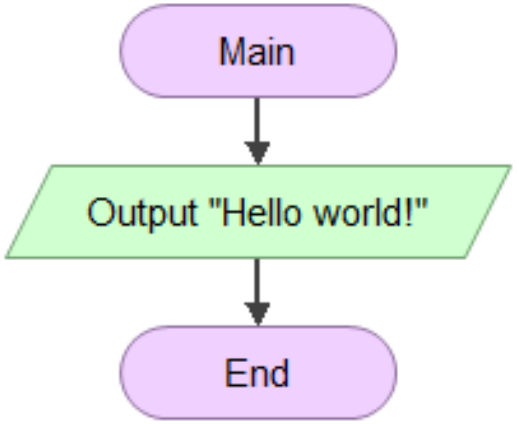
Основы программирования

Первая программа



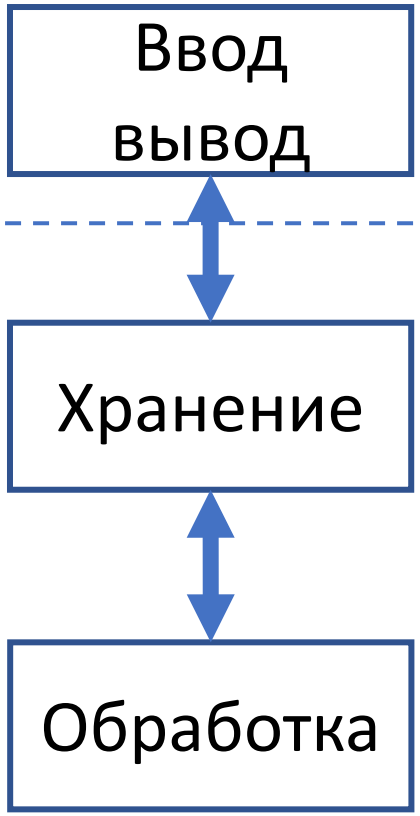
Первая программа

```
#include <stdio.h>                    // подключаем заголовочный файл
int main()                            // определяем функцию main
{                                     // начало функции
     printf("Hello World! \n");       // выводим строку на консоль
     return 0;                        // выходим из функции
}                                     // конец функции
```

Дерево языка

Внешняя среда

Программа



Хранение данных

Тип данных	Назначение
int	Целое число
char	Символ
float	Вещественное число одинарной точности
double	Вещественное число двойной точности

Размер?

MS-DOS (DOSBox)

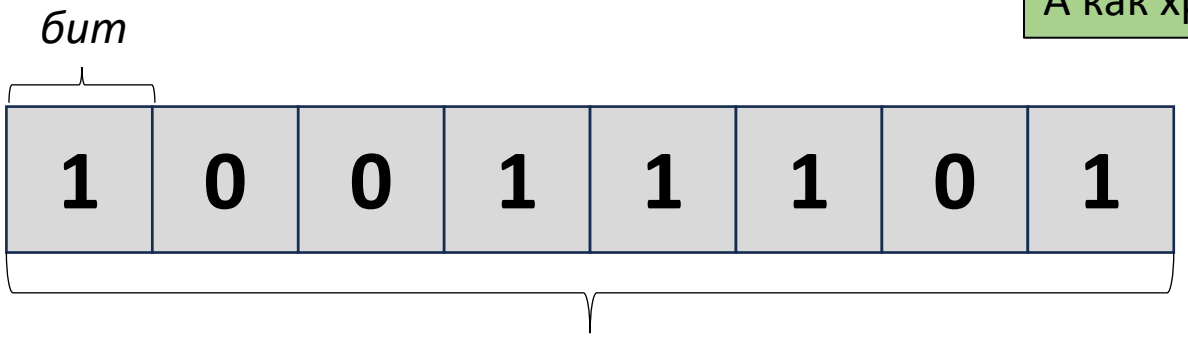
```
C:\PROGRAM>c:\Borlandc\bin\bc.exe
Sizeof  int = 2 bytes
Sizeof char = 1 bytes
Sizeof float = 4 bytes
Sizeof double = 8 bytes
```

Win10

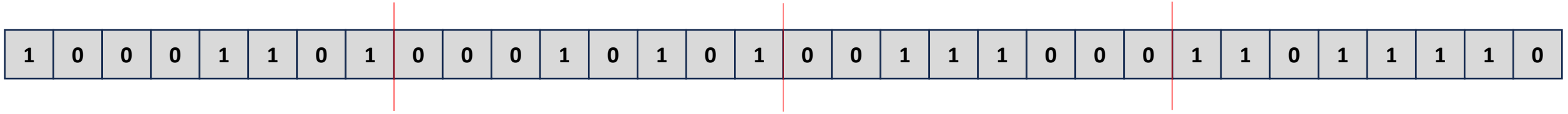
```
C:\Users\79629\Programs>varsize.exe
Sizeof int = 4 bytes
Sizeof char = 1 bytes
Sizeof float = 4 bytes
Sizeof double = 8 bytes
```

Целые числа

А как хранить отрицательные числа?



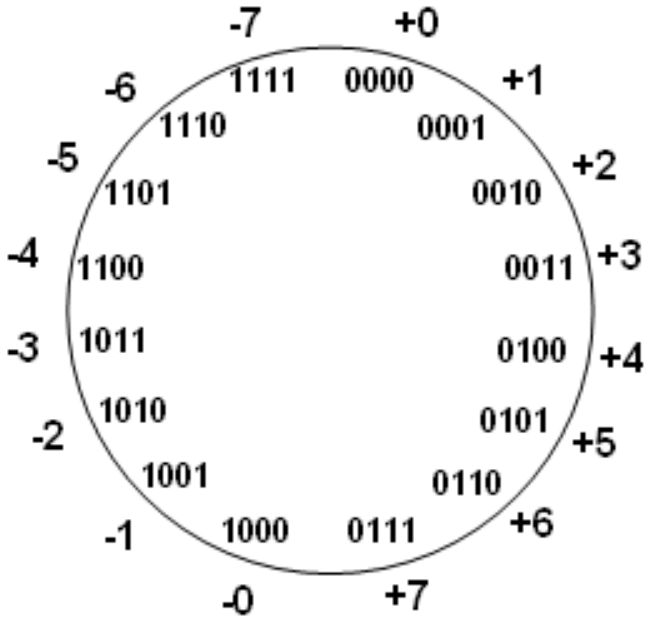
2^8 *возможных значений*



integer

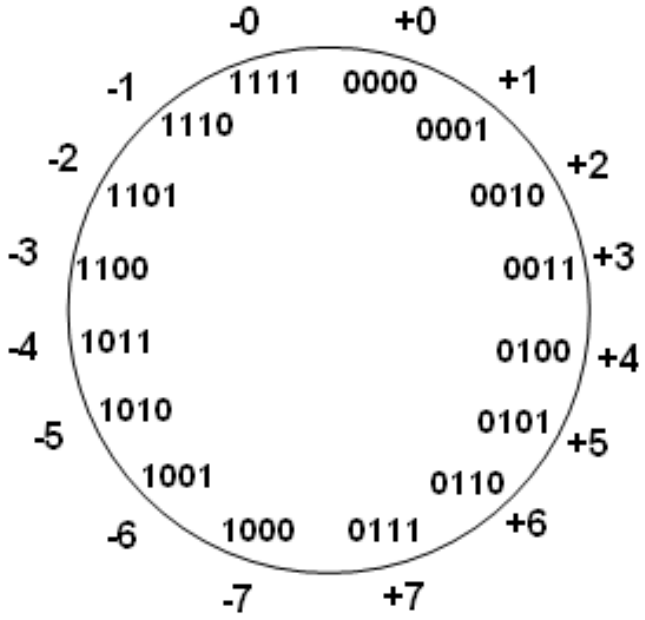
Хранятся как последовательности из бит

Представление отрицательных чисел



Прямой код

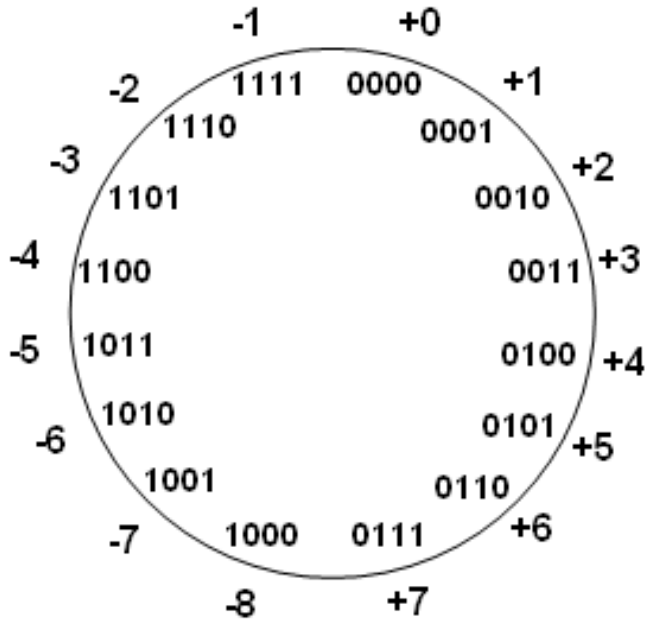
старший разряд числа показывает знак, остальные разряды содержат число



Обратный код

-X = NOT(X)

отрицательное число - это инвертированное положительное



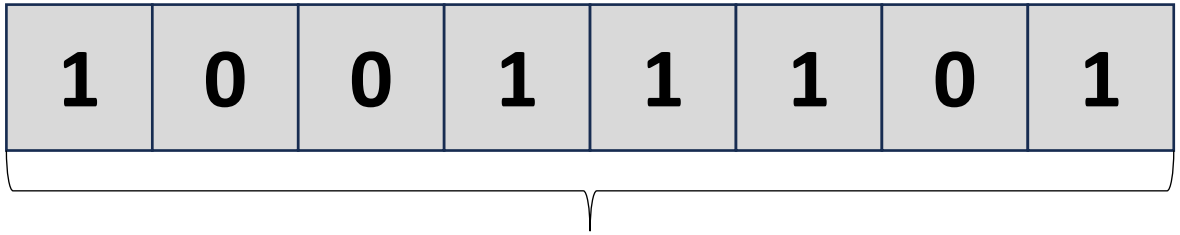
Дополнительный код

-X = NOT(X) + 1

отрицательное число - это инвертированное положительное + 1

Символы и строки

А как другие символы? Русские буквы?



ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

А как хранить дробные числа?

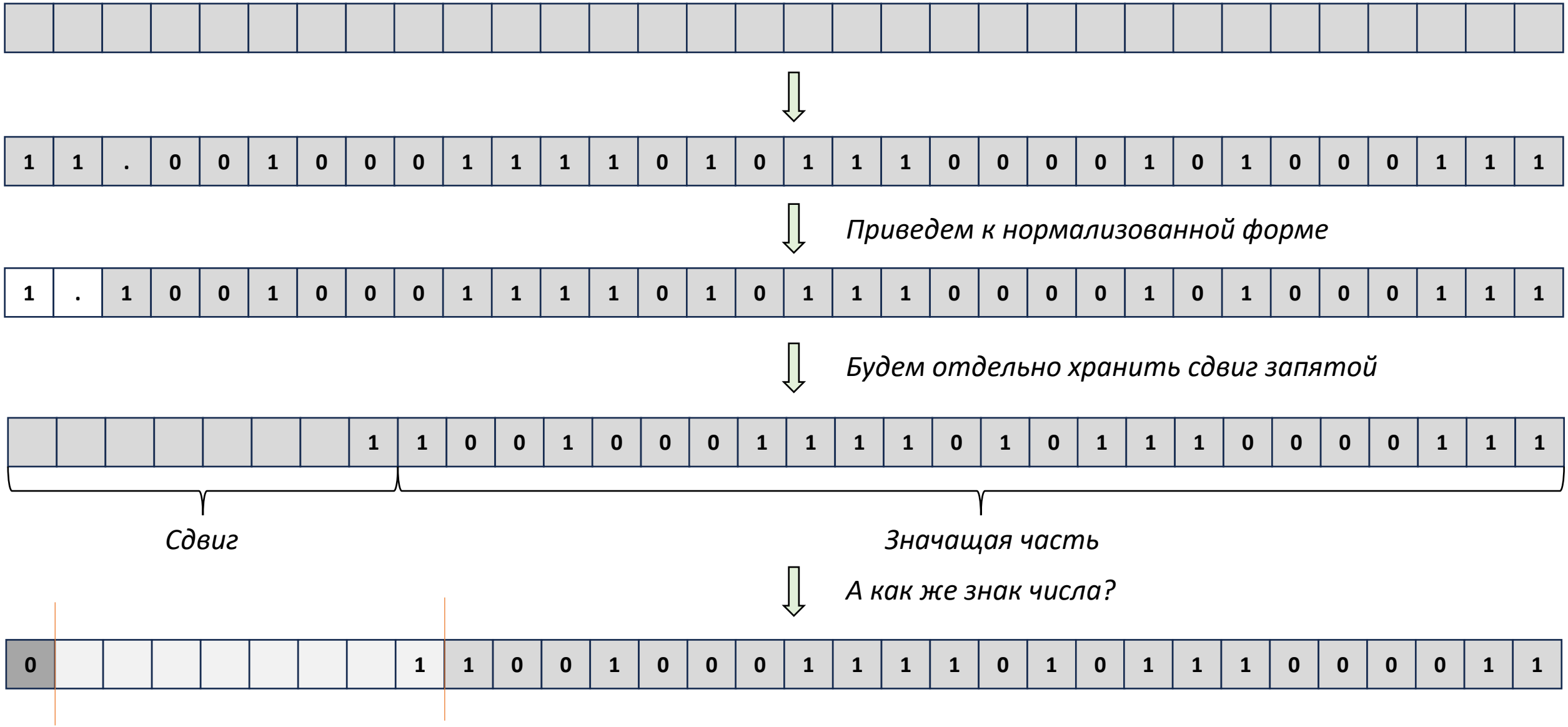
1.25_{10} -> 1.01_2

125.125_{10} -> 1111101.001_2

3.14_{10} -> $11.001000111101\ldots_2$

Как хранить такие дроби?

Идея!


1 1 . 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 1 0 1 0 0 0 1 1 1
Приведем к нормализованной форме
1 . 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 1 0 1 0 0 0 1 1 1
Будем отдельно хранить сдвиг запятой
1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 1 1 1
Сдвиг
Значащая часть
А как же знак числа?
0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 1 1

Продолжаем идею

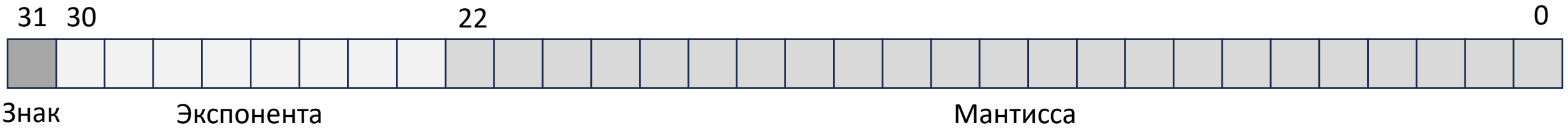
А как же знак степени?

Сдвинемся на 127

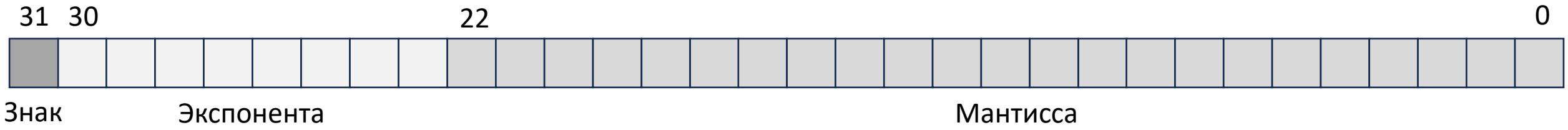
$$x = (-1)^{sign} \times 1.mant \times 2^{exp-127}$$

Представление дробных чисел

$$x = (-1)^{sign} \times 1.mant \times 2^{exp-127}$$



Теперь подумаем, к чему это приведет



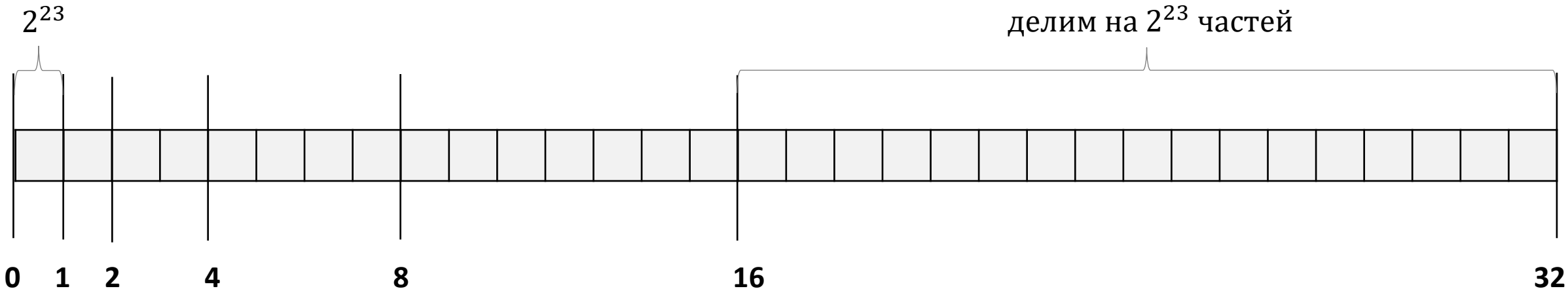
- Чем больше число, тем меньше знаков останется под хранение дробной части

$$12345{,}12345_{10} \rightarrow 1100000011001.000111111001_2$$
$$\rightarrow 1.10000001110010001111110\mathbf{01}_2$$
$$12345{,}12349_{10}$$

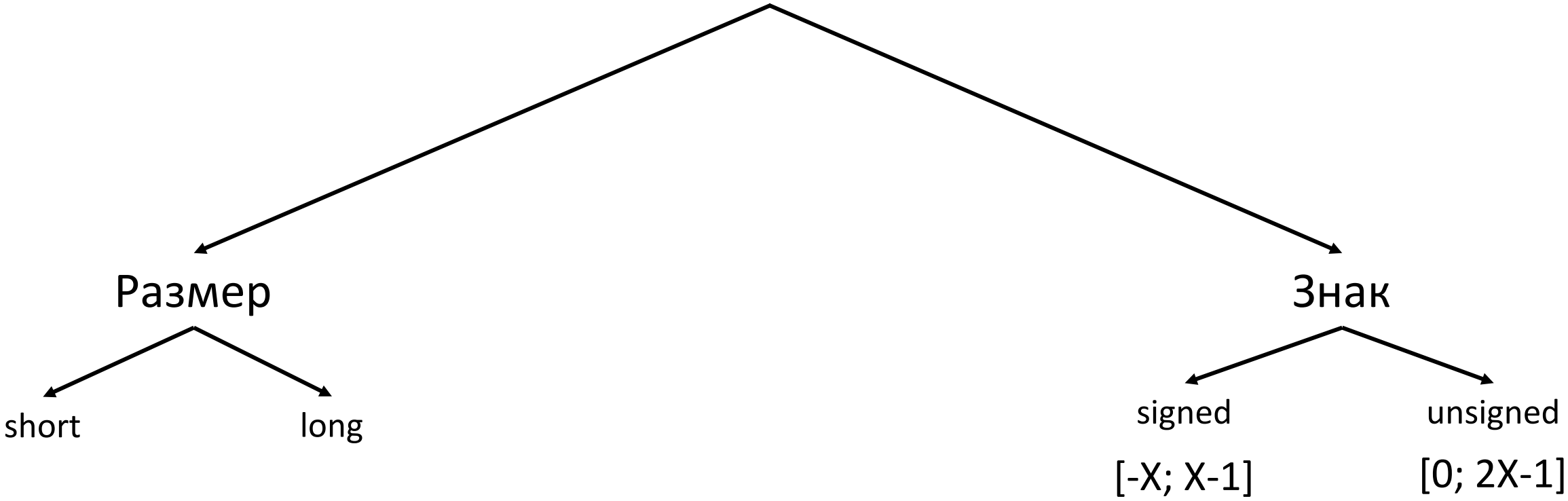
$$12345678{,}12345678_{10} \rightarrow 101111000110000101001110.000111111001_2$$
$$\rightarrow 1.01111000110000101001110\mathbf{000111111001}_2$$
$$12345678{,}\mathbf{1234678}_{10}$$

Теперь подумаем, к чему это приведет

- **Экспонента** – на сколько сдвинули запятую. Т.е. между какими степенями двойки лежит наше число
- **Мантисса** – на сколько мы сдвинулись между этими числами. Максимум мы можем разбить отрезок на 2^{23}
- Очевидно, что чем длиннее отрезок, то тем меньше точность при разбиении



Хранение данных. Модификаторы.



Хранение данных. Модификаторы.

Размер:

- short
- long

Знак:

- signed ->[-X; X-1]
- unsigned ->[0; 2X-1]

MS-DOS (DOSBox)

```
C:\PROGRAM>c:\Borlandc\bin\bc.exe
Sizeof  int = 2 bytes
Sizeof  char = 1 bytes
Sizeof  float = 4 bytes
Sizeof  double = 8 bytes

        With modifers

Sizeof  short  int = 2 bytes
Sizeof  long  int = 4 bytes
Sizeof  long long int = 4 bytes
```

Win10

```
C:\Users\79629\Programs>varsize.exe
Sizeof int = 4 bytes
Sizeof char = 1 bytes
Sizeof float = 4 bytes
Sizeof double = 8 bytes

        With modifers

Sizeof short int = 2 bytes
Sizeof long int = 4 bytes
Sizeof long long int = 8 bytes
```

Хранение данных

Тип	Размер в байтах (битах)	Интервал изменения
unsigned char	1 (8)	от 0 до 255
signed char (char)	1 (8)	от -128 до 127
unsigned short	2 (16)	от 0 до 65 535
signed short (short)	2 (16)	от -32 768 до 32 767
unsigned int (unsigned)	4 (32)	от 0 до 4 294 967 296
signed int (int)	4 (32)	от -2 147 483 648 до 2 147 483 648
unsigned long	4 (32)	от 0 до 4 294 967 295
signed long (long)	4 (32)	от -2 147 483 648 до 2 147 483 647
unsigned long long	8 (64)	от 0 до 18 446 744 073 709 551 615
signed long long (long long)	4 (32)	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
float	4 (32)	от 3.4E-38 до 3.4E+38
double	8 (64)	от 1.7E-308 до 1.7E+308
long double	10 (80)	от 3.4E-4932 до 3.4E+4932

Преобразование типов

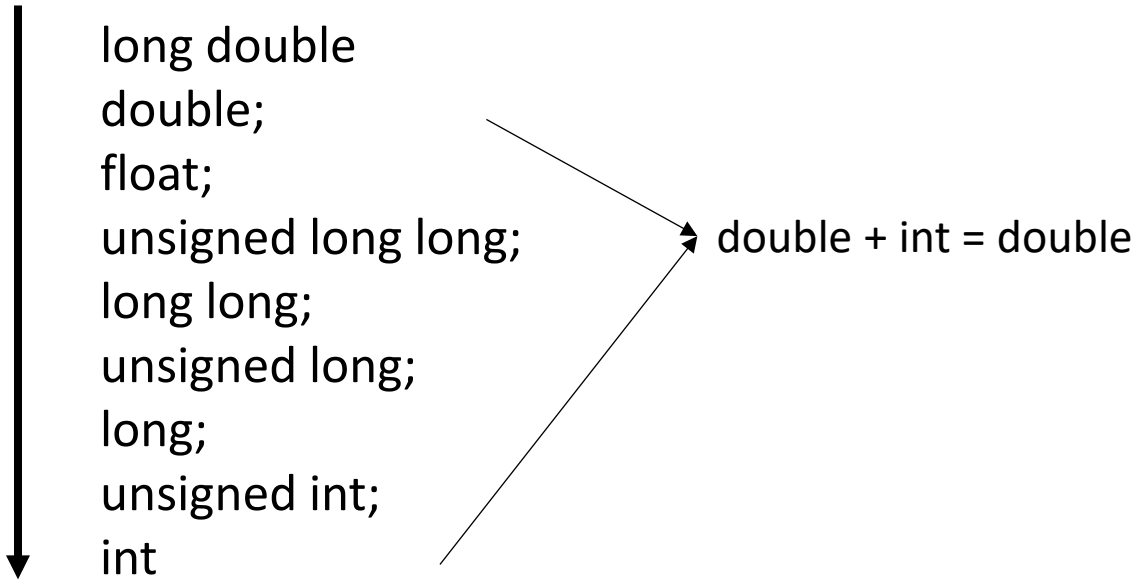
```
int i;
char c = 45;
i = (int)c;
```

Неявное преобразование

Явное преобразование

типы операндов	тип результата
float / float	float
float / int	float
int / float	float
int / int	int

Приоритет операций:



Вывод данных

printf()	
Заголовочный файл	stdio.h
Входные данные	(const char *format, ...)
Возвращаемое значение	отрицательное значение — признак ошибки; в случае успеха функции возвращают количество записанных/выведенных байтов
Пример использования	printf("Good %s%c my dear %d students\n","evening",',',4);
	Good evening my, dear 4 students

Вывод данных

Код	Формат
%c	Символ типа char
%d	Десятичное число целого типа со знаком
%i	Десятичное число целого типа со знаком
%e	Научная нотация (e нижнего регистра)
%E	Научная нотация (E верхнего регистра)
%f	Десятичное число с плавающей точкой
%g	Использует код %e или %f — тот из них, который короче (при использовании %g используется e нижнего регистра)
%G	Использует код %E или %f — тот из них, который короче (при использовании %G используется E верхнего регистра)
%o	Восьмеричное целое число без знака
%s	Строка символов
%u	Десятичное число целого типа без знака
%x	Шестнадцатиричное целое число без знака (буквы нижнего регистра)
%X	Шестнадцатиричное целое число без знака (буквы верхнего регистра)
%p	Выводит на экран значение указателя
%n	Ассоциированный аргумент — это указатель на переменную целого типа, в которую помещено количество символов, записанных на данный момент
%%	Выводит символ %

Ввод данных

scanf()	
Заголовочный файл	stdio.h
Входные данные	(const char *format, ...)
Возвращаемое значение	отрицательное значение — признак ошибки; в случае успеха функции возвращают количество записанных/выведенных байтов

Пример использования	int num_of_students;
	char comma;
	char line[8];
	scanf("%s",line);
	scanf(" %c",&comma);
	scanf("%d",&num_of_students);
	printf("Good %s%c my dear %d students\n",line,comma,num_of_students);

	Good evening my, dear 4 students

Ввод данных

Только для Visual Studio

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int var;
    printf("Enter number: \n");
    scanf("%d", &var);
    printf("%d\n", var);
    return 0;
}
```

Обработка. Ветвление

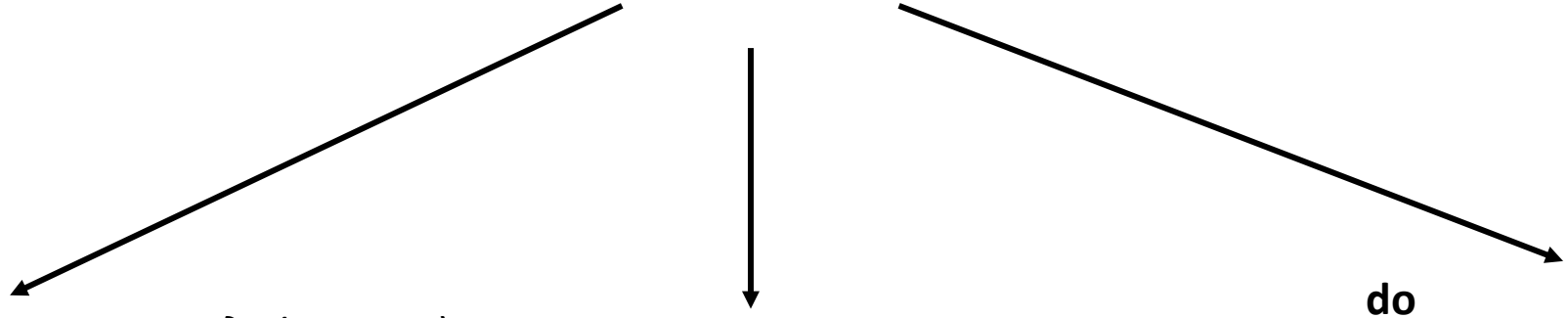
Неполная развилка

```
1  if (Условие)
2  {
3    БлокОпераций1;
4  }
```

Полная развилка

```
1  if (Условие)
2  {
3    БлокОпераций1;
4  }
5  else
6  {
7    БлокОпераций2;
8  }
```

Обработка. Циклы



```
for (Инициализация; Условие; Модификация)
{
  БлокОпераций;
}
```

```
while (Условие)
{
  БлокОпераций;
}
```

```
do
{
  БлокОпераций;
}
while (Условие);
```

```
for (int i=0; i <10; i++)
{
  printf("%d. Hello MIET!\n", i);
}
```

```
int i = 0;
while (i <10)
{
  printf("%d. Hello MIET!\n", i);  i++;
}
```

```
int i = 0;
do
{
  printf("%d. Hello MIET!\n", i);
  i++;
}
while (i <10)
```