

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра Название кафедры

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Основы алгоритмизации и программирования на языке
С++»
Тема: Типы данных и их внутреннее представление в памяти

Студент гр. 1324

Скопцов В.В.

Преподаватель

Глущенко А.Г.

Санкт-Петербург

2021

Цель работы.

Разработать алгоритм и написать программу, которая позволяет:

1) Вывести, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.

2) Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.

3) Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

4) Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

Основные теоретические положения.

В знаковом целом типе могут быть представлены положительные и отрицательные числа, а также ноль. Положительные числа и ноль в знаковом типе хранятся точно так же, как в беззнаковом – в двоичном виде. Знаковые отрицательные числа хранятся в компьютере в дополнительном коде. Рассмотрим следующий алгоритм.

Обозначим n – разрядность используемого типа данных, X – отрицательное число.

1. Найдем разность: $y = 2^n - |X|$

2. Переведем y в двоичную систему счисления.

Введем одно важное правило: если число представлено в памяти в знаковом типе, то старший (крайний левый) бит отвечает за знак числа: если старший бит равен 0, то число положительное, если 1, то отрицательное. Поэтому в знаковых типах старший бит называется Signum (лат. Знак).

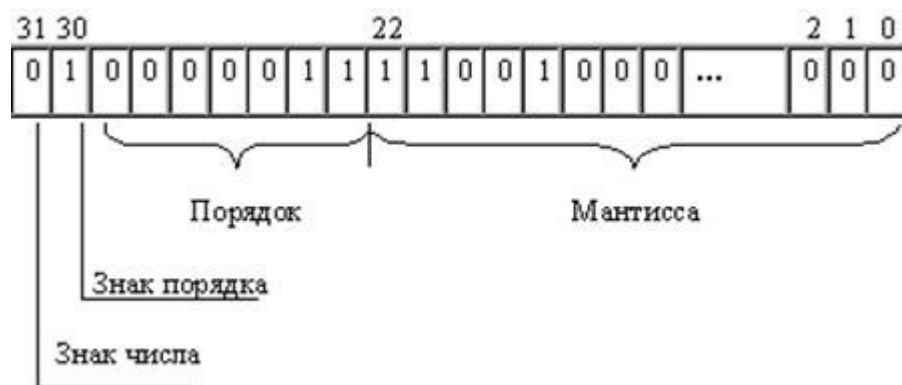
Для представления вещественных чисел в памяти компьютера разработан стандарт IEEE 754-2008 [1], в котором описаны форматы представления чисел, алгоритмы округления, операции над числами, обработка исключений (деление на ноль, переполнение и др.). В этом стандарте используется форма представления вещественных чисел с плавающей запятой. В этой форме вещественное число X записывается в следующем виде: $X = \pm m \cdot q^{\pm p}$ где m – мантисса числа (лат. добавление, прибавка), q – основание системы счисления, p – порядок. Для определенности вещественные числа в компьютерах представляются в нормализованном виде, т. е. запятая расположена справа от первой ненулевой цифры.

В двоичном нормализованном представлении числа старший бит, расположенный слева от запятой, всегда равен 1, поэтому его можно не хранить (т. н. «скрытая единица»), а диапазон нормализованных мантисс получается следующим: $1 \leq m < 2$.

Рассмотрим основные форматы представления вещественных чисел, описанные в стандарте IEEE 754-2008. Они отличаются размером ячейки памяти, отводимой для одного числа:

- 1) формат одинарной точности (single precision, 4 байт);
- 2) формат двойной точности (double precision, 8 байт);
- 3) формат расширенной точности (extended precision, 10 байт).

В формате одинарной точности под число отводится 32 бита, которые распределены следующим образом:



В этом формате указывается только знак всего числа: если бит знака равен 0, число положительное, если 1 – отрицательное. Знак порядка не указывается, так как применяется смещенный порядок sp , который всегда неотрицателен. Смещенный порядок получается прибавлением к исходному порядку константы 127: $sp = p + 127$

Таким образом, для представления вещественного числа X в формате одинарной точности используется следующая формула (в десятичном виде):

$$X = \pm 1, m \cdot 2^{sp-127}$$

В формат двойной точности: под все число отводится 64 бита (8 байт), из них 1 бит – знак, 11 бит – смещенный порядок, 52 бита – мантисса, смещенный порядок вычисляется по формуле: $sp = p + 1023$

Результаты выполнения.

Примеры выполнения программ:

```

1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     cout << sizeof(int) << endl;
6     cout << sizeof(short int) << endl;
7     cout << sizeof(long int) << endl;
8     cout << sizeof(float) << endl;
9     cout << sizeof(double) << endl;
10    cout << sizeof(long double) << endl;
11    cout << sizeof(char) << endl;
12    cout << sizeof(bool) << endl;
13    return 0;
14 }

```

```

> ТЕРМИНАЛ
-rwxrwxr-x 1 hav hav 16672 окт 15 15:18 a.out*
drwxrwxr-x 8 hav hav 4096 окт 13 21:17 .git/
-rw-rw-r-- 1 hav hav 359 окт 15 15:16 laba1.cpp
-rw-rw-r-- 1 hav hav 433 окт 11 18:32 laba2.cpp
-rw-rw-r-- 1 hav hav 1488 окт 13 20:53 laba3.cpp
-rw-rw-r-- 1 hav hav 1490 окт 13 21:04 laba4.cpp
-rw-rw-r-- 1 hav hav 160 окт 11 15:23 test.cpp
hav@hav-N53SV:~/Музыка/Praktika/Praktika$ ./a.out
4
2
8
4
8
16
1
1
hav@hav-N53SV:~/Музыка/Praktika/Praktika$

```

Рисунок 1: Лабораторная работа пункт 1

```
laba2.cpp > main()
4 int main()
5 {
6     long int A[32] = {}, number;
7     std::cin >> number;
8     if (number < 0) number = pow(2, 32) - abs(number);
9     for (int i=0; i<32; i++)
10     {
11         A[31-i] = number % 2;
12         number = number / 2;
13     }
14     for (int i=0; i<32; i++)
15     {
16         if (i%8==7 || i==0)
17             std::cout << A[i] << " ";
18         else
19             std::cout << A[i];
20     }
21     return 0;
22 }
```

ПРОБЛЕМЫ Выходные данные ТЕРМИНАЛ

TERMINAL

```
> -rwxrwxr-x 1 hav hav 16760 окт 15 15:19 a.out*
drwxrwxr-x 8 hav hav 4096 окт 13 21:17 .git/
-rw-rw-r-- 1 hav hav 359 окт 15 15:18 laba1.cpp
-rw-rw-r-- 1 hav hav 433 окт 11 18:32 laba2.cpp
-rw-rw-r-- 1 hav hav 1488 окт 13 20:53 laba3.cpp
-rw-rw-r-- 1 hav hav 1490 окт 13 21:04 laba4.cpp
-rw-rw-r-- 1 hav hav 160 окт 11 15:23 test.cpp
hav@hav-N53SV:~/Музыка/Praktika/Praktika$ ./a.out
127
0 0000000 00000000 00000000 01111111 hav@hav-N53SV:~/Музыка/Praktika/Praktika$
```

Строка 21, столбец 14 Пробелов: 4 UTF-8 LF C++ Linux

Рисунок 2: Лабораторная работа пункт 2

```
laba3.cpp > main()
1 #include <iostream>
2 #include <cmath>
3
4 int main()
5 {
6     int Mantissa[23] = {}, Exp[8] = {};
7     float number;
8     bool sign;
9     std::cin >> number;
10    sign = number < 0;
11    if (sign) number = -number;
12    long int integer_part = (int)number, exp = 0;
13    float fractional_part = number - integer_part;
14    integer_part /= 2;
15    for (int i = 0; i < 24; i++)
16    {
17        if (integer_part < pow(2, i))
18        {
19            exp = i;
20        }
21    }
22 }
```

ПРОБЛЕМЫ Выходные данные ТЕРМИНАЛ

TERMINAL

```
> -rwxrwxr-x 1 hav hav 16840 окт 15 15:20 a.out*
drwxrwxr-x 8 hav hav 4096 окт 13 21:17 .git/
-rw-rw-r-- 1 hav hav 359 окт 15 15:18 laba1.cpp
-rw-rw-r-- 1 hav hav 433 окт 11 18:32 laba2.cpp
-rw-rw-r-- 1 hav hav 1488 окт 13 20:53 laba3.cpp
-rw-rw-r-- 1 hav hav 1490 окт 13 21:04 laba4.cpp
-rw-rw-r-- 1 hav hav 160 окт 11 15:23 test.cpp
hav@hav-N53SV:~/Музыка/Praktika/Praktika$ ./a.out
3.14
0 10000000 10010001111010111000011hav@hav-N53SV:~/Музыка/Praktika/Praktika$
```

Строка 22, столбец 6 (выбрано 6) Пробелов: 4 UTF-8 LF C++ Linux

Рисунок 3: Лабораторная работа пункт 3

4) Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.