Slides #7
Random,
AND / OR

CMPT 130

© Dr. B. Fraser

Part 1: Random

1) Picking a random number

2) How can we code complex conditions (Part 2): "Grass is wet if it rained or the sprinkler was on"

'Random' numbers

- Computers are not Random
  - But we would like random numbers!

- Use rand() to return a..
  between 0 and RAND_MAX (32767)
  - #include <cstdlib>

  - int a = rand();
    int b = rand();
    int c = rand();

  RAND_MAX
  must be at least
  ~32K, but can be
  bigger.

- However:
  - Each time the program is run,
    a, b and c's values..

Seed

- The pseudorandom sequence is based on a seed
  - use srand() to seed the sequence once. srand(42);
  - Based on a certain seed, the program..

- Randomize by the timer
  - Computers have clocks.
  - Get what *seems* a random seed by using the timer: srand(time(nullptr));   // must #include <ctime>

time(nullptr)

- time() Function
  - Returns..
  - It takes one argument, a *pointer but can just* pass a null pointer for simple use (what we need).

- Example
  cout << "Seconds since Jan 1 1970: " << time(nullptr);

- srand() needs a seed number, so we can give it the number of seconds since Jan 1 1970!
  int numSec = time(nullptr);
  srand(numSec);

- Only call srand() once (usually)
  - Calling it again resets the pseudorandom sequence (which can be useful sometimes!).

Dice rolling

```cpp
#include <iostream>
#include <iomanip>
#include' <cstdlib>     // NEEDED for rand() and srand()
#include <ctime>        // NEEDED for time()
using namespace std;
```

```cpp
// Experiment with rand
const int NUM_ROLLS = 15;
const int MAX_VAL = 20;

int main()
{
    // Pick a random seed based on the timer
    int numSec = time(nullptr);
    srand(numSec);

    // Do a bunch of D20 rolls (1 to 20):
    int i = 0;
    while (i < NUM_ROLLS) {
        cout << "Rolling: " << setw(2)
                << (rand() % MAX_VAL + 1) << endl;
        i++;
    }
}
```

```
Rolling: 7
Rolling: 5
Rolling: 15
Rolling: 10
Rolling: 13
Rolling: 13
Rolling: 18
Rolling: 1
Rolling: 4
Rolling: 20
```

```cpp
// Explanation of math:
int randValue = rand();       // Between 0 and RAND_MAX (>32,000)
randValue %= MAX_VAL;  // Between 0 and 19
randValue += 1;               // Between 1 and 20
```

C++ Standard

- To use nullptr, must set C++ Standard to C++11:
  - The "standard" is revised from time-to-time.
  - The latest standard is C++20 (2020)

- In VS Code
  - Automatically compiles when using nullptr
  - But, be careful, different development environments have different default settings

Pseudo Random Hiking (analogy)

- Imagine hiking on a path with numbers written on signs:
  - Each sign you come to is a new pseudo-random number
    - In C++: it's like calling..
  - Each time you go on *that* hike, you get the..

- Imagine there being many different paths:
  - Each path has these numbered signs.
  - Which path you choose dictates the..
                                    you see.
    - In C++: calling..   picks the path
- If you restart the hike, you get the..
    - In C++: calling srand()..

Modern C++ Random (C++11)

```cpp
// Better Random Number Generator
#include <iostream>
#include <random>
using namespace std;

int main()
{
    // Create the random number generator
    // 1. Get random seed (different each time)
    std::random_device rd;
    // 2. Seed the random number generator
    default_random_engine engine(rd());
    // 3. Define distribution (uniform); range [1, 100]
    std::uniform_int_distribution<> distr(1, 100);

    // Generate N numbers
    for(int i = 0; i < 20; i++) {
        // Generate a random number
        std::cout << distr(engine) << ' ';
    }
    cout << endl;
    return 0;
}
```

rand() / srand() have many drawbacks.

Better to use the more complicated, but safer "modern" C++11 approach

1) Picking a random number (Part 1)

2) How can we code complex conditions:
"Grass is wet if it rained or the sprinkle was on"

Three logician just finished dinner.
The waiter asks, "do you **all** want dessert?"
The first logician says, "I don't know."
The second also says, "I don't know."
The last says, "Yes, we would."

Logical Operators

- Logical Operators work on Boolean values:
  - And..   (true && true) == true
  - Or..   (true || false) == true
  - Not..   !true == false, !false == true

- Example:

```cpp
int main()
{
    bool haveRainwater = ...;
    bool isWarm = ...;

    if (!isWarm) {
        cout << "Turn on heater overnight\n";
    }

    if (haveRainwater && isWarm) {
        cout << "Plant new seeds\n";
    }
}
```

```cpp
int main()
{
    bool isRetired = ... ;
    bool isUnemployed = ... ;
    int month = ... ;
    int day = ... ;

    // On the last day of the year, ...
    if ( (month == 12) && (day == 31) ) {...}

    // If either (or both) retired or unemployed then...
    if (isRetired || isUnemployed) {...}

    // If not retired or it's Jan 1st then ...
    if (!isRetired || ( month == 1 && day == 1) ) {...}

    // If not retired or it's Jan 1st then ...
    bool isJan1st = (month == 1 && day == 1);
    if (!isRetired || isJan1st) { ... }
}
```
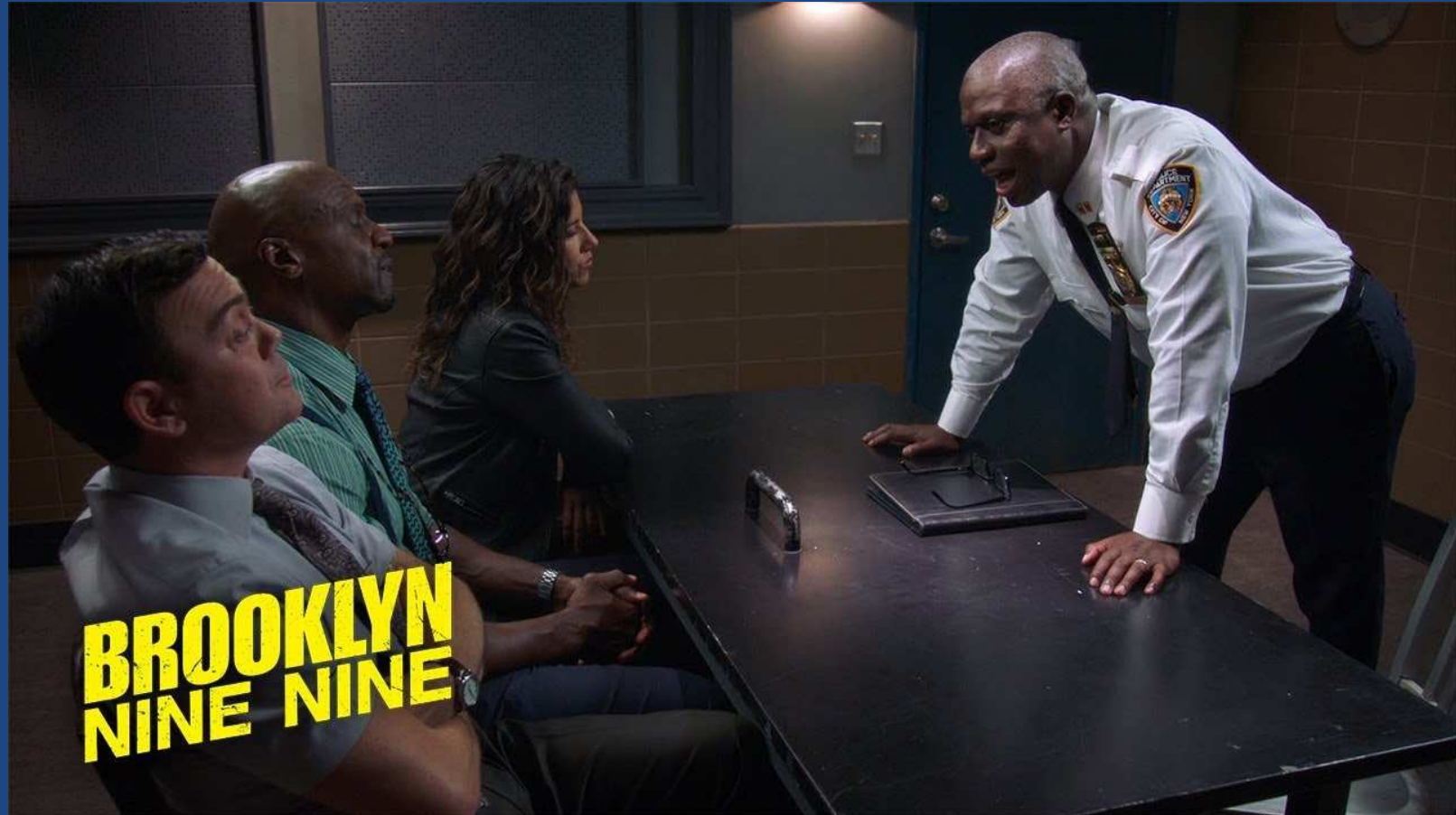
Truth Tables Reprise

- A truth table

  ..

  – One column for each of the logical variables or comparisons – i.e. Boolean values
  – One row for each possible combination of values
    - That is for each combination of true and false

A	B	A && B	A || B	!(A || B)	!A || B
T	T	T	T	F	T
T	F	F	T	F	F
F	T	F	T	F	T
F	F	F	F	T	T

Truth Table Example

- If(!(x > 7) && y <= 10))
  - Is this true for certain values?

x	y	x > 7	y <= 10	!(x > 7)	!(x > 7) && y <= 10
14	8	T	T	F	F
9	16	T	F	F	F
7	3	F	T	T	T
1	10	F	F	T	F

Precedence Revisited

- Examples:

```
int        x = 5;
int        y = 4;
bool       isDone = false;
```

isDone = x < y + 1;

isDone = x < y == 4;

isDone = x + 1 || y;

isDone = x < y || ! y >= x;

..

isDone = 1 < x < 4;

Prec. Level	Op.	Operation	Associates
1	+ - !	unary plus/minus not	R to L
2	* / %	mult, div, remainder	L to R
3	+ -	add subtract	L to R
4	< > <= >=	comparisons	L to R
5	== !=	equal, not equal	L to R
6	&&	AND	L to R
7	||	OR	L to R
8	= += -= *= ...	assignments	R to L

Order can be forced by parentheses.
See text for full list.

Quick test with Boolean

- Quick test for true:
  ```
  cout << "Enter your favourite number: ";
  int favNum = 0;
  cin >> favNum;
  bool greatNum = (favNum == 42);
  if (                    ) {
      cout << "Awesome choice!";
  }
  ```

  > if greatNum is true...

- The following are identical (for bool!):
  - if (greatNum) {...}
    if (greatNum == 1) {...}   if (greatNum == true) {..}
    if (greatNum != 0) {...}   if (greatNum != false) {..}

Explanatory Variables

- Explanatory variables simplify complex expressions:
  ..

```cpp
// Option 1: One expression
if ((height >= MIN_HEIGHT) && (age >= 18) && (age <= 65)) {
    cout << "Please pay adult fare.\n";
}

// Option 2: Two explanatory variables.
bool isTallEnough = (height >= MIN_HEIGHT);
bool isAdult = (age >= 18) && (age <= 65);
if (isTallEnough && isAdult) {                    ...
    cout << "Please pay adult fare.\n";
}
```

Review

- What is printed?
  cout << ((1 != 50) && (1 < 10 < 3)) << endl;

Note:
    cout << true;   // prints '1';
    cout << false; // prints '0'.

Summary

- Random uses:
rand(), srand(), and time()

- Logical expressions: &&, ||, !

- Suggested rand() review:
Write a program which
  - Picks a random number between 1 and 100.
Use named constants for the 1 and 100 in this case.
  - Print out if the number is odd
  - Print out if the number is between 40 and 60 inclusive