The <random> library

C++ code has a modern library <random> that improves over the old C and C++ use of rand() found in <cstdlib>.

In the program that we saw in week 1 for calculating the probabilities when rolling two dice:

```
//The following program computes
//the probability for dice possibilities
#include  <iostream>   //drops .h still available
#include  <cstdlib>
#include  <ctime>
using namespace std;
const int sides = 6; //replaces many sharp defines
inline int r_sides(){ return (rand() % sides + 1); }
int main(void)
{
   const int   n_dice = 2;
   srand(clock());   //why?
   cout << "\nEnter number of trials: ";
   int trials;
   cin >> trials;  //compare to scanf
   int* outcomes = new int[n_dice * sides +1];

  for (int j = 0; j < trials; ++j) {
  int roll = 0;
  for (int k = 1; k <= n_dice; ++k) {
  roll += r_sides();
  }
  outcomes[roll]++;
  }

  cout << "probability\n";
  for (int j = 2; j < n_dice * sides + 1; ++j)
      cout << "j = " << j << " p = "
      << static_cast<double>(outcomes[j])/trials
      << endl;
}
```

This is replaced by

```
//The following program computes
//the probability for dice possibilities
#include  <iostream>
#include  <random>
#include  <ctime>
using namespace std;
const int sides = 6;
int main(void)
{
   const int   n_dice = 2;
   uniform_int_distribution<unsigned> u(1,6);
   default_random_engine e(time(0));

   cout << "\nEnter number of trials: ";
   int trials;
   cin >> trials;  //compare to scanf
   int* outcomes = new int[n_dice * sides +1];

  for (int j = 0; j < trials; ++j) {
  int roll = 0;
  for (int k = 1; k <= n_dice; ++k) {
  roll += u(e);
  }
  outcomes[roll]++;
  }

  cout << "probability\n";
  for (int j = 2; j < n_dice * sides + 1; ++j)
      cout << "j = " << j << " p = "
      << static_cast<double>(outcomes[j])/trials
      << endl;
}
```

The library <random> has modern random number generator including a default random engine declaration as seen in this example. Popular engines included are

Pseudo-random number engines (instantiations)

Particular instantiations of generator engines and adaptors:	
default_random_engine	
	Default random engine (class)
mt19937	
	Mersenne Twister 19937 generator (class)
ranlux48	
	Ranlux 48 generator (class)
knuth_b	
	Knuth-B generator (class)

Ira Pohl March 23, 2018