How time and randomness work in programming, and how to use the math library

CS 10A – MATH, TIME, AND RNG

Allen Zhao – SRJC CS 10A

External Libraries – Quick Review

- <iostream>, <iomanip>, and <string> are all C++ libraries, also sometimes known as header files.
- Libraries are groups of functions, constants, and sometimes custom data types designed for a specific purpose, and are brought in via the #include command in your code.
- A library is a separate code file that exists elsewhere on your computer. The computer must know exactly where it is in order to use it. If your compiler is complaining about an unrecognized library name, then either you made a typo, the compiler needs reinstalling, or you can manually locate it.
- Modern IDEs and compliers all include the most commonly used libraries in their base installation, allowing us to use them with just the include statement.

The Math Library

- The math library, <cmath>, is exactly what it sounds like. It's a convenient library that makes a ton of different math operations available for use in C++.
- The math library is actually really big, so the next slide will only cover the common stuff that you'll likely use from the math library. Other functions can be found online.
- By default, every function in the math library uses double for input and output, but you can use of long double or float instead if you wish and maintain that accuracy.

Common Math Functions and Constants

- sin(x) – sine function, where x is in radians
- asin(x) – arc sine function, reports back in radians
- exp(x) – natural exponential function, does e^x
- log(x) – natural logarithm function, so log base e
- log10(x) – common logarithm function, so log base 10
- pow(x, y) – standard exponential function, does x^y
- sqrt(x) – square root function
- ceil(x) – rounds x up to the next closest integer
- floor(x) – rounds x down to the next closest integer
- abs(x) – get the absolute value of x
- NAN – reserved constant name representing not-a-number, usually infinite
- M_PI – reserved constant name representing pi

Math Library Functions - Examples

```
#include <cmath>
double x = 1, y;
int main()
{
        y = cos(x*M_PI/180.0);    // cosine via degrees
        y = pow(x, 1.0/3);  // cube root, remember int division rules
        y = log(0);    // printing this out will either get you inf or nan
        y = exp(1);   // do this to get the constant e by itself

        return 0;
}
```

How Computers Keep Time

- Time keeping is critical for synchronizing computer systems and keeping critical calculations accurate.
- For the common modern day computer, time keeping is done in seconds. Local time keeping accuracy is dependent on the CPU.
- For offline hardware, time is usually tracked as how many milliseconds or seconds have passed since the hardware turned on. Computers often use back up batteries to keep the clock ticking when the computer is disconnected from power.
- A single signed integer in computer memory keeps track of time by counting the number of seconds that have passed since the Epoch (defined as midnight on Jan. 1, 1970 UTC). This is the timekeeping standard defined by UNIX systems, so it’s often known as UNIX time.

Y2K and the Year 2038 Bug

- Back in the day, when memory was more limited, time was tracked in a more fragmented fashion. In particular, the year was just represented as a two digit number.
- Once 99 rolled over back to 00, major systems everywhere at the time would read that as 1900 rather than 2000. System crashes for critical infrastructure were definitely possible, but likely not world-ending like the media made it out to be at the time.
- Any potential disaster was averted because most computer systems were updated to change their time keeping system to our current model before the dawn of the new millennium.
- Due to the limits of the signed int, the next Y2K-like event is on 1/19/2038, 3:14:08 AM UTC

The Time Library

- The time library, <ctime>, allows us to track time in our code. As such, it's also a handy way to also track resource consumption and program performance.
- Alternatively, you can use the time library any time a calendar is needed. The time library contains additional data types and functions that converts between UNIX time and the Gregorian calendar format we use.
- The time library is also a highly recommended component in random number generation.

Some Time Library Functions and Constants

- clock() – returns how much processor time was used by the program, a -1 is returned if the function fails. Units are in clock ticks, an arbitrary unit whose definition can vary. You can store this value in an integer.
- CLOCKS_PER_SEC – reserved integer constant that stores the number of clock ticks per second
- time(&x) – gets the current UNIX time, a -1 is returned if the function fails. You can store this value in an integer. For convenience, you can just put NULL or 0 as the input.

Time Functions - Examples

```
#include <ctime>
double runtime;          // How long it took for the program to finish in seconds
int main()
{
        cout << time(NULL) << endl;    // Prints out current UNIX time
        // … some program commands

        runtime = 1.0*clock()/CLOCKS_PER_SEC;     // int to double conversion
        cout << “Program runtime: ” << runtime << endl;

        return 0;
}
```

Randomness in Computer Science

- Randomness has enormous applications in computer science, but because computers are designed to work as a fully predictable system, creating true randomness in any computer is impossible.
- All STEM fields deal with statistical analysis, so true randomness is ideal when testing any kind of theory or model.
- More importantly, true randomness would allow for much stronger cybersecurity (encryption).
- To get around this, computers fake their randomness, also known as being pseudorandom. It’s actually still predictable.

Random Number Generation

- Random Number Generation (RNG) starts by creating a seed. This seed serves as a base calculation point from which all pseudorandom numbers are created.
- In reality, RNG is just a simple sequence of calculations. If your seed is the same every time you start up a random number generator, you'll get the exact same sequence.
- To ensure that a unique seed is created every time a program runs, we use the current time stamp to serve as the seed. After all, it's impossible to run a program with the same timestamp more than once (on 1 PC).
- Random functions can be found in the C Standard Library <cstdlib>. Make sure to include this if you need RNG.

Setting Up RNG in C++

```
#include <cstdlib>    // C Standard General Utilities Library
#include <ctime>      // Time library
int main()
{
        int seed = time(NULL);        // Set seed to current UNIX time
        srand(seed);  // Seed the random function

        cout << rand() << endl;       // rand() is the RNG
        // You can also reseed rand() later in the program (not recommended)

        return 0;
}
```

Range Control for rand()

```
/* By itself, rand() returns a random number between 0 and RAND_MAX, which is a constant defined in <cstdlib>. Its value is subject to variance, but it's usually at least the max of a short int, or more commonly an int. */
int rand_num;
int max_rand = 100;     // Choose your maximum
int min_rand = 50;      // Choose your minimum
int main()
{
        rand_num = rand() % (max_rand + 1 – min_rand) + min_rand;
        // The range is now set to [50, 100]

        return 0;
}
```