ФГОБУ ВПО "СибГУТИ"
Кафедра вычислительных систем

Дисциплины
"ЯЗЫКИ ПРОГРАММИРОВАНИЯ"
"ПРОГРАММИРОВАНИЕ"

Практическое занятие №11

Программное исследование внутреннего представления чисел в вычислительной технике.

Преподаватель:

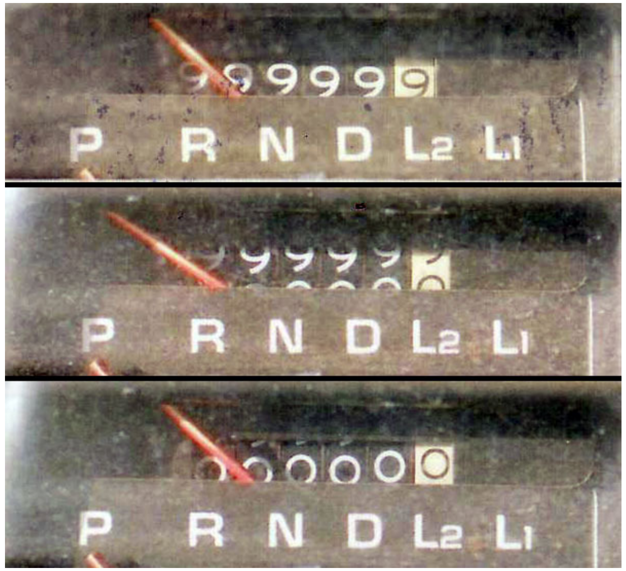
Доцент Кафедры ВС, к.т.н.

Поляков Артем Юрьевич

Переполнение целой беззнаковой переменной

Ситуация, в которой результат операции вычитания меньше *значения* переменной, обрабатывается путем дописывания виртуального старшего разряда к уменьшаемому:

$$\mathbf{0 - 1 = 1\,0000\,0000_2 - 1 = 1111\,1111_2}$$



1	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1

Аналогия с механическим одометром: набором шестерней, связанных друг с другом определенным передаточным числом

C11.01 Переполнение беззнакового целого

Разработать программу, демонстрирующую переполнение беззнакого целого, используемого в качестве счетчика в цикле. На каждом шаге значение счетчика увеличивается на 1.

1. Разработайте программу для однобайтового беззнакового целого.

1.1. Чему равно минимальное количество итераций цикла, необходимое для демонстрации переполнения, если начальное значение счетчика 0?

1.2. Чему равно минимальное количество итераций цикла, необходимое для демонстрации переполнения, если начальное значение счетчика 100?

2. Разработайте программу для двухбайтового беззнакового целого. Ответьте на вопросы 1.1 и 1.2 для этой реализации.

Целая знаковая переменная

Как было сказано ранее для представления знака в ВТ не предусмотрено специальных средств. Знаковые числа представляются в дополнительном коде. При этом знак числа определяется старшим битом целочисленной ячейки.

1	1	0	1	1	0	0	1

Целое беззнаковое: 217
Целое знаковое: -39
256 – 39 = 217

$$x' = \begin{cases} x, & x \geq 0 \\ 0 - |x| = 100000000_2 - |x| = 256 - |x|, & x < 0 \end{cases}$$

$$x = \begin{cases} x', & x \leq 127 \\ x' - 100000000_2 = x' - 256, & x > 127 \end{cases}$$

Целая знаковая переменная (2)

Как было сказано ранее для представления знака в ВТ не предусмотрено специальных средств. Знаковые числа представляются в дополнительном коде. При этом знак числа определяется старшим битом целочисленной ячейки.

1	1	0	1	1	0	0	1

Целое беззнаковое: 217
Целое знаковое: -39
256 – 39 = 217

```
char c = -39;
printf("char: %hhd\n", c);
printf("unsigned char = %hhu\n", c);
```

Математика: –39 = – 39
Вычислительная техника: – 39 = 217

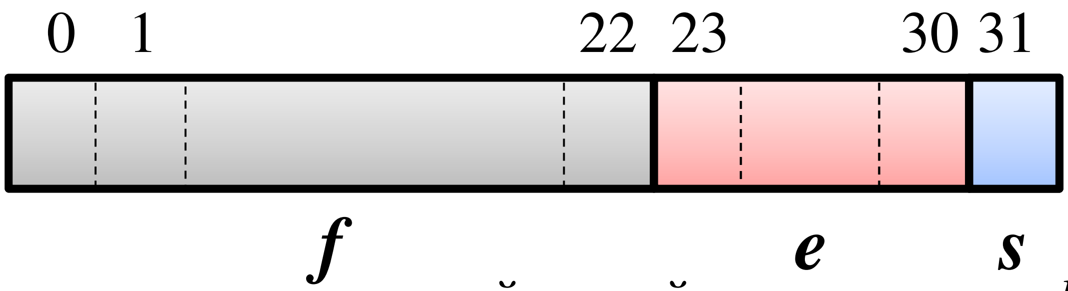
C11.02 Представление знаковых чисел

Разработайте интерактивную программу, демонстрирующую внутреннее (беззнаковое) представление знаковых однобайтовых целых. Программа взаимодействует с пользователем через простейшее меню. Пример сеанса работы программы приведен ниже (красным цветом выделены данные, вводимые пользователем):

```
Want to continue? (1/0): 1
Input signed integer: -39
Unsigned representation: 217
Want to continue? (1/0): 1
Input signed integer: -38
Unsigned representation: 218
Want to continue? (1/0): 0
Exiting!
```

Используя разработанную программу докажите справедливость соотношений, приведенных на слайде 4.

Числа с плавающей точкой



p-разрядным числом с плавающей точкой по основанию *b* с избытком *q* называется пара величин (e,f), которой соответствует значение:

$$(e, f) = f \cdot b^{(e-q)}$$

e – порядок – ***беззнаковое*** целое число.

f – мантисса – нормализованное знаковое с ***фиксированной точкой***.

q – избыток, для знакового представления ***порядка***.

Распространенным методом нормализации является приведение числа к виду, в котором целая часть является нулевой:

1) наиболее значимая цифра в представлении *f* отлична от нуля:

$$1/b \leq |f| < 1 \ (1/10 = 0.1 \leq |f| < 1)$$

2) $f = 0$ и e принимает наименьшее возможное значение

Например: Только выделенное представление числа 650 нормализовано:

$\underline{6500} \cdot 10^{-1}$, $\underline{650}$, $\underline{65}.0 \cdot 10$, $\underline{6}.5 \cdot 10^2$, $\mathbf{0.65 \cdot 10^3}$, $0.\underline{0}65 \cdot 10^4$

Представление вещественных чисел (2)

Размер, выделяемый для хранения: 4 байта (32 бита)

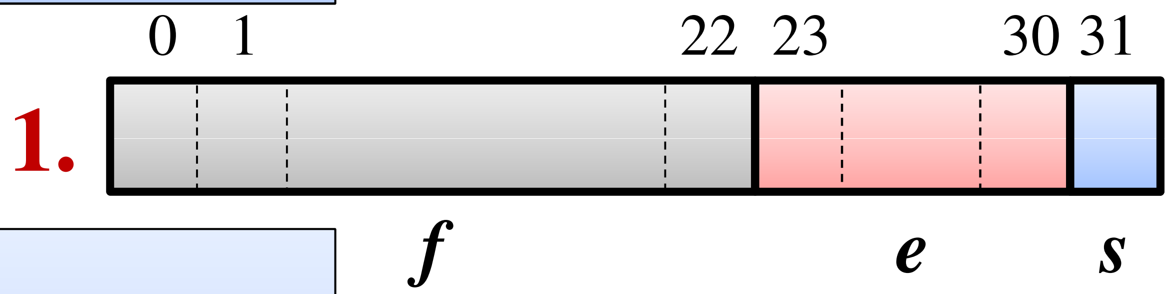
$$(e, f, s) = s\mathbf{1}.f \cdot b^{e-q}$$

f: 23 бит; e: 8 бит; s: 1 бит,

$b = 2$, $q = 2^{8-1} - 1 = 127$

Стандарт IEEE 754-2008 Single precision

Мантисса
23 **реальных** разряда
24 **виртуальных** разряда



Недостаток:
мантисса *не может быть нулевой*!
0 представляется наименьшим возможным числом, представимом в данном формате

Нормализованная дробь (IEEE 754-2008) :

$$1_2 \leq |f| < 10_2$$

Например:

$10101.11 \rightarrow \mathbf{1}.010111 \cdot 2^4$,

$0.0001010 \rightarrow \mathbf{1}.01 \cdot 2^{-4}$

Машинный ноль

Идентификатор	Размер, байт	Диапазон значений
float	4	от $\pm 3.4 \cdot 10^{-38}$ до $\pm 3.4 \cdot 10^{38}$ (~ 7 значащих цифр)

```
#include <stdio.h>

int main()
{
    float f = 1;
    double d = 1;
    int i;
    for(i=0;i<160;i++){
        f /= 2;
        d /= 2;
        printf("%d: %e = %le\n", i, f, d);
    }
    return 0;
}
```

C11.03/H11.01 Характеристики типа double

Разработайте программу, позволяющую для типа double экспериментально определить :

1) разрядность порядка и мантиссы;

2) диапазон допустимых значений;

Указания к решению:

1. За основу взять программы вычисления машинного нуля и машинной бесконечности, рассмотренные в лекции №4 (в качестве эталонного значения использовать тип long double).

2. Для того, чтобы вычислить разрядности мантиссы и порядка исследовать два на количество возможных делений пополам:

а) 1, многократное деление этого числа на 2 приведет сначала к уменьшению порядка, после того, как порядок достигнет минимально возможного значения, виртуальная 1 будет игнорироваться и дельнейшее уменьшение числа будет происходить за счет мантиссы;

б) Число 1.9999999999999997779553950749686919152736663818359375, все разряды мантиссы которого единичны (в double-представлении), следовательно когда порядок достигнет минимально возможного значения, дальнейшее деление приведет к падению точности и, следовательно, отличиям между эталонным значением(long double) и анализируемым значением (double).

Н11.02 Арифметический корень

Для квадратов чисел верны следующие равенства:

$$1 = 1^2$$
$$1 + 3 = 4 = 2^2$$
$$1 + 3 + 5 = 9 = 3^2$$
$$1 + 3 + 5 + 7 = 16 = 4^2$$
$$\ldots$$

Используя данное свойство можно арифметически вычислить целую часть корня. Пусть x – число, для которого необходимо найти корень.

1. **Если** $(x - 1) = 0$, то $\sqrt{x} = 1$, **если** $(x - 1) < 0$, то $[\sqrt{x}] = 0$, **иначе** $x = x - 1$
2. **Если** $(x - 3) = 0$, то $\sqrt{x} = 2$, **если** $(x - 3) < 0$, то $[\sqrt{x}] = 1$, **иначе** $x = x - 3$
3. **Если** $(x - 5) = 0$, то $\sqrt{x} = 3$, **если** $(x - 5) < 0$, то $[\sqrt{x}] = 2$, **иначе** $x = x - 5$
4. **Если** $(x - 7) = 0$, то $\sqrt{x} = 4$, **если** $(x - 7) < 0$, то $[\sqrt{x}] = 3$, **иначе** $x = x - 7$

....

Задание:

1. Записать рекуррентное соотношение по которому изменяется вычитаемое и x!
2. Разработать блок-схему алгоритма вычисления **целой части** квадратного корня числа x.
3. Реализовать алгоритм в виде программы на языке Си.

Н11.03 Численный корень

Итерационная формула Герона задает убывающую (начиная со 2-го элемента) последовательность, которая при любом выборе быстро сходится к величине $\sqrt{a}$ (квадратный корень из числа). Рекуррентное соотношение, описывающее данную последовательность выглядит следующим образом:

$$\begin{cases} x_0 = a \\ x_{n+1} = \dfrac{1}{2}\left(x_n + \dfrac{a}{x_n}\right) \end{cases} \qquad \lim_{n\to\infty} x_n = \sqrt{a}$$

Вычисление прекращается, когда ($x_n - x_{n+1}$) < ε. Значения *a* и *ε* являются входными для алгоритма.

Задача:

1. Для чисел 9, 121, 1024 вычислить первые 5 элементов последовательности.
2. Записать блок-схему алгоритма вычисления квадратного корня по формуле Герона.
3. На основе предложенного алгоритма разработать программу.
4. Провести конечные и промежуточные результаты работы программы на числах 9, 121, 1024, 27, 1980.

A11.01 Число с единичной мантиссой

В задаче **C11.03/H11.01** было задано число 1.9999999999999997779553950749686919152736663818359375, особенностью которого является то, что его мантисса полностью состоит из единиц при записи в типе double.

Такое же число можно построить и для типов float и long double.

Задание:

Предложите универсальный алгоритм, позволяющий построить число $x \in [1,2)$, мантисса которого (при представлении его в некотором типе данных с плавающей точкой) будет содержать единицы на всех позициях.

Указание:

Алгоритм: необходимо вычислять разряды по убыванию и прибавлять их к накопителю до тех пор, пока накопитель не перестанет изменяться. Накопитель инициализируется единицей.

Внимание! После переполнения будет произведено округление после которого искомый результат будет потерян, т.е. предложенный алгоритм сделает на 1 шаг больше, чем требуется. В процессе разработки рекомендуется осуществлять отладочный вывод на каждом шаге, что позволит определить скорректировать предложенный алгоритм.

В результате должно быть 3 программы, использующие один алгоритм и отличающиеся только типом ячеек: float, double, long double.

Замечание: этот алгоритм может быть использован для определения размерности мантиссы.