Сегодняшнее занятие достаточно простое — мы научимся работать с числами с плавающей точкой (то есть с действительными числами).

**Как хранятся действительные числа в компьютере**

Для хранения действительных чисел в памяти компьютера отводится определённое количество бит. Действительное число хранится в виде знака (плюс или минус), мантиссы и экспоненты. Что такое мантисса и экспонента лучше объяснить на примере: масса Земли равна 5.972\*1024 килограмм. Здесь 5.972 — мантисса, а 24 — экспонента.

При выводе больших (или очень маленьких) чисел в программе на C++ можно увидеть на экране запись типа 5.972E23. Сначала выводится мантисса, затем — буква E, а затем — экспонента. Запись представлена в десятичной системе счисления. В таком же формате можно вводить большие или очень маленькие действительные числа. Этот формат называется экспоненциальной записью числа.

Мы будем работать с типом double (с числами двойной точности), который занимает 8 байт. Один бит отводится под знак числа, 11 под экспоненту и 52 под мантиссу. С помощью 52 бит можно хранить числа длиной до 15-16 десятичных цифр. Таким образом, независимо от того, какая у числа экспонента, правильные значения будут иметь только первые 15 цифр. В примере с массой Земли точно заданы первые 4 цифры, таким образом, погрешность составляет 1020 килограмм. Это довольно большая погрешность. Чтобы масса Земли с точностью до первых четырёх знаков изменилась, на неё нужно дополнительно поселить миллиард миллиардов довольно упитанных людей.

 Таким образом, можно сказать, что числа в компьютере хранятся не с абсолютной, а с относительной погрешностью (то есть погрешность зависит от значения хранимого числа).

То, что числа хранятся неточно, создаёт нам множество проблем.

**Создание, ввод и вывод действительных переменных**

Работа с действительными переменными очень похожа на работу с целыми числами, только вместо типа int мы используем тип double. Чтобы создать две действительные переменные x и y, нужно написать:

double x, y;

Если мы хотим использовать действительные числа в коде программы, то в качестве разделителя целой и дробной части нужно использовать точку:

y = 3.1415;

Считываются действительные переменные так же, как целые:

cin >> x;

При выводе действительных чисел нужно помнить о некоторых особенностях. В большинстве задач требуется выводить числа с заданной точностью. Например, если от нас требуется вывести число с точностью 10-3, значит, необходимо вывести три знака после десятичной точки. Числа с заданным количеством знаков после точки выводятся так:

cout << setprecision(3) << fixed;

Первая команда устанавливает точность, а вторая сообщает, что все числа нужно выводить с точкой (а не в экспоненциальной форме). После выполнения этой команды все действительные числа будут выводиться с точностью до 3 знаков после точки. Чтобы эти команды работали, необходимо подключить библиотеку iomanip.

**Операции с действительными числами**

Описывая эти операции, мы будем считать, что все числа в арифметических выражениях действительные. Случай, когда в выражении используются целые и действительные числа вперемешку, будет рассмотрен в следующем разделе.

Над действительными числами можно выполнять те же операции, что и над целыми (кроме взятия остатка). При этом деление действительных чисел даёт уже настоящий, не округлённый результат.

Также для действительных чисел определена функция abs, которая возвращает модуль числа.

**Целые и действительные числа в одном арифметическом выражении**

Если мы хотим сохранить в действительной переменной целое число, то достаточно присвоить переменной типа double значение целочисленного арифметического выражения. Есть одна неприятность: если оба операнда (те, что стоят слева и справа от знака операции) целые, то операция будет выполняться в целых числах. Например, если x – действительное число, а, k и m – целые и равны 2 и 3, то после выполнения строки x = k/m в переменной x окажется число 0.0. Деление будет выполнено в целых числах, а уже после этого результат будет преобразован в действительное число. Чтобы избавиться от этой проблемы, необходимо явно привести один из операндов к типу double. Чтобы привести переменную к другому типу в C++ имя типа записывается в круглых скобках перед именем переменной. Чтобы в x оказался правильный результат, можно написать x = (double) k/m. Таким образом, у нас будет происходить вычисление выражения 2.0/3, а если один из операндов будет типа double, то второй также приведётся к этому типу.

**Из действительных в целые**

При переводе из действительных чисел в целые дробная часть теряется. Есть несколько способов потерять дробную часть. Чтобы использовать эти функции, необходимо подключить библиотеку cmath. В нашей программе, кроме #include <iostream>, нужно в следующей строке написать #include <cmath>.

Способ 1

Функция trunc, которая просто отбрасывает дробную часть числа и выглядит так: n = trunc(x);

Способ 2

Функция floor, которая округляет число в меньшую сторону и выглядит так: n = floor(x);

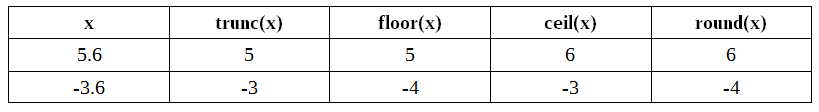
Способ 3

Функция ceil, которая округляет число в большую сторону и выглядит так: n = ceil(x);

Способ 4

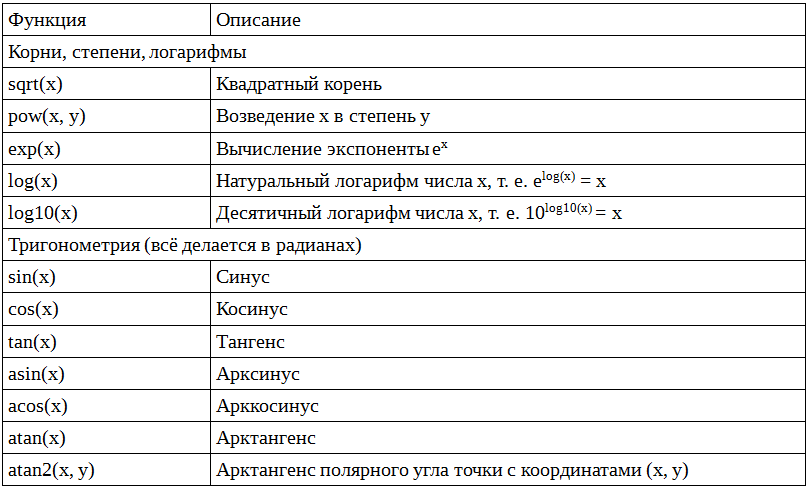
Функция round, которая округляет число по математическим правилам и выглядит так: n = round(x);

Чтобы разница между способами была понятнее, приведём таблицу с примерами:



**Полезные функции библиотеки cmath**

Все функции библиотеки cmath принимают параметры типа double и возвращают значения типа double. Разобьём их по типам.



В C++ нет стандартного имени для константы π. Самый универсальный способ посчитать её, который будет работать во всех версиях компилятора, это взять арктангенс единицы и умножить его на 4. Записывается это как atan(1) \* 4.

**Во всех задачах этого занятия следует выводить числа с шестью знаками после точки.**