COS 126 Lecture 2: Introduction to C

Programs are a sequence of FUNCTIONS

FUNCTIONS are built-in or defined by users

library	**#include <stdio.h>**
user-defined	**float f(float x)** **{ return 2.0 - x*x*x; }**

Functions consist of
a sequence of DECLARATIONS
followed by
a sequence of STATEMENTS

DECLARATIONS name variables, define types

float	**float h;**
integer	**int i;**

STATEMENTS manipulate data, control execution

assignment	**inc = 0.0;**
control	**while (inc < 2.0) { ... }**
function call	**printf(...)**

Anatomy of a C program

Sample program:
    print table of values of a function

```
#include <stdio.h>
float f(float x)
   { return 2.0 - x*x*x; }
main()
   {
      float h;
      h = 0.0;
      while (h < 2.0)
         {
            printf("%4.1f %6.3f\n", h, f(h));
            h = h + 0.1;
         }
   }
```

Your goals
    this week: understand programs like this
    next week: write programs like this

printf library function

Contact between your C program
and the outside world

Puts characters on "Standard Output"
default: "terminal" that you're typing at

Formatted output
How do you want the numbers to look?

```
printf("%4.1f %6.3f\n", i, f(i));
```

Very flexible: see K&R pp. 13, 154.

Ex: Could print out title line for table with

```
printf(" x    f(x)\n");
```

Running a program

```
When you type commands,
     you are controlling an abstract machine
     (called the UNIX shell)

COMPILE: convert the program
     from "human's" language (C)
     to "machine's" language (stay tuned)
```

lcc function.c

```
Result of compilation:
     1st try: errors in C program SYNTAX
     eventually: a file named a.out

EXECUTE: "start the machine"
     starts at machine language instruction
     corresponding to first statement of main
```

a.out

```
Result of execution:
     1st try: errors in C program SEMANTICS
     eventually: desired "printf" output
```

```
.       % lcc function.c
.       % a.out
.
.         x    f(x)
.        0.0  2.000
.        0.1  1.999
.        0.2  1.992
.        0.3  1.973
.        0.4  1.936
.        0.5  1.875
.        0.6  1.784
.        0.7  1.657
.        0.8  1.488
.        0.9  1.271
.        1.0  1.000
.        1.1  0.669
.        1.2  0.272
.        1.3 -0.197
.        1.4 -0.744
.        1.5 -1.375
.        1.6 -2.096
.        1.7 -2.913
.        1.8 -3.832
.        1.9 -4.859
```

Characteristic C shortcuts

These two programs are equivalent:

```
#include <stdio.h>
float f(float x)
  { return 2.0 - x*x*x; }
main()
  {
    float i;
    i = -1.0;
    while (i < 2.0)
      {
        printf("%4.1f %6.3f\n", i, f(i));
        i = i + 0.1;
      }
  }
```

```
#include <stdio.h>
float f(float x)
  { return 2.0 - x*x*x; }
main()
  {
    float i;
    for (i = -1.0; i < 2.0; i += .1)
      printf("%4.1f %6.3f\n", i, f(i));
  }
```

Concise programs are the norm in C
Many other languages are more verbose

Example Program: Random Integers

Print 10 random integers

    library function rand (in stdlib.h)
    returns positive integers < RAND_MAX
    RAND_MAX is usually 32768

```
#include <stdio.h>
#include <stdlib.h>
main()
  {
    int i;
    for (i = 0; i < 10; i++)
      printf("%d\n", rand());
  }
```

Output:

```
  16838
  5758
  10113
  17515
  31051
  5627
  23010
  7419
  16212
  4086
```

Random Reals

Print 10 random numbers between 0 and 1

```
#include <stdio.h>
#include <stdlib.h>
main()
  {
    int i;
    for (i = 0; i < 10; i++)
      printf("%f\n", 1.0*rand()/RAND_MAX);
  }
```

C has conventions for converting types
    in expressions with mixed types

Output:

```
  0.513871
  0.175726
  0.308634
  0.534532
  0.947630
  0.171728
  0.702231
  0.226417
  0.494766
  0.124699
```

Random Numbers ??

Example: Print 9-by-9 random pattern

```
#include <stdio.h>
#include <math.h>
main()
{
   int i, j;
   for (j = 0; j < 9; j++)
      {
         for (i = 0; i < 9; i++)
            if ((rand()>>13)&1)
                  printf("*");
            else printf(" ");
         printf("\n");
      }
}
```

Q: Why not just use the following test?

```
   if (rand() % 2) ...
```

A: Random numbers are not random

Ex:

often, rightmost bits alternate
depends on implementation
(see next slide)

Never can have *all* properties of random bits
Ex: sequence is always the same!

Moral: check assumptions about library functions

```
 * * * *
* * * * *
 * * * *
* * * * *
 * * * *
* * * * *
 * * * *
* * * * *
 * * * *
```

```
** ***
   **** *
    *   *
* ***
  **** **
   *   **
 ***    *
 **** ***
  *   ***
```

```
* **    **
***    **
  * * *
**  * **
*  *
*** *** *
**     *
    *  **
*     ***
```

```
 **  **
  **  **
*  **  **
**  **  *
 **  **
  **  **
*  **  **
**  **  *
 **  **
```

```
*     ***
   * *
   **   *
*****   *
** * ****
**  *** *
    ***
 * *
 **    * *
```

```
*   * ***
 **  *   *
 **** * *
* **  **
 *
  ***
*      **
*  *  *
 *     **
```

```
  ****
***    **
     ****
 ****
**    ***
****    *
*    ****
  ****
```

```
 ***
** * *
 **  *  *
** **  **
*** *  *
  ******
* * ***
** ** * *
```

```
* *  **
*  ***  *
 *  **   *
*  * ** *
* *  ***
* * * *
**    **
*** *** *
```

Another Example: Gambler's Ruin

Simulate gambler placing $1 even bets
How long does the game last?

```
#include <stdio.h>
#include <stdlib.h>
main()
  { int i, cash, seed;
     scanf("%d %d", &cash, &seed);
     srand(seed);
     while (cash > 0)
       {
         if ((rand()>>13)&1) cash++; else cash--;
         for (i = 0; i < cash; i++) printf(" ");
         printf("*\n");
       }
  }
```

scanf function takes input from terminal
srand initializes random number generator

```
.     % a.out              % a.out
.     4 1231               4 1234
.        *                      *
.         *                    *
.        *                    *
.         *                    *
.        *                    *
.       *                      *
.      *                        *
.                              *
.                               *
.                                *
.                               *
.                                *
.
```

Hmmmmm.

Gambler's Ruin Experiment

```
#include <stdio.h>
#include <stdlib.h>
int doit(int cash)
  { int cnt;
    for (cnt = 0; cash != 0; cnt++)
      if ((rand()>>13)&1) cash++; else cash--;
    return cnt;
  }
main()
  { int j, t;
    for ( j = 4; j < 10; j++)
      {
        printf("%2d  ", j);
        for (t = 0; t < 5; t++)
          printf(" %7d", doit(j));
        printf("\n");
      }
  }
```

Output:

```
 2         2       6     304       2       2
 3        33      17      15      53      29
 4        22    1024    7820      22      54
 5       243      25      41       7     249
 6       494      14     124     152      14
 7       299      33     531      49      93
 8       218   10650      36   42048     248
 9 174090315   83579     299     759      69
```

Well-studied random process in mathematics
Ref: Feller. Probability Theory and Applications

LEVELS OF ABSTRACTION
    random bit, ruin sequence, experiment