float or double? (1/3)

- Q: Should we use float or double for real numbers in our programs?
- A: Usually the task statement will indicate the data type you are to use. If it is not given, then it is your choice.
- The double type is more accurate (as it uses more bits) so some people prefer to use it over float. Also, double is the default floating type in C.
  - Recall that to read a double value, you need to use %lf instead of %f in the scanf() function.
  - For writing a double value in a printf() function, %f is good enough (though %lf also works).
  - Refer to Unit 3 slide 21.

float or double? (2/3)

- Example:

```
float a = 2.9;
double b = 2.9;

printf("%.12f\n", a);
printf("%.12f\n", b);
```

Output:

```
2.900000095367
2.900000000000
```

- Note that in general, we do not like or entertain questions such as “what is the output of this program?”
  - We want you to run the program and see its output for yourself.
- Real numbers are stored in computer using floating-point representation, which is covered in CS2100.
  - If you are interested in floating-point number representation, google to find out more. (Eg: http://www.ntu.edu.sg/home/ehchua/programming/java/datarepresentation.html)

float or double? (3/3)

- As double is the default floating type in C, sometimes (quite rarely actually) if you want to force a constant to be float instead of double, you may cast it to float or suffix the value with ‘f’ or ‘F’.
- Examples:
  - 3.456 is of type double (default)
  - 3.456f (or 3.456F) is of type float

```
double a = 3.456f;
double b = 3.456;

printf("%.12f\n", a);
printf("%.12f\n", b);
```

Output:

```
3.456000089645
3.456000000000
```

Non-deterministic output

- There are rules in C, but sometimes the implementation of certain constructs is left to the platform and hence the output could be non-deterministic.
  - That is, when run on different machines, the same program gives different output. (So don't be surprised!)
- Example:

```
printf("%f\n", 5.0/3.0);
printf("%d\n", 5.0/3.0);
printf("%d\n", 5/3);
printf("%f\n", 5/3);
```

Correct

Output on sunfire:

```
1.666667
1073392298
1
0.000000
```

Output may vary on different machines

  - Moral of the story: Use the correct format specifier in your printf() statement.

End of File