CISC 1110 (CIS 1.5)
Introduction to Programming Using C++

Spring 2012
Instructor : K. Auyeung

Email Address: kenny@sci.brooklyn.cuny.edu
Course Page: http://www.sci.brooklyn.cuny.edu/~kenny/cisc1110

Class Hours: MW 10:35 – 12:40PM 4428N

Agenda

- Random Numbers
- Control Structures
- `for` loop
- `break` statement
- Operation Precedence
- Submitting work via class website

RADOM NUMBERS

computers can generate ``random'' numbers, which is like picking a number by rolling dice

- there are two steps necessary for generating random numbers:1. initializing the random number generator 2. picking the random number

- the random numbers generated are integers (of type)

- The srand() function (which is part of the C++ language) is used to initialize the random number generator

- The rand() function (which is also part of the C++ language) is used to return a random number

- rand() returns a random integer between and RAND_MAX, where RAND_MAX is a constant (defined as part of the C++ language) that represents the biggest integer allowed in the system (in our case, 2^31 – 1) using the modulo operator (%) you can scale the result returned from rand() to get a number in the range you want (e.g., 0...10)

- note that you may need to have the following two #include statements at the top of your program file, in order for the compiler to recognize srand() and rand():

#include <time.h>
#include <stdlib.h>

these are IN ADDITION to the #include <iostream> that you normally put, so the beginning of your program file will look like this:

```
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;
```

RANDOM NUMBERS: COMPLETE EXAMPLE

```
#include <iostream>
using namespace std;
#include <time.h>
#include <stdlib.h>

int main() {
 //initialize random number generator using the current time as the seed
  srand( time( NULL ));

  // declare variables
  int x, y, z;

  // pick a random number
  x = rand();

  // scale the value of x to an integer between 0 and 10
  y = x % 11;

  // or do the above two steps in one,
  // essentially picking a random number between 0 and 10
  z = rand() % 11;

  return 0;
} // end of main()
```

Control Structures

- Allows programs to bifurcate, repeat code or make decisions
- Compound-statement, sometimes called a block, is a group of statements separated by '`;`' and bounded by braces '`{}`'
- The `for` Loop
- Repeats a block of code a predetermined number of times

```
for (initialization; condition; increase) statement;
```

Sample

```
#include <iostream>

using namespace std;

int main () {

        for (int n=10; n>0; n--)    {
                cout << n << ", ";
        }

        cout << "STOP!\n";

        return 0;
}
```

Variation

- The initialization and increase fields are optional.

```
int count = 3;

for (; count<=6;count++)
  cout << endl << " count " << count;

count = 3;
cout << endl << endl;

for (; count<=6;)  {
  cout << endl << " count " << count;
  count++;
}
```

- The semicolon must be written even if a field is omitted

Break Statement

- Terminates a loop prior to its predefined termination condition

```
// break loop example

#include <iostream>
using namespace std;

 int main () {

  for (int n=0; n<10; n++)  {
    cout << n << ", ";
    if (n==3)  {
      cout << "countdown aborted!";
      break;
    }
  }

return 0;
}
```

Operation Precedence

1	() [] -> . ::	Grouping, scope, array/member access
2	! ~ - + * & sizeof *type cast* ++x --x	(most) unary operations, sizeof and type casts
3	* / %	Multiplication, division, modulo
4	+ -	Addition and subtraction
5	<< >>	Bitwise shift left and right
6	< <= > >=	Comparisons: less-than, ...
7	== !=	Comparisons: equal and not equal
8	&	Bitwise AND
9	^	Bitwise exclusive OR
10	|	Bitwise inclusive (normal) OR
11	&&	Logical AND
12	||	Logical OR
13	?:	Conditional expression (ternary operator)
14	= += -= *= /= %= &= |= ^= <<= >>=	Assignment operators
15	,	Comma operator

www.wikipedia.com