≥
3	Г	У	г	│	├	╙	у	≤
4	Д	Ф	д	┤	─	╘	ф	⌠
5	Е	Х	е	╡	┼	╒	х	⌡
6	Ж	Ц	ж	╢	╞	╓	ц	÷
7	З	Ч	з	╖	╟	╫	ч	≈
8	И	Ш	и	╕	╚	╪	ш	°
9	Й	Щ	й	╣	╔	┘	щ	∙
A	К	Ъ	к	║	╩	┌	ъ	·
B	Л	Ы	л	╗	╦	█	ы	√
C	М	Ь	м	╝	╠	▄	ь	ⁿ
D	Н	Э	н	╜	═	▌	э	²
E	О	Ю	о	╛	╬	▐	ю	■
F	П	Я	п	┐	╧	▀	я	

Машинное представление значений типа char, signed char, unsigned char

- Linux (KOI8)

Code	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
80 :																
90 :																
A0 :									Ё							
B0 :									ё							
C0 :	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D0 :	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
E0 :	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F0 :	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

- Win 1251

Code	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
80 :																
90 :																
A0 :									Ё							
B0 :									ё							
C0 :	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D0 :	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
E0 :	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F0 :	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

- Mac OS

Code	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
80 :	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
90 :	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
A0 :																
B0 :																
C0 :																
D0 :														Ё	ё	я
E0 :	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F0 :	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	

Машинное представление беззнаковых (unsigned) целых

- *Двоичная запись* числа Ч -- набор $b_n \ldots b_1 b_0$ такой, что $Ч = b_0 \cdot 2^0 + b_1 \cdot 2^1 + \ldots + b_n \cdot 2^n$

- М.П. unsigned числа x – это *двоичная запись* числа $x \bmod 2^{8 \cdot sizeof(T)}$

Машинное представление целых со знаком (signed)

- М.П. signed числа x
  - двоичная запись x mod $2^{8 \cdot sizeof(T)}$, если x >= 0
  - *дополнительный код* |x| -- двоичная запись $2^{8 \cdot sizeof(T)} - |x| \bmod 2^{8 \cdot sizeof(T)}$, если x < 0

- Свойство дополнительного кода
  - М.П.(x) + М.П.(-x) = М.П.(0)

Машинное представление целых со знаком (signed)

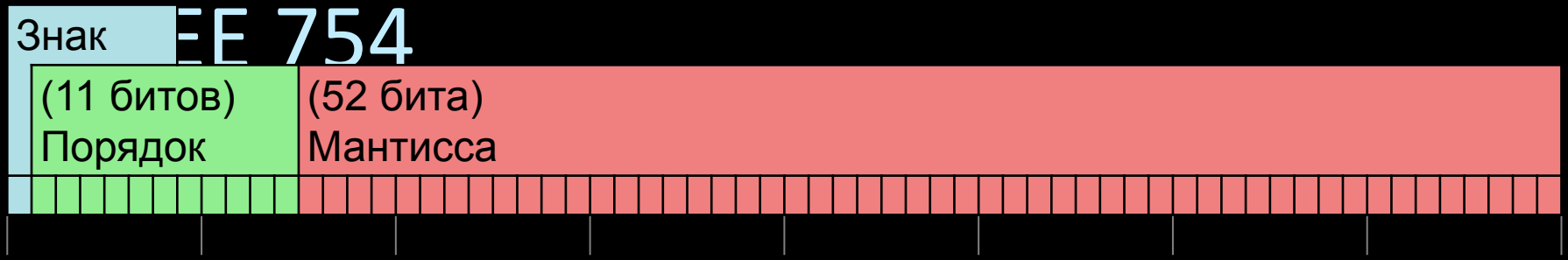
- Построение дополнительного кода |x|
- b[n] – двоичная запись |x|
- d[n] – дополнительный код |x|
- Алгоритм

```
for (i = 0; i < n; i = i+1) d[i] = 1-b[i];
for (i = 0; i < n && d[i] == 1; i = i+1) d[i] = 0;
if (i < n) d[i] = 1;
```

Машинное представление чисел с плавающей точкой

- Числа вида $S \cdot M \cdot 2^P$
- S – знак +1 или -1, 1 бит
- M – *мантисса*, $x/2^{mb}$ от 0 до 1
  - mb – число битов в мантиссе
    - Intel, AMD, ARM -- 23 для float, 52 для double
  - x – целое от 0 до $2^{mb}-1$
- P – порядок
  - Intel, AMD, ARM – 8 битов для float, 11 битов для double
- float занимает 1+8+23 = 32 бита
- double занимает 1+11+52 = 64 бита
- long double обычно совпадает с double или эмулируется

Машинное представление значений типа double – стандарт IEEE 754



Порядок	Мантисса 0	Мантисса != 0	Формула
0x000	0 и -0	Денормализов. числа	$(-1)^{знак} \cdot 2^{порядок-1022} \cdot (0.\text{мантисса})_{(2)}$
0x001 … 0x7fe	Нормализованные числа		$(-1)^{знак} \cdot 2^{порядок-1023} \cdot (1.\text{мантисса})_{(2)}$
0x7ff	+∞ или -∞	NaN	

$3ff0\ 0000\ 0000\ 0000_{(16)} = 1$

$3ff0\ 0000\ 0000\ 0001_{(16)} \approx 1.0000000000000002$

$0000\ 0000\ 0000\ 0000_{(16)} = 0$

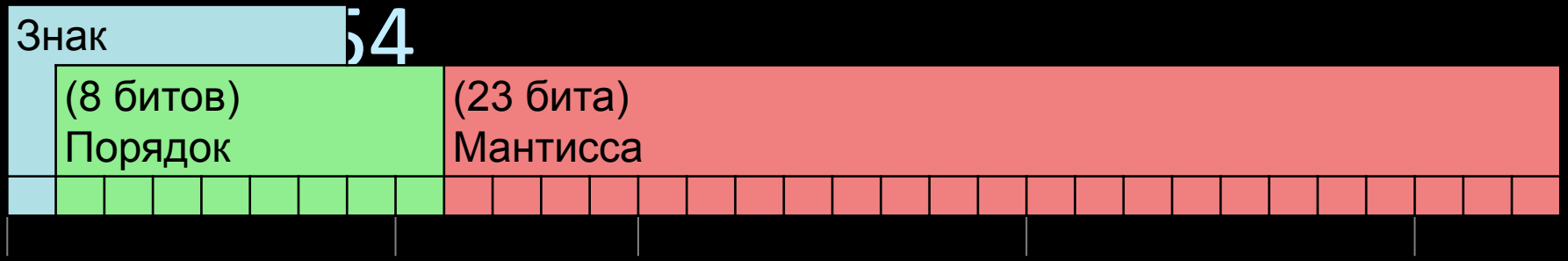
$8000\ 0000\ 0000\ 0000_{(16)} = -0$

$7ff0\ 0000\ 0000\ 0000_{(16)} = \infty$

$fff0\ 0000\ 0000\ 0000_{(16)} = -\infty$

$3fd5\ 5555\ 5555\ 5555_{(16)} \approx 1/3$

Машинное представление значений типа float – стандарт 754



Порядок	Мантисса 0	Мантисса != 0	Формула
0x00	0 и -0	Денормализов. числа	$(-1)^{знак} \cdot 2^{порядок-126} \cdot (0.мантисса)_{(2)}$
0x01 … 0xfe	Нормализованные числа		$(-1)^{знак} \cdot 2^{порядок-127} \cdot (1.мантисса)_{(2)}$
0xff	+∞ или -∞	NaN	

Машинное представление данных простых типов -- разное

- Адрес значения переменной простого типа B *выровнен* (кратен) sizeof(B)

Системы счисления

Значение и обозначение числа

- 9, IX, девять, nine, $1001_{(2)}$
- *Значение* числа
  - Числовая величина, «чистая»
  - Отвлеченная от измеряемых объектов и единиц измерения
- *Обозначение* (форма, внешнее представление) числа
  - Название или знак в некотором языке или системе обозначений, позволяющих отличать данное число от других
- Значение числа не зависит от обозначения

Система счисления (с.с.)

- Система правил для построения названий чисел некоторым регулярным способом
- *Непозиционные* системы счисления возникли первыми
  - Основаны на простом суммировании «весов» – цифр - «разновесов», занятых в записи числа.
  - Римская с.с.
    - Цифры берутся с плюсом или минусом в зависимости от веса соседа слева – IX = 9, XI = 11
- *Позиционные* системы счисления
  - Вклад каждой цифры зависит от «веса» ее позиции в записи числа

Представление целых чисел в позиционных системах счисления с произвольным основанием

- Позиционная система счисления, использующая b цифр, называется b-ичной системой счисления (с. с.)
- Если $b \leq 10$, то первые b цифр из 0, 1, …, 9
- Если $10 \leq b \leq 36$, то первые b знаков из 0, 1, …, 9, A, B, …, Z
  - 10 – A, 11 – B, и т.д.

Запись целого числа

S = $a_{k-1}a_{k-2}a_{k-3}\ldots a_2a_1a_{0(b)}$

$0 \le a_i < b$,

i – индекс позиции (разряда), в которой расположена цифра a_i

Запись числа называется k-значной, если индекс разряда первой значащей цифры числа равен $k - 1$

Примеры

$10011001_{(2)}$, 248933, $7DAB_{(16)}$

$12345_{4(5)}$ - неправильная запись

Соотношение записи целого числа со значением

$S = a_{k-1}a_{k-2}a_{k-3}\ldots a_2 a_1 a_{0(b)}$

S – запись числа

$N(S)$ – значение числа

$$N(s) = a_{k-1}b^{k-1} + a_{k-2}b^{k-2} + \ldots a_1 b^1 + a_0 b^0 = \sum_{i=0}^{k-1} a_i b^i$$

b^i – вес разряда, единица i-го разряда b-ичного числа

a_i – количество полных единиц i-го разряда, которое останется после вычета всевозможного числа единиц старших разрядов

Соотношение записи целого числа со значением – схема Горнера

$$N(s) = a_{k-1}b^{k-1} + a_{k-2}b^{k-2} + \ldots a_1 b^1 + a_0 b^0 = \sum_{i=0}^{k-1} a_i b^i$$

$$N(s) = (((\ldots((a_{k-1}b + a_{k-2})b + a_{k-3})b \ldots + a_1)b + a_0$$

$$N_k = 0;$$

$$N_{i-1} = N_i b + a_{i-1}, \quad i = k \ldots 1;$$

$$N(s) = N_0$$

Примеры

$N(10011_{(2)}) = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19$

$N(10011_{(2)}) = (((1 \cdot 2 + 0) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1 = 19$

$N(30A_{(16)}) = 3 \cdot 16^2 + 0 \cdot 16^1 + 10 \cdot 16^0 = 778$

$N(30A_{(16)}) = (3 \cdot 16 + 0) \cdot 16 + 10 = 778$

Теорема 1

- Любое число однозначно представимо в виде цифр заданной b-с. с.
- Доказательство -- упражнение

Алгоритм перевода b-ичной записи значение

Вход: $b > 0$, $k > 0$ (число цифр), набор $a_{k-1}, a_{k-2}, \ldots, a_1, a_0$.

$$N = a_0; S = b$$

цикл по $i = 1$ до $k - 1$

$$N = N + a_i \cdot S;$$
$$S = S \cdot b;$$

конец цикла;

Выход: N

(2k - 2) операций *

(k-1) операций +

Схема Горнера

Вход: $b > 0$, $k > 0$ (число цифр), набор a_i;

$N = a_{k-1}$;

цикл по i от k - 2 вниз до 0

$N = N \cdot b + a_i$;

конец цикла;

Выход: N

k-1 операция *

k-1 операция +

Построение записи в b-ичной с. с.

Вход: $N \geq 0,\ b > 0;$

$i = 0;$

цикл

$a_i = N \bmod b;$ (*mod* – остаток от деления нацело)

$N = N \operatorname{div} b;$ (div – целое деление)

$i = i + 1;$

пока $N \neq 0;$

Выход:

набор a_i, i (число значащих цифр)

число операций деления = i

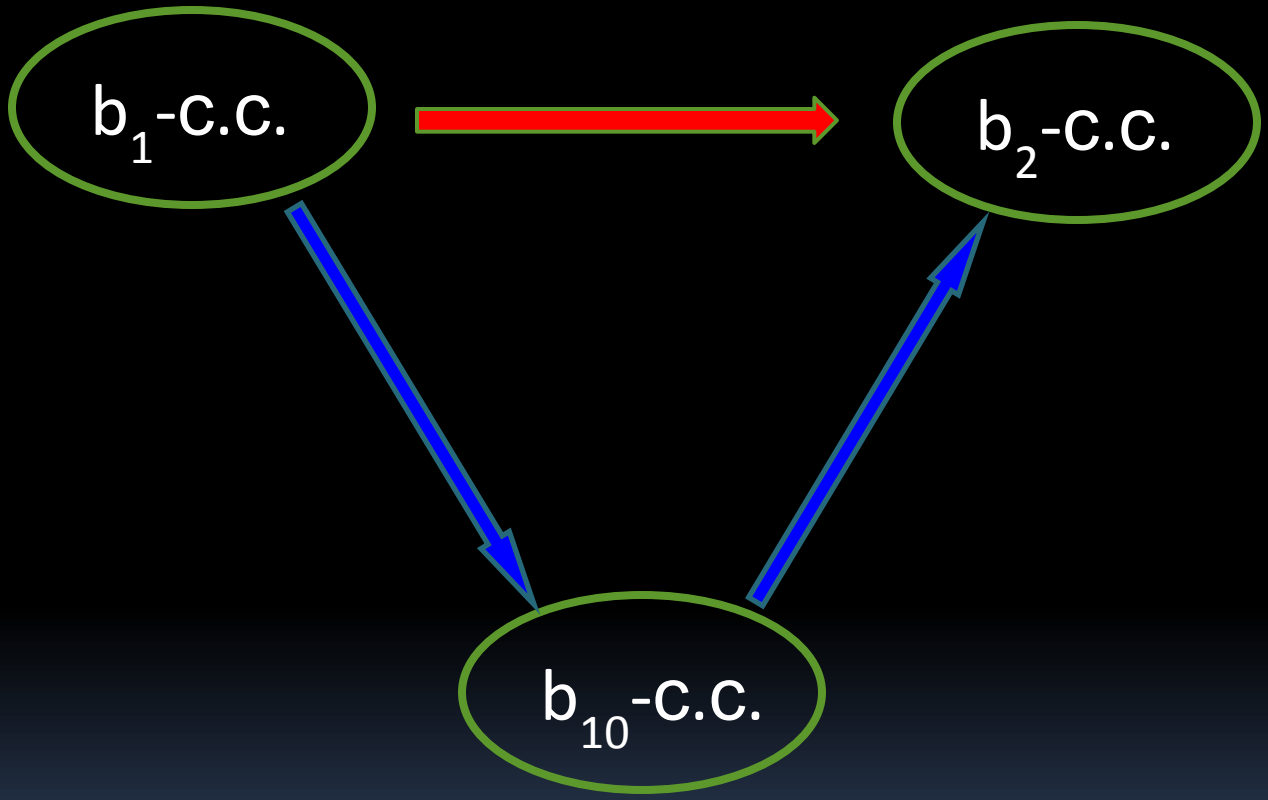
Пример – построение 2-ной записи

325 | 2

Целая часть	Остаток от деления на 2
162	1
81	0
40	1
20	0
10	0
5	0
2	1
1	0
0	1

$325_{(10)} = 101000101_{(2)}$

Перевод числа из b_1-с.с. в b_2-с.с.



Представление действительных чисел

$S = a_{l-1}a_{l-2}a_{l-3}\ldots a_2 a_1 a_0 . a_{-1}a_{-2}a_{-3}\ldots a_{-1}\cdots_{(b)}$

$$N(s) = a_{l-1}b^{l-1} + \ldots + a_0 b^0 + a_{-1}b^{1-} + \ldots + a_{-i}b^{-i} + \ldots = \sum_{i=-\infty}^{l-1} a_i b^i .$$

Если в дробной части числа конечное число знаков *k*, то нижний индекс суммы равен — *k* .

$$N_f(s) = (a_{-1} + (a_{-2} + (a_{-3} + (\ldots)/b)/b)/b)/b$$

0.375=(3+(7+5/10)/10)/10=(3+(7+(5+0)/10)/10)/10

Связь дробной части числа со значением

$$N_f(s) = (a_{-1} + (a_{-2} + (a_{-3} + (...)/b)/b)/b)/b,$$

$$N_{-k-1} = 0;$$

$$N_{-i} = (a_{-i} + N_{-i+1})/b;$$ где $i = k, \ldots, 1$;

$$N_f(s) = N_{-1}.$$

Примеры

$N(\text{«}1.101_{(2)}\text{»}) = 1 \cdot 2^{0} + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$

$= 1 + 0.5 + 0.125$

$= 1.625$

$N_f(\text{«}0.101_{(2)}\text{»}) = (1 + (0 + (1 + 0)/2)/2)/2$

$= (1 + (0 + 0.5)/2\ )/2$

$= (1 + 0.25) / 2 = 0.625$

$N_f(\text{«}0.01_{(3)}\text{»}) = 1 \cdot 3^{-2} = \ = 0.(1)$

Окончание лекции

Алгоритм (построения) записи дробной части в b-с.с

Вход -- *Nf* (0 ≤ *Nf* < 1), *b* >1

i = -1;

цикл

a[i] = ц.ч.(Nf*b); // первая цифра дробной
// части числа Nf

Nf = Nf*b-a[i];

i = i – 1;

пока Nf != 0

Выход -- набор i < 0, a[i+], a[i+2], …, a[-1]

Когда алгоритм не завершится?

Пример построения 2-ичной записи дробного числа

$$0.375 = (3 + (7 + 5/10)/10)/10 = (3 + (7 + (5 + 0)/10)/10)/10$$

$$N_f = 0.375;\ N_{-1} = 0.375;$$

$$0.375 * 2 = 0.75;\ N_{-2} = 0.75;\ a_{-1} = 0;$$

$$0.75 * 2 = 1.5;\ N_{-3} = 0.5;\ a_{-2} = 1;$$

$$0.5 * 2 = 1.0;\ N_{-4} = 0;\ a_{-3} = 1;$$

$$0.375_{(10)} = 0.011_{(2)}$$

$$1/4 + 1/8 = 3/8 = 0.375$$

Конечная представимость рациональных чисел

Несократимая дробь p/q *конечно представима* в b-ной с. с. тогда и только тогда, когда все простые делители q делят b

- 121/675 *конечна* в 15-с.с., т.к. $675 = 3^3*5^2$ и $15 = 3*5$
  - 1/675 = 5*15-3 = 0.005(15);
  - 121*5*15-3 = (2*152 + 10*151 + 5)*15-3 = 2*15-1 + 0*15-2 + 5*15-3
  - 121/675 = 0.2A5(15);
- Запись 1/10 *бесконечна* в 2-с.с.

Вычисление значения по b-ичной записи

Вход: b > 1, к > 0 (число дробных цифр),
набор a[-k], a[-k+1],…, a[-1]

Nf = 0; S = 1/b; // S накапливает степень

цикл по i от -1 вниз до –k

Nf = Nf+a[i]*S;

S = S/b;

конец цикла

Выход: Nf

2k операций *, /

k операций +

Вычисление значения по b-ичной записи по схеме Горнера

Вход: b > 1, k > 0 (число цифр), набор a[-k], a[-k+1], … a[-1]

```
Nf = 0;
цикл по  i от -k  до -1
    Nf = (a[i]+Nf)/b;
конец цикла
```

Выход: Nf

k операций + и /

Кратные системы счисления

- Если основания двух систем счисления b1 и b2 связаны соотношением $b2 = b1^m$ для некоторого натурального m, то такие системы счисления называются *кратными*
- Упражнение – сформулируйте алгоритм перевода для кратных с.с.

Объявление и инициализация переменных простых типов

Для РБНФ <X> обозначим <X>* РБНФ <список X>, заданную правилом

<список X> ::= <X> | <список X> <X>

Объявление и инициализация переменных простых типов

<единица-трансляции> ::=
    <внешнее-объявление>*

<внешнее-объявление> ::=
    <определение-функции> | <объявление>

<определение-функции> ::=
    [<спецификаторы-объявления>] <объявитель>
[<список-объявлений>] <составная-инструкция>

<объявление> ::=
    <простое-объявление> | <составное-объявление>

Объявление и инициализация переменных простых типов

<инструкция>::=
| <помеченная-инструкция>
| <инструкция-выражение>
| <составная-инструкция>
| <инструкция-выбора>
| <циклическая-инструкция>
| <инструкция-перехода>

<инструкция-выражение>::= [<выражение>] ';'

<составная-инструкция>::=
'{' [<объявление>*] [<инструкция>*] '}'

Объявление и инициализация переменных простых типов

<простое-объявление> ::=
<спецификаторы-объявления>
[<простой-объявитель-инициализатор>*]

C89:

Объявления переменных встречаются либо вне самого внешнего блока { }, либо сразу же после {

C99:

Объявления переменных встречаются в любом месте кода

Объявление и инициализация переменных простых типов

<простой-объявитель-инициализатор> ::=
    <простой-объявитель>
| <простой-объявитель> '=' <инициализатор>

<простой-объявитель> ::= <идентификатор>

<инициализатор> ::= <выражение-присваивания>

Объявление и инициализация переменных простых типов

<спецификаторы-объявления> ::=

(

<спецификатор-класса-памяти>

| <спецификатор-простого-типа>

| <квалификатор-типа>

)*

Объявление и инициализация переменных простых типов

<спецификатор-класса-памяти> ::=
| 'auto'
| 'register'
| 'static'
| 'extern'
| 'typedef'

- auto
  - На стеке (по умолчанию)
- register
  - В регистре
- static
  - В статической памяти единицы компиляции
- extern
  - В статической памяти программы
- typedef
  - Вне памяти, объявляемый идентификатор далее обозначает тип

Объявление и инициализация переменных простых типов

<спецификатор-простого-типа> ::=
    'void' | 'char' | 'short' | 'int' | 'long' | 'float'
| 'double' | 'signed' | 'unsigned'
| <спецификатор-enum> -- было
| <typedef-имя>

<typedef-имя> ::= <идентификатор>

<квалификатор-типа> ::= 'const' | 'volatile'

- const
  - Неизменяемое значение
- volatile
  - Значение может асинхронно изменяться – например, в многопоточной программе

Примеры объявлений переменных простых типов

- int x;
- auto int x; // то же, что выше
- const int x; // как задать начальное
  // значение?!
- const double x = 1.234567;
- float x = 0, y = x+1;
- static int x = 5;
- extern unsigned long long global_uuid;

Примеры объявлений переменных простых типов

- typedef int my_int; // my_int – синоним int
  my_int x = 0, y = x+1;

Заключение

- Простые типы данных
- Ограничения на простые типы данных
- Машинное представление простых типов данных
- Системы счисления
  - Представление целых и вещественных чисел
- Объявление и инициализация переменных простых типов

Число N в b-с.с. имеющее k дробных цифр, при умножении на b становится целым (это умножение соответствует сдвигу точки на k позиций влево)

Алгоритм А7

- найти целое $N_1 = N * b_1^k$ (умножением или сдвигом точки);
- выполнить для N_1 один из алгоритмов А1, А2, А3;
- разделить полученный результат на b_1^k в системе b_2

Пример

Перевести $101.101_{(2)}$ **в 10-с.с.**

1) умножим на 2^3 $\rightarrow$ $101101_{(2)}$
2) переведем в **10-с.с.** $\rightarrow$ **45**
3) разделим: **45/8 = 5.625**$_{(10)}$

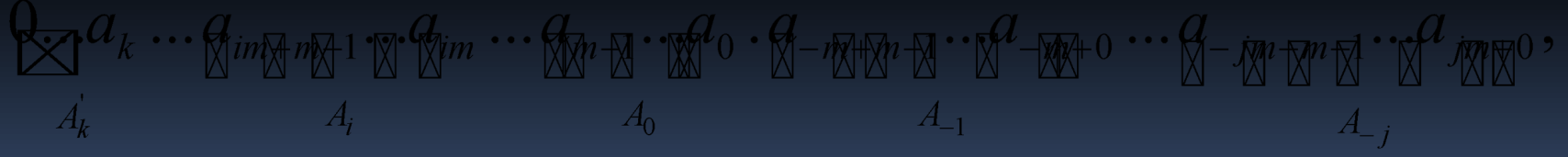
$$101.101 = 1*2^2+1*2^0+1*2^{-1}+1*2^3$$
$$= 5+1/2+1/8=5.625$$

Кратные системы счисления

Если основания двух систем счисления b_1 и b_2 связаны соотношением $b_2 = b_1^m$ для некоторого натурального m, то такие системы счисления называются *кратными*.

Перевод числа из одной с. с. в другую для таких систем можно выполнить проще.

Сгруппируем цифры в b_1-записи числа по m от точки влево и вправо (добавив при нехватке цифр нужное количество незначащих нулей):



затем также сгруппируем слагаемые в формуле (5) (они содержат множитель b_1 в степени, равной индексу цифры), вынесем за скобки из каждой группы i общий множитель

$$(b_1^{im} = (b_1^{m})^i = b_2^{i})$$

и обозначим для каждой группы

Тогда значение исходного числа может быть представлено в виде:

$$N(S') = A_{k'} * b_2^{k'} + \ldots + A_i * b_2^{i} + \ldots + A_0 * b_2^{0} + A_{-1} * b_2^{-1} + \ldots A_{-j}\ b_2^{-j},$$

что по определению совпадает со значением записи того же числа в b_2-с.с. с цифрами A_i, если заметить, что A_i, действительно могут принимать все значения от 0 до $b_1^{m} - 1 = b_2 - 1$.

Таблицы соответствия последовательностей цифр кратных с.с.

9-с.с.	3-с.с.
0	00
1	01
2	02
3	10
4	11
5	12
6	20
7	21
8	22

8-с.с.	2-с.с.
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

16-с.с.	2-с.с.
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Алгоритм А8: перевод из меньшей кратной с.с. в большую

Вход: $b_1 > 1$, $b_2 = b_1^m$, b_1 - представление числа;

- разбить число на группы по m цифр, начиная от точки, в обе стороны (если в крайних группах цифр меньше m, добавить незначащие нули: в целой части спереди, в дробной сзади);
- заменить каждую группу b_2-цифрой по формуле (8) или таблице.

Выход: b_2 -представление исходного числа.

Алгоритм А9: перевод из большей кратной с.с. в меньшую

Вход: $b_1 > 1$, $b_2 = b_1^m$; b_2-представление числа;

- заменить каждую b_2-цифру цепочкой из m b_1-цифр по формуле (8) или таблице;
- отбросить незначащие нули слева и справа.

Выход: b_1-представление исходного числа.

Универсальные алгоритмы для арифметических операций

- Все так называемые *численные* алгоритмы для арифметических операций сложения, вычитания, умножения и деления (в том числе, вычисления «столбиком») являются *символьными*, потому что оперируют входными, выходными и промежуточными данными как строками символов.
- Символьные вычисления являются *формальными в* том смысле, что манипулируют только знаками, не обращаясь к их значениям.
- *Абстрагирование* от смысла данных различной природы и описание алгоритма в терминах чисто символьных преобразований является одним из основных методов программирования обработки данных произвольной природы.

Алгоритм А10: сложение двух чисел

Вход: две строки цифр, представляющие слагаемые;

- *выравнивание*: расположить слагаемые одно под другим в произвольном порядке так, чтобы разряды с одинаковым весом находились друг под другом; если какое-то число короче других слева или справа, дополнить его нулями;
- *начальные установки*:

  обнулить цифру переноса в следующий разряд;

  установить результат равным пустой строке;
- *цикл* по текущему разряду от младшего до старшего:

  определить сумму переноса и цифр в столбце текущего разряда чисел; младшую цифру суммы записать в текущий разряд результата, старшую — в перенос;

  конец цикла;
- *окончание*: если перенос не равен 0, то дописать перенос в начало результата

Выход: строка, представляющая результат.

Единственное место в алгоритме, где присутствует обращение к значениям цифровых символов, — это поразрядное сложение в цикле.

Эти сведения можно задать, например, двумя таблицами сложения: в одной для каждой пары цифр записать младшую цифру результата, в другой — цифру переноса («0» или «1»); исчерпав таким образом все немногочисленные случаи, можно заменить операцию сложения значений операцией выборки знака из таблицы.

Чтобы учесть сложение с переносом, можно завести две пары таблиц или записать в каждую клетку по две цифры.

Алгоритм А10 применим к произвольной позиционной с. с. при соответствующей замене таблиц сложения.

+	0	1
0	0	1
1	1	0

++	0	1
0	0	0
1	0	1

*	0	1
0	0	0
1	0	1

Нетрудно обобщить алгоритм *А*10 для одновременного сложения нескольких чисел, а также аналогичными рассуждениями показать, что алгоритмы вычисления «столбиком» для вычитания, умножения и деления универсально применимы к произвольной с. с. при замене соответствующих таблиц.

Особенности умножения и деления на основание системы счисления

В *b*-с. с. число *b* всегда имеет представление «$10_{(b)}$».

Умножение на *b* сводится к дописыванию справа к целому числу или (что то же), сдвигом *b*-ичной точки на один разряд влево.

Деление на *b* равносильно сдвигу точки на один разряд вправо

или отбрасыванию младшей цифры целого числа — при делении нацело.

Аналогично число *b* всегда представляется единицей с *k* нулями, а умножение (деление) на *b* сводится к сдвигу точки на *k* позиций вправо (влево).

Остатком от деления целого числа нацело на *b* является

число, составленное из *k* младших цифр. Добавление *k* нулей

справа и отбрасывание *k* младших цифр можно рассматривать

как две новые операции *арифметического сдвига* на *k* позиций.

Арифметические сдвиги

Добавление *k* нулей справа и отбрасывание *k* младших цифр можно рассматривать как операции арифметического сдвига на *k* позиций.

В Си определены операции арифметического сдвига на *k* позиций, которые равносильны умножению или целочисленному делению на 2^k.

<< — сдвиг влево

>> — сдвиг вправо

Примеры:

```
a = 5 << 3;  /*  после выполнения присваивания a
   будет иметь            значение 40 */
b = 112 >> 4; /* b будет равно 7 */
```

Особенности двоичной арифметики

+	0	1
0	0	1
1	1	0

++	0	1
0	0	0
1	0	1

*	0	1
0	0	0
1	0	1

Если сопоставить нулю логическую «ложь», а единице — «истину», то таблица сложения совпадет с таблицей значений для логической операции «исключающее или», а таблицы умножения и переноса при сложении — с операцией «и».

На этом совпадении основана схемная реализация в компьютерах поразрядной двоичной арифметики с помощью примитивных логических элементов (вентилей).

Другая аналогия — «минимаксная»: нетрудно видеть, что $ab = min(a,b)$, $a+b = min(a,b)+ max(a,b)$.

Умножение «столбиком» многозначных чисел в двоичной с. с. реализуется только с помощью операций сложения и сдвига.

Сложность арифметических алгоритмов

Затраты памяти на хранение чисел и времени на выполнение операций с ними зависят от длины записи числа в цифрах рабочей системы счисления.

Для заданной b-с. с. следующие величины:

k_n — длина записи (натурального) числа N,

N_k — максимальное натуральное число, записываемое k цифрами, связаны соотношениями:

$k_n = [log_b N] + 1$, где $[x]$ — наибольшее целое, не превышающее x;

$N_k = b^k - 1$.

Верхние оценки для размера результата арифметической операции над парой целых чисел $N1$ и $N2$ (пусть $N1 > N2$):

для сложения и вычитания — $k_{N1} + 1$,

для умножения — $k_{N1} + k_{N2}$,

для деления — $k_{N1} + 1$, (так как $N2 > 1$).

Время исполнения

Алгоритмы сложения содержат один проход по всем разрядам числа, причем каждый разряд обрабатывается не более одного раза. Поэтому время работы алгоритма сложения линейно по k: $T_{слож}(k) \sim k$.

Алгоритмы умножения и деления выполняют сложение и вычитание несколько раз (не более, чем k), со сдвигом на одну позицию. Так как время сложения линейно, время умножения и деления квадратично по k: $T_{умн} \sim k2$,, $T_{дел}(k) \sim k2$.

В системах команд компьютеров есть команды типа сложения
и умножения, которые работают не с отдельными битами, а с
байтами; они обычно рассматриваются как элементарные.
Проведенные выше оценки сохраняют свою силу, если заменить базовую с. с. кратной ей (со степенью кратности равной длине слова).

Упражнения

1. Выразить целую часть 17.5 * X через сложение и операции поразрядных сдвигов числа X вправо и влево.

$17.5_{(10)} = 16 + 1 + 0.5 = 2^4 + 2^0 + 2^{-1} = 10001.1_{(2)}$

$17.5 * X = X * (2^4 + 2^0 + 2^{-1}) =$

$= X*2^4 + X*2^0 + X*2^{-1} =$

$= (X << 4) + X + (X >> 1)$

2. Если $120_{(x)}$ делится на $11_{(10)}$, то как выглядит (чему равно?) 3^{10} в системе счисления с основанием *x*?

Подбором можно определить, что x = 9, т.к. $120_{(9)} = 99_{(10)}$ – делится на 11 без остатка.

$3^{10} = 3^{2*5} = (3^2)^5 = 9^5 = 100000_{(9)}$

Объявление и инициализация переменных простых типов

<объявитель> ::= [<указатель>] <собственно-объявитель>

<указатель> ::= ('*' [<квалификаторов-типа>*])*

<собственно-объявитель> ::=
    <идентификатор>
| '(' <объявитель> ')'
| <собственно-объявитель> '[' [<константное-выражение>] ']'
| <собственно-объявитель> '(' <список-типов-параметров> ')'
| <собственно-объявитель> '(' [<список-идентификаторов>] ')'

<список-типов-параметров>::= <список-параметров> | <список-параметров> ',' '...'

список-параметров::= <объявление-параметра>*

объявление-параметра::=

- спецификаторы-объявления объявитель
- спецификаторы-объявления абстрактный-объявительнеоб

- инициализатор:
  - выражение-присваивания
  - { список-инициализаторов }
  - { список-инициализаторов , }