ECE 150 Floating-point numbers

Douglas Harder

December 2023

1 Floating-point numbers

A floating-point number is one of four primitive data types in C++, including also integers, characters and Boolean values.
A floating-point number is represented in C++ by the types `float` and `double`. The latter occupies eight bytes (64 bits) and has twice the precision of the former, which only uses four bytes (32 bits), and thus `double` should always be used in engineering applications.
A literal floating-point number can be written into your source code by having one or more decimal digits with one decimal point in any location (so `.325`, `5.42` and `9432.`), but it is preferable to always prefix or suffix a leading or trailing decimal point with a zero for clarity, (so `0.325` and `9432.0`). Another alternative is to append an `e` followed by an integer n to a literal float or an integer, and this represents the first number multiplied by 10.0^n, but in general, it is best to use scientific notation, so one leading digit, a mantissa, and then the exponent, so `3.25e5`, `2e3` or `5.43e-12`. Any floating-point number can be prefixed by a minus sign to represent a negative floating-point number.
While an integer can be exactly represented using integer data types, real numbers can have a non-terminating mantissa, and therefore we cannot represent real numbers on a computer. Floating-point numbers approximate real numbers, and thus, to differentiate the actual real numbers that have infinite precision and the finite floating-point approximations, always refer to floating-point numbers as such, and do not call a `double` a "real" number.

Not on the examination

For integers, `0` equals `-0` (and the latter is simply saved as `0`), but for floating point numbers, `0.0` represents either zero or a very small positive number, and `-0.0` represents either zero or a very small negative number.

2 Arithmetic operations

The arithmetic operations that can be performed on floating-point numbers are `+`, `-`, `*` and `/`. If one operand of any of these binary operators is a floating-point

number and the other is an integer, then the integer will be implicitly converted to a floating-point number before the operation is performed.

Not on the examination

Floating-point arithmetic does not obey the rules of arithmetic for real numbers: it may not be true that $x + (y + z)$ equals $(x + y) + z$, and if $x + y = x$, this does not mean that $y = 0$, it only means that y is significantly smaller than x in absolute value.

There is no modulus operation (`%`) for floating-point numbers, and bit-wise and bit- shifting operations, too, are not defined for floating- point numbers.

3 Comparison operations

The comparison operators are defined for floating-point numbers, and it is always true that `x == y` if and only if `(x - y) == 0.0`.