Slides #7
Random, AND / OR
CMPT 130
© Dr. B. Fraser
Topics

1) Picking a random number
2) How can we code complex conditions:
   “Grass is wet if it rained or the sprinkle was on”
Random Numbers
'Random' numbers

● Computers are not Random
  – But we would like random numbers!

● Use rand() to return a number between 0 and RAND_MAX (32767)
  – #include <cstdlib>
    – int a = rand();
      int b = rand();
      int c = rand();

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

RAND_MAX must be at least ~32K, but can be bigger.
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!).
#include <iostream>
#include <iomanip>
#include <cstdlib>     // NEEDED for rand() and srand()
#include <ctime>       // NEEDED for time()
using namespace std;

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++;
    }

    // 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
}

Rolling: 7
Rolling: 5
Rolling: 15
Rolling: 10
Rolling: 13
Rolling: 13
Rolling: 18
Rolling: 1
Rolling: 4
Rolling: 20
C++ Standard

- To use nullptr, must set C++ Standard to C++11:
  - The “standard” is revised from time-to-time.

- In CLion
  - Automatically adds -std=c++11 to the build flags in CmakeList.txt
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()..
AND & OR
Logical Operators

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

• Examples:

```c
int month = ... ;    bool isRetired = ... ;
int day = ... ;      bool isUnemployed = ... ;

// 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) ) {...}
```
# Precedence

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+ -</td>
<td>unary plus/minus</td>
<td>R to L</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>not</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>* /</td>
<td>mult, div, remainder</td>
<td>L to R</td>
</tr>
<tr>
<td></td>
<td>%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>+ -</td>
<td>add subtract</td>
<td>L to R</td>
</tr>
<tr>
<td></td>
<td>&lt;= &gt;=</td>
<td>comparisons</td>
<td>L to R</td>
</tr>
<tr>
<td>4</td>
<td>== !=</td>
<td>equal, not equal</td>
<td>L to R</td>
</tr>
<tr>
<td>5</td>
<td>&amp; &amp;</td>
<td>AND</td>
<td>L to R</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>= +=</td>
<td>assignments</td>
<td>R to L</td>
</tr>
<tr>
<td>8</td>
<td>-= *=</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See back cover of text for full list.

Order can be forced by parentheses.

See back cover of text for full list.
Quick test with Boolean

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

- 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'.

In class Example: Roll dice

• Create a program which rolls many dice:
  – Ask user for number dice (enforce > 0)
  – Show roll of each die and show sum
  – Randomize rolls between runs

• Use functions to break up tasks

• Additional Features: Read # faces on die (>1, <=100)

Enter # sides on dice: 6
Enter number dice to roll: 4
Rolls: 3, 5, 3, 5,
Total is: 16.

Enter # sides on dice: -1
Invalid dice size. Must be between 1 and 100.
Re-enter size: 100
Enter number dice to roll: -5
Invalid number of dice. Must be 1 or more.
Enter number dice to roll: 10
Rolls: 69, 22, 74, 29, 56, 50, 29, 54, 86, 3,
Total is: 472.
In class Example: Square Root

- Data type double works with decimal places.
- Create a program which calculates a square root
  - Ask the user to enter a number \( n \) (double)
  - Compute its square root using Babylonian algorithm:
    1. set guess = \( n/2 \) (initial guess)
    2. compute \( r = n / \text{guess} \)
    3. set guess = \((\text{guess} + r) / 2\) (average)
    4. goto step 2 until close enough
      - Get answer within 1% of previous guess
  - Use functions, constants, and good programming
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