Представление данных в памяти компьютера. Числа с плавающей точкой

(стандарт IEEE 754)

Числа с плавающей точкой

Число с плавающей запятой состоит из набора отдельных двоичных разрядов, условно разделенных на **знак**, **порядок** и **мантиссу.**

Любое число *N* в системе счисления с основанием *q* можно записать в виде

$N = M \cdot q^p$, где *M* — множитель, содержащий все цифры числа (мантисса), а *p* — целое число, называемое порядком.

Длина *мантиссы* определяет *точность* числа.

Длина *порядка* — его *диапазон*.

Распределение битов в вещественных типах данных языка C++

1 бит – знак

Тип **float** (одинарный – 32 разряда):

8 бит – порядок,

23 бита – мантисса,

Тип **double**: (двойной – 64 разряда):

11 бит – порядок,

52 бита – мантисса,

Тип **long double**: (расширенный – 80 разрядов):

15 бит – порядок,

64 бита – мантисса,

Тип **long double**: (расширенный – 128 разрядов):

15 бит – порядок,

112 бит – мантисса,

Точность действительных чисел

- точность float: от 6 до 9 цифр (в основном 7);

  23 бита мантиссы –> 2^23 = 8388608

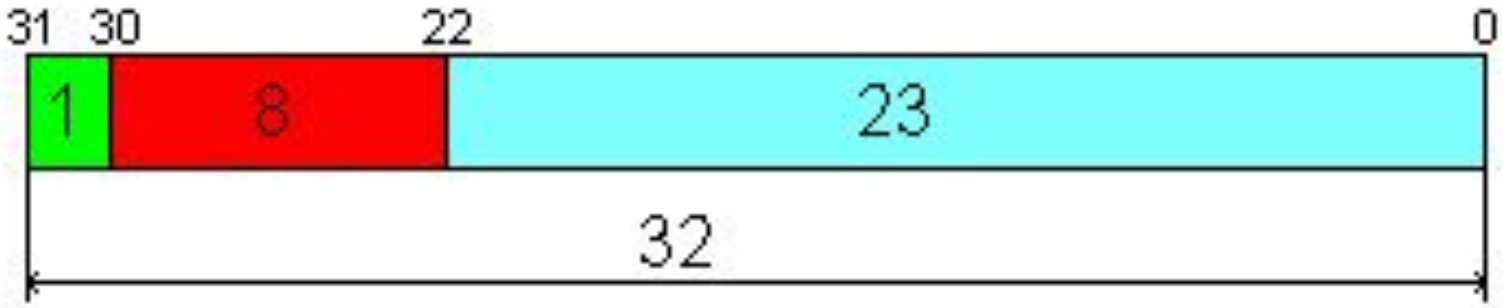
- точность double: от 15 до 18 цифр (в основном 16);

  52 бита мантиссы = 4503599627370496

- точность long double: 15, 18 или 33 цифры (в зависимости от того, сколько байт занимает тип данных на компьютере).

  64 бита мантиссы = 18446744073709551616

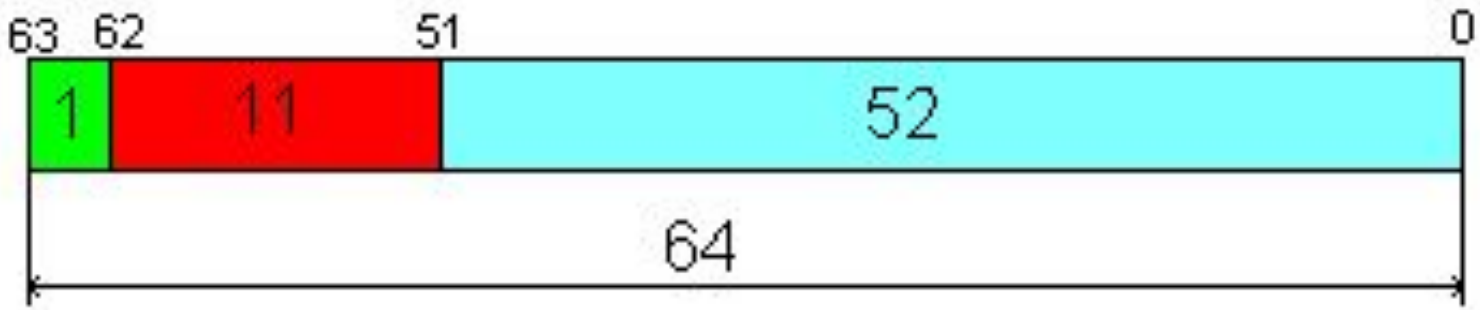
Формат числа одинарной точности (32 бита)



$$F = (-1)^S 2^{(E-127)} (1 + M / 2^{23})$$

Для определения знака экспоненты, чтобы не вводить ещё один бит знака, добавляют смещение к экспоненте в половину байта

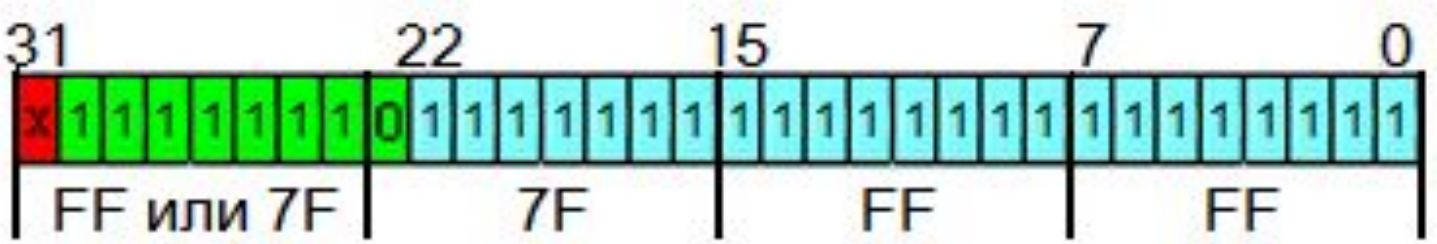
Формат числа двойной точности (64 бита)



$$F = (-1)^S 2^{(E-1023)} (1 + M / 2^{52})$$

Максимальное нормализованное число

$$F = (-1)^S 2^{(E-127)} (1 + M / 2^{23})$$



$$7F\ 7F\ FF\ FF = 2^{127} \cdot (1 + (2^{23} - 1) / 2^{23}) = 2^{127} \cdot (2 - 2^{-23}) \approx 3{,}40282347 \cdot e^{+38}$$

Округление при выводе

Стандартный вывод делает округление результата до 6 знаков

```
double t;
cin >> t; //123.4567
cout << t; // 123.457
```

Манипулятор setprecision()* позволяет задать точность вывода, тем не менее само значение может быть приближенным и зависит от точности, которую может обеспечить мантисса

```
cin >> t;                    // 123456789.0123456789
cout << setprecision(19) << t; // 123456789.0123456717
```

*манипулятор определен в заголовочном файле iomanip

Ошибки округления

Из-за ограничений по памяти некоторые дроби представлены в виде округленного усечения, а математические операции над такими значениями только увеличивают ошибки округления

```
float t;
cin >> t;                    // 0.1
cout << setprecision(15) << t; // 0.100000001490116
cout << t + t + t + t + t + t; // 0.600000023841858
```

Сравнение значений с плавающей запятой

Даже самая маленькая ошибка округления приведет к тому, что два числа с плавающей запятой не будут равны

Оператор == имеет высокий риск возврата false, когда можно было бы ожидать true (аналогично для !=)

Проверка на равенство значений с плавающей запятой обеспечивается использованием «эпсилон».

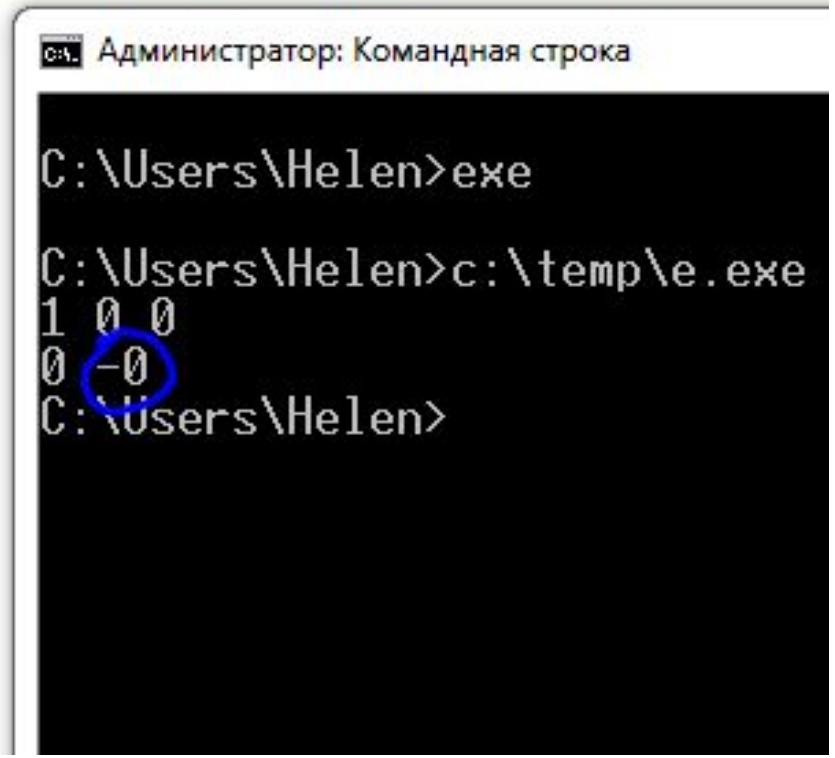
```
float x, y, eps = 1E-6; // eps = 0.000001
cin >> x >> y;
if (abs(x - y) <= eps)
   cout << "equal";
else
   cout << "not equal";
```

Значения nan и inf

```
double zero = 0.0;
double posinf = 1.0 / zero;
double neginf = -1.0 / zero;
double nan = zero / zero;
cout << posinf << endl
    << neginf << endl
    << nan << endl;
```

Знак у числа 0

```cpp
#include <iostream>
#include<cmath>
using namespace std;
int main(){
double a,b,c;
cin>>a>>b>>c;
double d=b*b-4*a*c;
if(d>=0){
double x1=(-b+sqrt(d))/2/a;
double x2=(-b-sqrt(d))/2/a;
cout << x1<<' '<<x2;
}
else
cout << "No real roots";
}
```



Источники

http://www.softelectro.ru/ieee754.html

https://en.wikipedia.org/wiki/Extended_precision