COMPUTER SCIENCE 1: STARTING COMPUTING CSCI 1300

Ioana Fleming / Vipra Gupta
Spring 2018
Lecture 25

Announcements

- NO LECTURE – Friday 3/23
  - OHs for H7 and Project Proposal
- Rec 10 due on 3/24
- Hmwk 7 (Project 2)
  - Part II – due on 3/25
- Hmwk 8 (Project 3)
  - Proposal – due 3/19 – deadline passed, see TA
  - Classes & Code Skeleton – due 4/8
  - Final deliverables – due 4/22

Agenda

- Today:
  - Simulations, generating random numbers
  - Classes that contain other classes

Random Numbers and Simulations

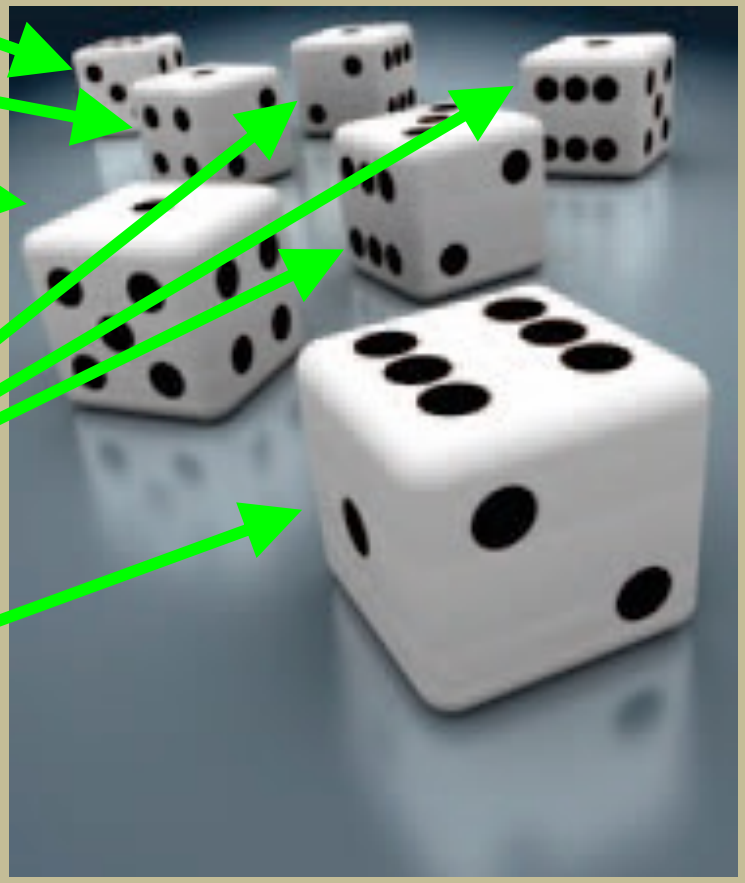
A die toss

another die toss

and another die toss

several tosses of dice

one more die is tossed



Simulations

A *simulation program* uses the computer to simulate an activity in the real world (or in an imaginary one).

- Simulations are commonly used for
  - Predicting climate change
  - Analyzing traffic
  - Picking stocks
  - Many other applications in science and business

Randomness for Reality (Simulating)

- Programmers must model the “real world” at times.
- Consider the problem of modeling customers arriving at a store.

  Do we know the rate?

  Does anyone?

  How about the shopkeeper!

Randomness for Reality (Simulating)

Ask the shopkeeper:

It's about every five minutes
…or so…
…give or a take a couple…
…or three…

…but on certain Tuesdays…

Randomness for Reality (Simulating)

To accurately model customer traffic, you want to take that random fluctuation into account.

How?

The `rand` Function

The C++ library has a random number generator:

`rand()`

The `rand` Function

`rand` is defined in the **`cstdlib`** header

Calling **`rand`** yields a random integer between 0 and **`RAND_MAX`**

(The value of **`RAND_MAX`** is implementation dependent)

The `rand` Function

Calling **`rand`** again yields a different random integer

Very, very, very rarely it might be the same random integer again.

(That's OK. In the real world this happens.)

The `rand` Function

`rand` picks from a very long sequence of numbers that don't repeat for a long time.

But they do eventually repeat.

These sorts of "random" numbers are often called *pseudorandom numbers*.

Example

Run *random.cpp* in Cloud9 … a couple of times

The `rand` Function

`rand` uses only one pseudorandom number sequence

and it always starts from the same place.

Oh dear

The `rand` Function

When you run your program again on another day, the call to **`rand`** will start with:

*the **same** random number!*

Is it very "real world" to use the same sequence over and over?

No, but it's really nice for testing purposes.

but…

Seeding the `rand` Function

You can “seed” the random generator to indicate where it should start in the pseudorandom sequence

Calling **`srand`** sets where **`rand`** starts

`srand` is defined in the **`cstdlib`** header

Seeding the `rand` Function

But what value would be different every *time* you run your program?

(hint)

How about the time?

Seeding the `rand` Function

You can obtain the system time.

Calling **`time(0)`** gets the current time

Note the zero. It is required.

`time` is defined in the **`time`** header

Seeding the `rand` Function

Calling **`srand`** sets where **`rand`** starts.

Calling **`time(0)`** gets the current time.

So, to set up for "really, really random" random numbers on each program run:

```
srand(time(0)); // seed rand()
```

(Well, as "really random" as we can hope for.)

Modeling Using the `rand` Function

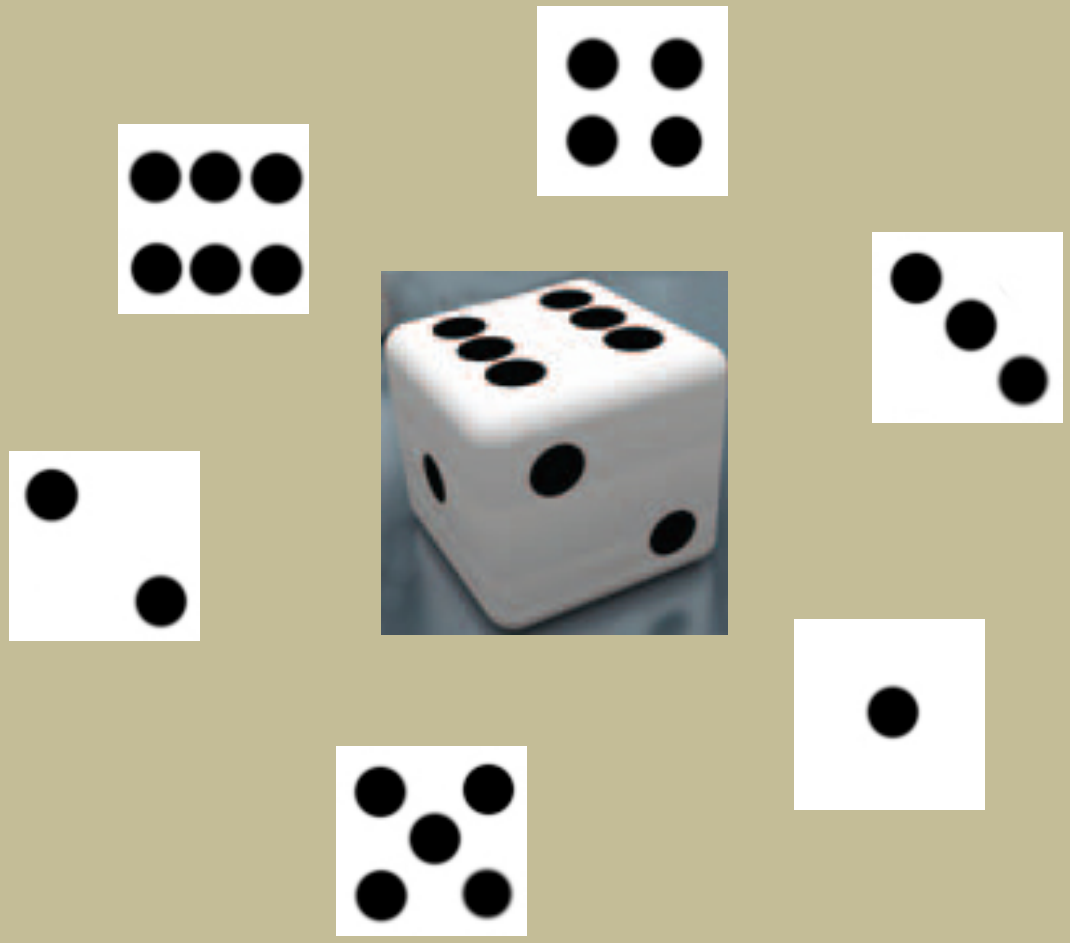
Let's model a pair of dice,

Modeling Using the `rand` Function

one die at a time.

Modeling Using the `rand` Function

What are the numbers on one die?



Modeling Using the `rand` Function

What are the bounds of the range of numbers on one die?

1 and 6 (inclusive)

We want a value randomly between those endpoints (inclusively)

Modeling Using the `rand` Function

We need random integers between {1, 2, 3, 4, 5, 6}

or ….

{0, 1, 2, 3, 4, 5} + 1

We can use the remainder (%) of the division by 6, then add 1

Example

Run *dice.cpp* in Cloud9

Simulations

Back to simulations!

- Simulations are commonly used for
  - Predicting climate change
  - Analyzing traffic
  - Picking stocks
  - Many other applications in science and business

- Oregon Trail Project – ”every turn, … there is a **40% probability** a misfortunate event might occur.”

Probabilities

Oregon Trail Project – ”every turn, … there is a **40% probability** a misfortunate event might occur.”

Hmmm …. 40%

…. 40 per 100

….. 40 out of 100

Probabilities

Oregon Trail Project – ”every turn, … there is a **40% probability** a misfortunate event might occur.”

….. 40 out of 100

What if we toss a 100 sided die?

When will a misfortune occur?

```
// toss a 100 sided die
// if the value is less than or equal to 40
// then a misfortune will occur
```

Example

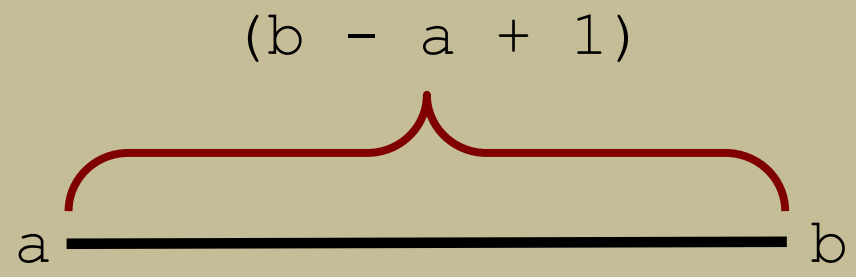
What if we have a game with multiple dice?

- Yahtzee
- Risk
- Dungeons and Dragons

Let's go to cloud9, define a Dice class and go from there …

Modeling Using the `rand` Function

Given two endpoints,
a and **b**, how many values are between **a** and **b**,
(including the bounds themselves)?

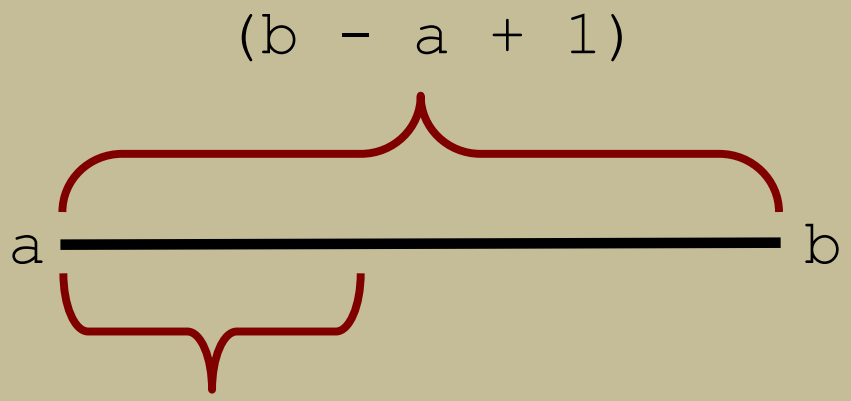


There are

(b - a + 1)

values between **a** and **b**,
(including the bounds themselves).

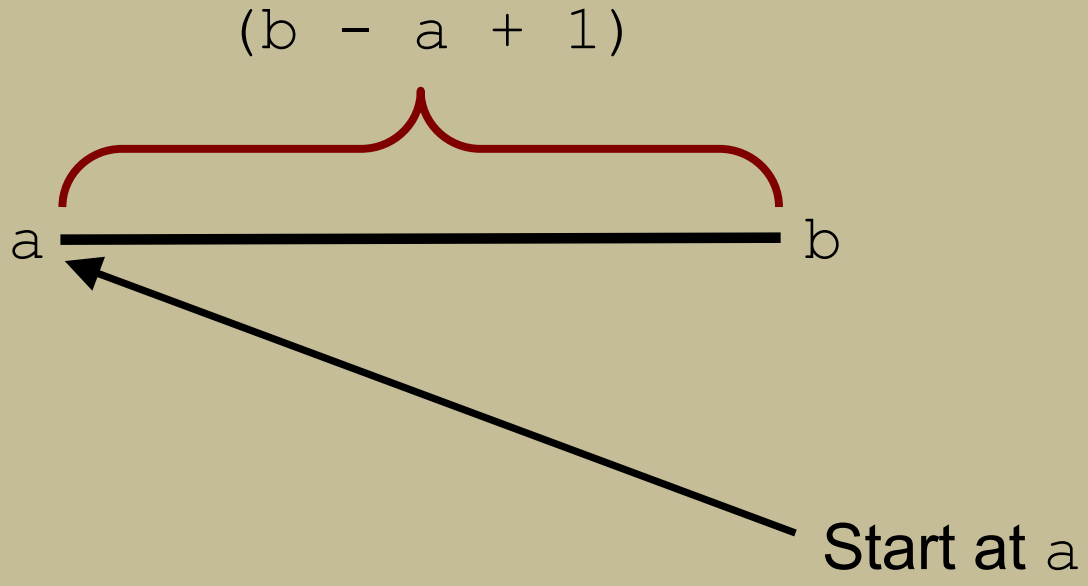
Modeling Using the `rand` Function



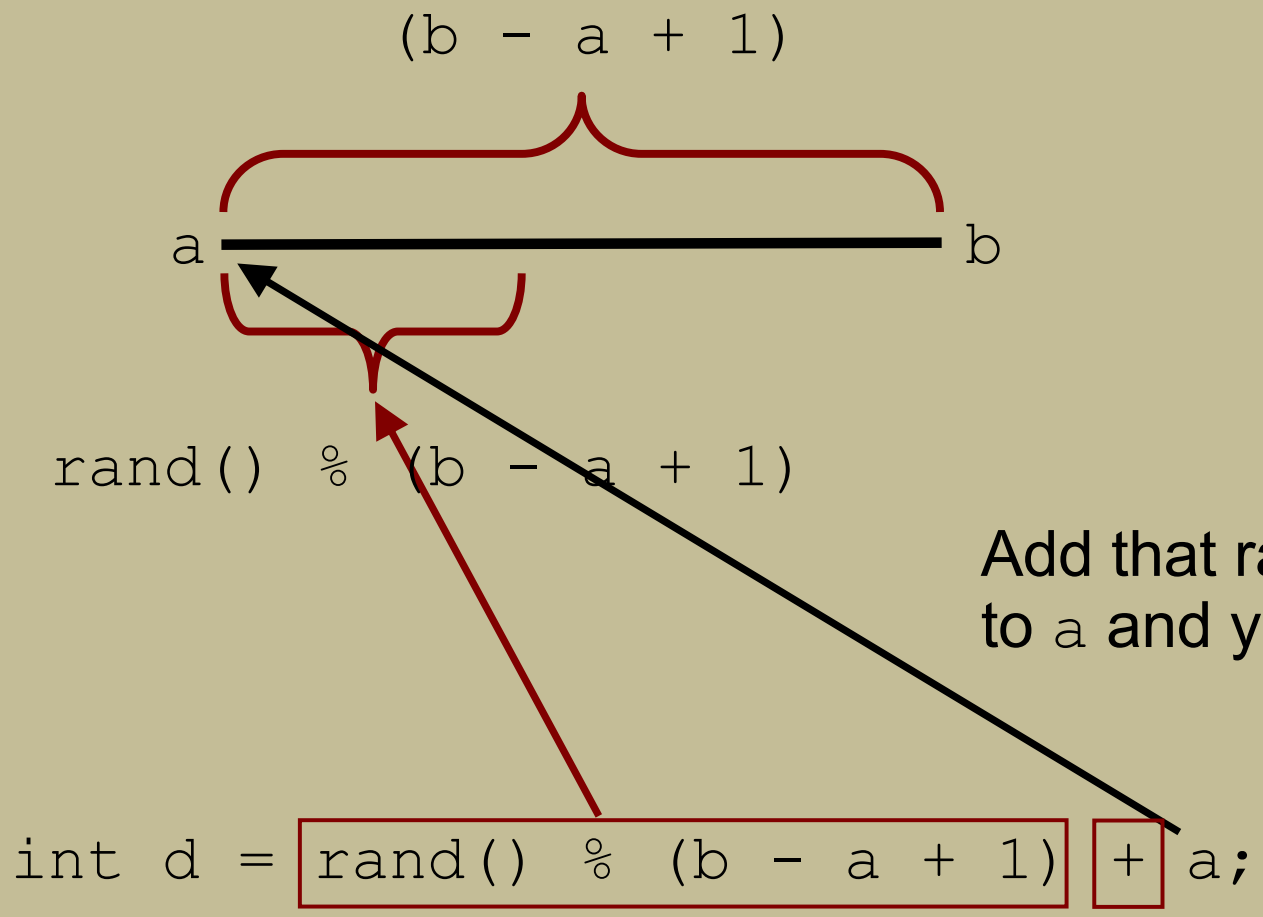

```
rand() % (b - a + 1)
```

Obtain a random value
between `0` and `b - a`
by using the `rand()` function

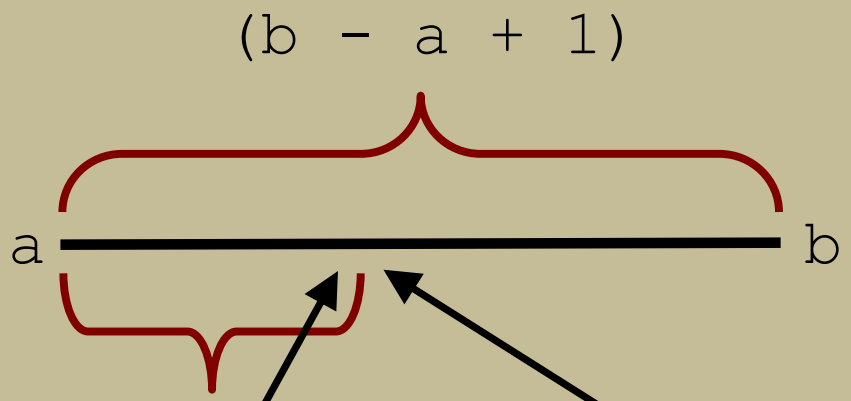
Modeling Using the `rand` Function



Modeling Using the `rand` Function



Modeling Using the `rand` Function



`rand() % (b - a + 1)`

a random value in the range.

```
int d = rand() % (b - a + 1) + a;
```

Agenda

- Today:
  - Simulations, generating random numbers
  - Classes that contain other classes

Example

Dice.h, Dice.cpp, Player.h, Player.cpp, yahz.cpp