CMPSC 16
Problem Solving with Computers I
Spring 2014

Instructor: Lucas Bang

Lecture 7: One More Loop Problem,
Generating “random” values, Midterm Review
One More Loop Example
Loops, final example

- A projectile launched with initial velocity $v_0$ at an angle $\alpha$ will travel a horizontal distance $d$ given by

$$d = \frac{v_0^2}{g} \sin(2\alpha)$$

- Find the angle that maximizes the distance.
Loops, final example

d_max = 0;
alpha_max = 0;
for(alpha = 0; alpha <= 90; alpha += 5) {
    alpha_radians = alpha*PI/180;
    d = v0*v0*sin(2*alpha_radians)/G;
    if(d > d_max){
        d_max = d;
        alpha_max = alpha;
    }
}
}
Generating Random Numbers
Generating random values

• Some programs require generation of “random” numbers

• A “pseudorandom” number generator is an algorithm that generates a sequence of numbers within a range
  – For example: integers between 1 and 10

• A “pseudorandom” generator actually creates the numbers in a deterministic way, but they appear to be random
  – After a while a pseudorandom generator would repeat the sequence
Generating random values

- The numbers generated by a “pseudorandom” generator approximate properties of random numbers
  - For example: values are uniformly distributed within the range

- Pseudorandom number generator requires a “seed” value that determines the values that will be generated
  - Giving different seed values generates different sequences
  - If you give the same seed, you will get the same sequence
Generating random values in C

• C’s stdlib includes a pseudorandom number generator
  – Include <stdlib.h> for rand and srand functions

• Use rand() to get pseudorandom value from 0 to RAND_MAX (defined in <stdlib.h> too)
  – But always same sequence unless change seed value
  – Pass an unsigned int seed to srand function: srand(seed)

• How to get a different seed every different time? Use time!
  – C has a library: <time.h> with a time function
  – Using srand( time(NULL) ) will provide a different seed based on time

• Can use % operator to restrict range. E.g., see ...
demos/rolldice.c
Midterm Review and Practice
What did we learn so far?

• We covered the first 3 chapters of the book:
  – General concepts about computers
  – Basic concepts about C programs: basic types, assignment, input-output, etc.
  – C control structures, stepwise refinement

• Let’s do a review and some practice problems
Chapter 1: General concepts about computers

• Basic architecture
• Basic concepts about computers:
  – what is: hardware, software, operating system, compiler, assembly language, programming language, source code, object code, software life cycle, …
Practice problem

• Match the following words and definitions

Compiler, ALU, operating system kernel, processor, machine language

1. A software component that manages the interface between the hardware and the software applications
2. The part of a computer that performs mathematical computations
3. A program that translates programs written in high level language to a lower level (or machine) language
Chapter 2: Basic C concepts

• Basic C concepts:
  – representation of numeric values, different base representations
  – representation of characters, ASCII code
  – basic types, type-casting
  – variable declarations, identifiers
  – overflow, underflow
  – evaluation of numeric expressions (precedence, associativity)
  – assignment statement
  – input/output with printf and scanf
Practice problem

• Is there a syntax error in the following declarations, and what is it?

  • `int x_5; /* OK */`
  • `int _y; /* OK */`
  • `int 2y; /* syntax error */`
  • `int y2; /* OK */`
  • `float /* something */ z, y = 4; /* OK */`
  • `float int1; /* OK */`
Practice problem

What will be printed?

```c
int x = 25; float y; float z = 3.0;
printf("%x", x);
prints: 19
printf("%o", x);
prints: 31
printf("%d", x);
prints: 25
y = x / 3;
printf("%+10.3f", y);
prints: +8.000
y = x / z;
printf("%+-10.3f", y);
prints: +8.333
```
Practice problem

What will be printed?

```c
int x;
x = 12;
printf("%d", x);
prints: 12

x = 012;
printf("%d", x);
prints: 10

x = 0x12;
printf("%d", x);
prints: 18
```
Chapter 3: Structured programming, control flow

• Relational operators, relational expressions, logical operators, logical expressions

• Control flow structures in C
  – if, if/else, while, for, do/while
  – Use of { and } to group statements into a block

• Stepwise refinement
  – Top-down approach for solving a programming problem
Practice problem

Determine if the given expressions evaluate to true or false

```c
float a = 5.5; float b = 1.5; float k = -3;
• a < 10.0 + k       /* true */
• a + b >= 6.5        /* true */
• k != a - b          /* true */
• b - k > a           /* false */
• ! ( a == 3 * b)     /* true */
• -k <= k + 6;        /* true */
• a < 10 && a > 5     /* true */
• fabs(k) > 3 || k < b - a   /* false */
```
Practice problem

Fully parenthesize the expressions based on the evaluation order

• a < 10.0 + k
• a + b >= 6.5
• k != a - b
• b - k > a
• ! (a == 3 * b)
• -k <= k + 6;
• a < 10 && a > 5
• fabs(k) > 3 || k < b - a
Practice problem

Parenthesized version

• \( a < (10.0 + k) \)
• \((a + b) \geq 6.5\)
• \(k \neq (a - b)\)
• \((b - k) > a\)
• \(! (a == (3 * b))\)
• \((-k) <= (k + 6)\);
• \((a < 10) \&\& (a > 5)\)
• \((\text{fabs}(k) > 3) \mid \mid (k < (b - a))\)
Practice problem

```c
int x = 2, y = 10, z = 5;
if (x < y / z)
    x = 1;                  /* program point 1 */
else
    x = 0;                  /* program point 2 */
if (z < y - x)
    if (y > 2*x)
        y = 0;              /* program point 3 */
    else if (y < 20)
        y = 1;              /* program point 4 */
    else if (y < 20)
        y = 1;              /* program point 4 */
z = (x + 1) / (y + 2) + 7;
```
Practice problem

• Determine the number of times the following for loops are executed

```java
for (k = 3; k <= 10; k++) { ... } /* 8 times */
for (k = 3; k <= 10; ++k) { ... } /* 8 times */
for (k = -4; k <= 14; k += 2) { ... } /* 10 times */
```
Practice problems

• What is the value of count after the nested for loops are executed?

```java
int k, j, count = 0;

for (k = -1; k <= 3; k++)
    for (j = 3; j >= 1; j--)
        count++;
```

• Outer loop executes 5 times
• Inner loops executes 3 times for each execution of the outer loop
• So, `count++` is executed $5 \times 3 = 15$ times
• At the end the value of count is 15
Practice problems

Write program segments which do the following:

• Write a program that reads in a test score and prints a letter grade using an if-else control structure.

  100 <= points >= 90  gets grade A
  90 >     points >= 80  gets grade B
  80 >     points >= 70  gets grade C
  70 >     points >= 60  gets grade D
  60 >     points >= 80  gets grade F
  points > 100 or points < 0 prints error message

• Compute the sum from 1 to n for a given n
  Example: user inputs 6, output should be 21 because
  \[1+2+3+4+5+6 = 21\]

• Compute the maximum value among 3 numbers
  Example: user inputs 3, 7, 1. Output 7

• Compute the maximum value among n (arbitrary) numbers
  Example: user inputs 4, 6, 15, 9, 7, 14, 6. Output should be 15.

• Sort three numbers
  Example: user inputs 3, 7, 1. Output 1, 3, 7 (ascending order)
Project Submission and Questions