PIC 10A 2B

TA: Bumsu Kim

UCLA

Today…

- Random Numbers
  - Applications to Loop Exercises
- Functions
- More Loop Exercises
  - Collatz Conjecture

UCLA

Random Numbers

Syntax, Libraries, and Usage

rand() <cstdlib>

- `rand()` generates a (pseudo-) random integer,
  - Between `0` and `RAND_MAX`
    - RAND_MAX depends on the library
    - But at least 0x7fff = $2^{15} - 1 = 32767$ on any stand library implementation
  - Using a "seed"

- Using the same seed, `rand()` will generate the same sequence of random numbers
  - More on this later

UCLA

Using `rand()` – Floating Point numbers

- Since it generates an `int` between `0` and `RAND_MAX`, you can generate a random number between `0` and `X` (inclusive) by

  - `rand_num = (rand()*1./RAND_MAX)*X;`

    Type Conversion

    Normalized to 0 ~ 1

  - Or,

  - `rand_num = (static_cast<double>(rand())/RAND_MAX)*X;`

    which is preferable

UCLA

Using `rand()` – Floating Point numbers

- If you want a real number between A and B,
  - The size of interval = (B – A)
  - Starts from A

- `rand_num = ((rand()*1./RAND_MAX)*(B-A) + A;`

Between 0 and (B-A)

UCLA

Using `rand()` – Integers

- For integers, we can use the % (mod/remainder) operator

- Random int between 0 and N (inclusive):
  - `n = rand()%(N+1)`
  - The remainder is always between 0 and N
    - Note that, there are **N+1** different integers in [0,N]

- Random int between A and B:

  = A + (random int between 0 and (B-A))
  - `n = A + rand() % (B-A+1)`

UCLA

A Seed for `rand()`

- The function `rand()` generates a sequence of random numbers using a seed
- The seed can be set by `srand(some_number)`;
- Ex) `some_number` == 1 → `rand()` gives
  - 41
  - 18467
  - 6334
  - 26500
  - 19169
  - 15724 …

  - Exercise: Use rand() several times without srand() commands, and verify that you get the sequence above

UCLA

Seed for `rand()`

- With the same seed, you'll have the same sequence of random numbers

- So, to get a random-like numbers (pseudo-random numbers), use different seed every time you run the code

- Commonly used trick: use a ***current time*** as a seed
  - `time(nullptr)` in `<ctime>` library returns the current time in **seconds**, since *00:00, Jan 1 1970 UTC*
  - Ex) `time(nullptr) == 1666067370` when this slide was created

UCLA

Summary

- #include `<cstdlib>` for `rand()` and `srand()`

- #include `<ctime>` for `time()`

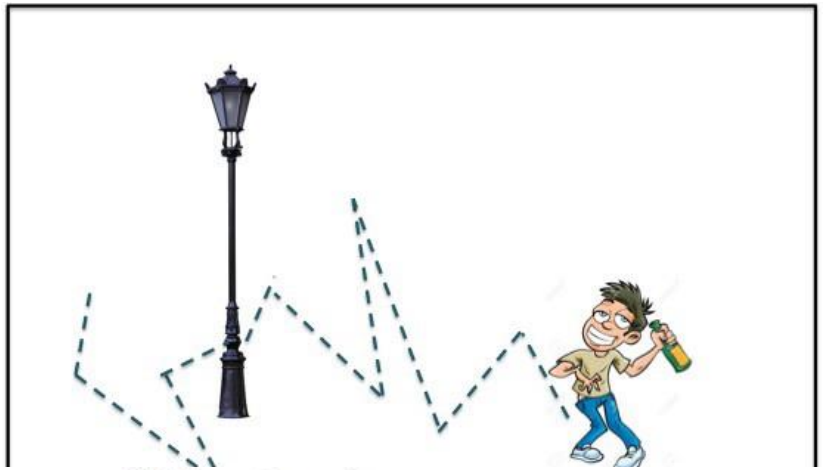
- Use `time(nullptr)` to generate a seed:
  - `srand(time(nullptr));`
  - And then use `rand()`

- Intervals:
  - `r_double = (rand()*1./RAND_MAX)*(B-A) + A;`
  - `r_int = A + rand()%(B-A+1);`

UCLA

Exercise – Random Walk Simulator

- Suppose someone walks in a constant speed, but chooses the direction randomly in each step
- The path (to be more precise, this *stochastic process*) is called a random walk



- We will simulate a 1-D random walk:
  - An object can only move forward or backward at each step
  - When the object hits the wall (upper/lower bounds), it stops walking
  - The time that it hits the wall is called the "stopping time"
  - Important in mathematical finance

Functions

Basic Syntax and Terminology

Functions

- Functions can be
  - Declared and then defined later
    - If you declare the function first (not define it right now) then need ;
  - Defined when it is declared (so both are done at the same time)

- The declaration determines a “signature” of the function
  - [Return Type] [Function Name] [Input Parameters] (and other options later)

```
double find_max(double a, double b);
```

UCLA

Functions

- The declaration determines a “signature” of the function
  - [Return Type] [Function Name] [Input Parameters] (and other options later)

```
double find_max(double a, double b);
```

- Return Type
  - The type of the expression returned by the function
  - If the function returns nothing, can be `void`
- Function Name
  - Name of the function; cannot be a reserved word, the same naming rule applies with variables
- Input Parameters
  - The objects passed to the function
  - Number of arguments can be 0, 1, or more
  - Still need parentheses () even when the function gets zero parameters

UCLA

Functions – Examples

- A function returning the maximum of two numbers

```
double find_max(double a, double b) {
        return (a > b) ? a : b;
}
```

Recall: The ternary operator

```
A ? B : C
```

is (almost) equivalent to

```
if (A) { B; }
else {C;}
```

- A ***procedure*** is a function returning `void`
  - Which means, it doesn't return anything
  - Example: a function printing the max to the console

```
void print_max(double a, double b) {
    cout << "The max of " << a << " and " << b
         << " is " << find_max(a, b);
}
```

Possible if `find_max` is ***declared*** before this expression

Q: [`true` or `false`?]
Using "`return`" keyword in a ***procedure*** results in a syntax error.

UCLA

Functions – Examples

- A function that inputs several different types of arguments

```
void func(unsigned int i, string str) {
        cout << str[i];
}
```

Here `i` cannot be negative

- A ***predicate*** function is a function returning `bool`
  - The following function checks whether the first letter of the string is capitalized or not

```
bool isCapitalized(string str) {
    if ('A' <= str[0] && str[0] <= 'Z') {
        return true;
    } else {
        return false;
    }
}
```

True if `str[0]` is between ‘A’ and ‘Z’ (otherwise it is not a capital letter)

UCLA

More Loop Exercises

Collatz Conjecture

Exercise – Collatz Conjecture

- Given a positive integer, perform the following operation each step:
  - If the number is even, divide it by two $x \leftarrow x/2$
  - If the number is odd, triple it and add one $x \leftarrow 3x + 1$
- The Collatz conjecture states that you will always reach 1.
- Write a function that computes how many steps it takes to reach 1 starting from a positive integer. The input and output should be:

```
Starting from X.
Step 1: X
Step 2: X
...
Step X: X
Starting from X, it took X steps.
```

Output example starting from 6

```
Starting from 6
Step 1: 3
Step 2: 10
Step 3: 5
Step 4: 16
Step 5: 8
Step 6: 4
Step 7: 2
Step 8: 1
Starting from 6, it took 8 Steps.
```

- Hints
  - Again, the `while` loop will be useful
  - This function is a ***procedure***, i.e., it will return `void`

UCLA

Your Feedback is welcome

- Don’t hesitate to give a feedback on the discussion
- Use the link on my Github repo, or the link below:
  - https://forms.gle/erZj1iSgHNrHQuXk6

My Github repo on the web looks like:

code	Week2 Tu
LICENSE	Initial commit
README.md	Update README.md

README.md

PIC10A

PIC10A discussion 2B, UCLA for Fall 2022

Google form link for feedbacks: https://forms.gle/erZj1iSgHNrHQuXk6

Click this link

UCLA