C LANGUAGE INTRODUCTION

CSSE 120—Rose Hulman Institute of Technology

The C Programming Language

- Invented in 1972 by Dennis Ritchie at AT&T Bell Labs
- Has been the main development language for UNIX operating systems and utilities for a couple of decades
- Our Python interpreter was written in C
- Used for serious coding on just about every development platform
- Especially used for embedded software systems
- Is usually compiled to native machine code
  - Faster and less portable than Python or Java

Why C in CSSE 120?

- Practical
  - Several upper-level courses in CSSE, ECE, ME, and Math expect students to program in C
  - None of these courses is a prerequisite for the others.
  - So each instructor had a difficult choice:
    - Teach students the basics of C, which may be redundant for many of them who already know it, or
    - Expect students to learn it on their own, which is difficult for the other students
  - But a brief C introduction here will make it easier for you (and your instructor!) when you take those courses

Why C in CSSE 120?

- Pedagogical
  - Comparing and contrasting two languages is a good way to reinforce your programming knowledge
  - Seeing programming at C's "lower-level" view than Python's can help increase your understanding of what really goes on in a program
  - Many other programming languages (notably Java, C++, and C#) derive much of their syntax and semantics from C
    - Learning those languages will be easier after you have studied C

Some C Language trade-offs

- Programmer has more control, but fewer high-level language features to use
- Strong typing makes it easier to catch programmer errors, but there is the extra work of declaring types of thing
  - “Once an int, always an int”
- Lists and classes are not built-in, but arrays and structs can be very efficient
  - and a bit more challenging for the programmer

Q1

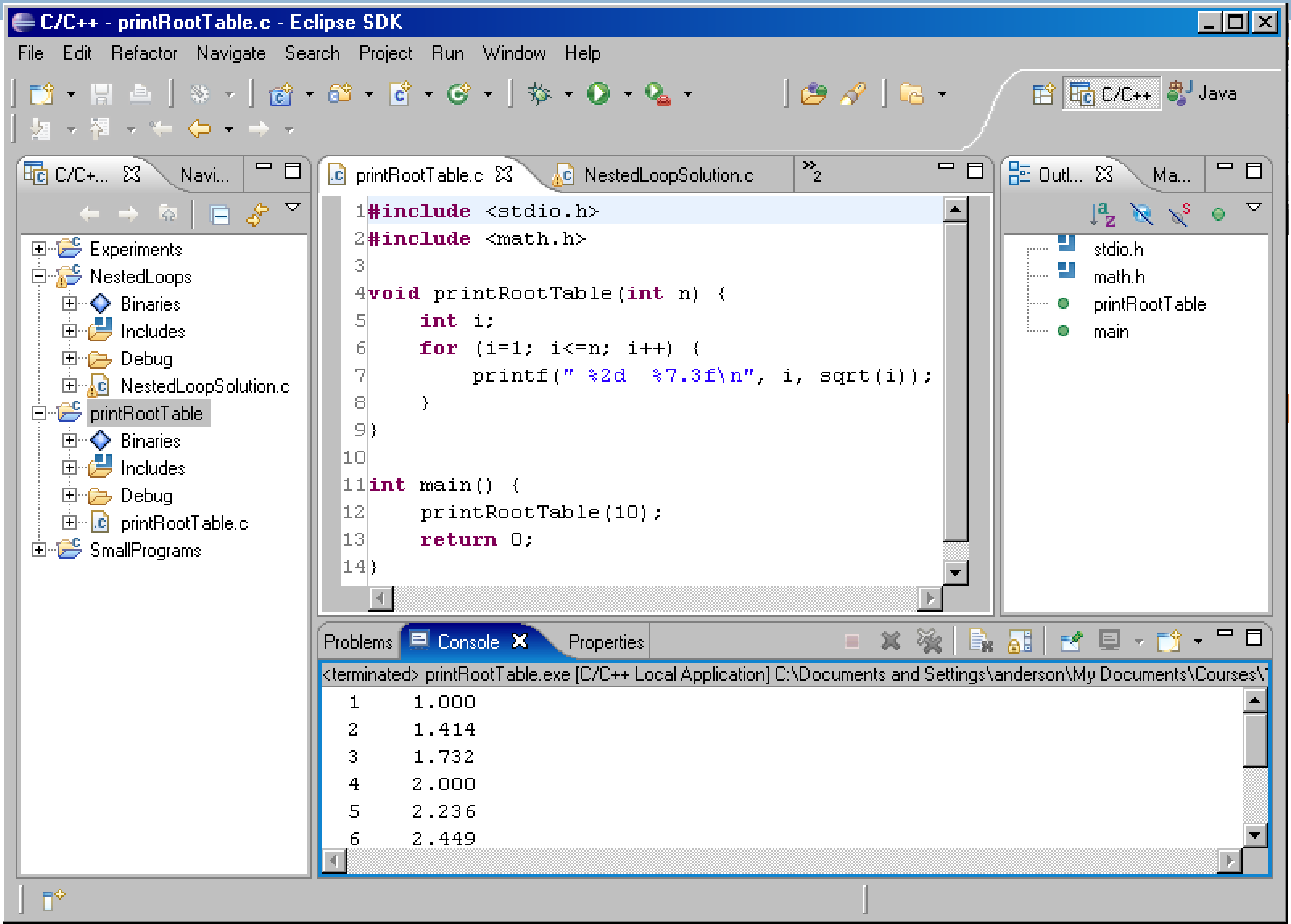
Using C with Eclipse

- We assume that you have already installed the MinGW compiler and C++ tools for Eclipse, as described in the Installation links from the course's Resources page on ANGEL
- You must use a different Eclipse workspace for your C programs than the one you use for Python programs. If you have not already created it,
    - In Windows explorer, create a folder to use for this
    - Back in Eclipse: File→ Switch Workspace, then the Browse button
    - Browse to the folder you created. Click OK

Don’t change your repository structure

- You may be concerned that you have many folders in your repository, some for Python and some for C projects
- Please don’t move any folders in the repository!
  - We use scripts to automatically extract all homework assignments for grading, and they can’t find your work that you move
  - You want to receive grades for the substantial work you do!
- They will be organized *on your laptop* into two Eclipse workspaces.

C/C++ perspective



```
#include <stdio.h>
#include <math.h>

void printRootTable(int n) {
    int i;
    for (i=1; i<=n; i++) {
        printf(" %2d  %7.3f\n", i, sqrt(i));
    }
}

int main() {
    printRootTable(10);
    return 0;
}
```

```
<terminated> printRootTable.exe [C/C++ Local Application] C:\Documents and Settings\anderson\My Documents\Courses\
 1     1.000
 2     1.414
 3     1.732
 4     2.000
 5     2.236
 6     2.449
```

Starting a New Project

- New→C Project. Hello World ANSI C Project (Call it **RootTable**)
- Open src to find the file it created
- Call the file **rootTable.c.**   Finish
- Note that if you right-click rootTable.c,
  Run as … is missing from the context menu
  - Why? unlike in PyDev, each Eclipse C Project must have exactly one code file containing the **main()** function
  - Thus **Run As …** is not even an option for an individual C code file

Parallel examples in Python and C.

```python
from math import *

def printRootTable(n):
   for i in range(1,n):
      print " %2d  %7.3f" % (i, sqrt(i))

def main():
   printRootTable(10)

main()
```

```c
#include <stdio.h>
#include <math.h>

void printRootTable(int n) {
   int i;
   for (i=1; i<=n; i++) {
      printf(" %2d  %7.3f\n", i, sqrt(i));
   }
}

int main() {
  printRootTable(10);
  return 0;
}
```

Recap: Comments in C

- Python comments begin with # and continue until the end of the line
- C comments begin with /* and end with */.
- They can span any number of lines
- Some C compilers (including the one we are using) also allow single-line comments that begin with //

String constants in C

- In Python, character strings can be surrounded by single quotes (apostrophes), or double quotes (quotation marks)
- In C, only double quotes can surround strings
  - An array of chars

  -
    ```
    char s[] = "This is a string";
    printf(s); /* more about printf() soon */
    ```
- Single quotes indicate a single character, which is not the same as a string whose length is 1. Details later

  -
    ```
    char c = 'x';
    printf("%c\n", c);
    ```

Q3c

printf statement

C:
```
printf(" %2d  %7.3f\n", i, sqrt(i));
```

Python equivalent:
```
print " %2d  %7.3f" % (i, sqrt(i))
```

- printf's first parameter is used as a format string
- The values of **printf**'s other parameters are converted to strings and substituted for the conversion codes in the format string.
- **printf** does not automatically print a newline at the end

printf – frequently used conversion codes

code	data type	Example

d	decimal
	(int, long)
		`int x=4, y=5;`
		`printf("nums %3d, %d%d\n", x, y, x+y");`
		`/*prints nums   4, 59*/`

f	real
	(float)
		`float p = 1.3/9, q = 2.875;`
		`printf ("%7.4f %0.3f %1.0f %f\n", p, p, q, q);`
		`/* prints  0.1444 0.144 3 2.875000 */`

lf	real (double)	`double p = 1.3/9, q = 2.875;`
		`printf ("%7.4lf %0.3lf %1.0lf %f\n", p, p, q, q);`
		`/* prints  0.1444 0.144 3 2.875000 */`

c	character
	(char)
		`char letter = (char)('a' + 4);`
		`printf ("%c %d\n", letter, letter);`
		`/* prints e 101 */`

s	string
	(char *)
		`char *isString = "is";`
		`printf("This %s my string\n", isString);`
		`/* prints This is is my string! */`

e	real
	(scientific
	notation)
		`double c = 62345892478;`
		`printf("%0.2f %0.3e  %14.1e", c, c, c);`
		`62345892478.00 6.235e+010        6.2e+010`

Getting Values from Functions

- Just like in Python (almost)
- Consider the function:

  -
    ```
    double convertCtoF(double celsius) {
      return 32.0 + 9.0 * celsius / 5.0;
    }
    ```
- How would we get result from a function in Python?
  - fahr = convertCtoF(20.0)
- What's different in C?
  - Need to declare the type of `fahr`
  - Need a semi-colon

Using if and else

-
  ```
  if m % 2 == 0:
      print "even"
  else:
      print "odd"
  ```
- Python:
  - Colons and indenting

-
  ```
  if (m % 2 == 0) {
      printf("even");
  } else {
      printf("odd");
  }
  ```
- C:
  - Parentheses, braces

else if

-
  ```
  if gpa > 2.0:
          print "safe"
  elif gpa >= 1.0:
          print "trouble"
  else:
          print "sqrt club"
  ```
- Python:
  - Colons and indenting
  - elif

-
  ```
  if (gpa > 2.0) {
          printf("safe\n");
  } else if (gpa >= 1.0) {
          printf("trouble\n");
  } else {
          printf("sqrt club");
  }
  ```
- C:
  - Parentheses, braces
  - else if

Braces are sometimes optional, but…

- Braces group statements
- Can omit for single statement bodies
-
```
if (gpa > 2.0)
      printf("safe");
else if (gpa >= 1.0)
      printf("trouble");
else
      printf("sqrt club");
```

What happens when you add a line of code?

- if (gpa > 2.0)
      printf("safe\n");
      printf(“You have a passing GPA\n”)

- What is printed when gpa is 3.0?
- What is printed when gpa is 1.5?

Nested ifs

-
  ```
  if (n > 0)
          if (a > 0)
                  printf("X");
  else
          printf("Y");
  ```
- **else** goes with closest **if**, *no matter how it's indented.*

```
if (n > 0)  {
     if (a > 0) {
          printf("X");
     }
} else {
     printf("Y");
}
```

Therefore use braces to avoid confusion!

Does C have a boolean type?

- Enter the following C code in Eclipse:

```
void testBoolean(int left, int right) {
    int result = left < right;
    printf("Is %d less than %d? %d\n",
                left, right, result);
}
```

- Add a couple of test calls to your **main()** function:

```
testBoolean(2,3);  testBoolean(3,2);
```

- **0** in C is like **False** in Python
- All other numbers are like **True**

Boolean operators in C

- Python uses the words **and**, **or**, **not** for these Boolean operators. C uses symbols:
  - && means "and“
  - || means "or“
  - ! means "not“
- Example uses:
  - `if (a >= 3 && a <= 5) { … }`
  - `if (!same (v1, v2)) { … }`

Q6

I Could While Away the Hours

- How do you suppose the following Python code would be written in C?

```
n = 10
while n >= 0:
    n = n – 1
    print n
```

- How do you break out of a loop in Python?
- How do you suppose you break out of a loop in C?

Q7

A Little Input, Please

- To read input from user in C, use **scanf()**
- Syntax: **scanf(<formatString>, <pointer>, …)**
- Example:

```
int age;
fflush(stdout);      // Done once prior to scanf
scanf("%d", &age);
```

Q8-Q10

Another Example

Pushes prompt string to user before asking for input.

- To read input from user in C, use **scanf()**
- Syntax: **scanf(<formatString>, <pointer>, …)**
- Example:

```
float x, y;
printf("Enter two real numbers separated by a comma:");
fflush(stdout);
scanf("%f,%f", &x, &y);
printf("Average: %5.2f\n", (x + y)/2.0);
```

Use %f, not %5.2f (don't enter a width for input)

Comma is matched against user input

Rectangular output in C

```
#include <stdio.h>
void rectangleSameNumEachRow(int numRows, int numCols) {
        int i, j;
        for (i=1; i<= numRows; i++) {
                for (j=1; j<=numCols; j++) {
                    printf ("%d", i);
                }
                printf ("\n");
        }
}
int main() {
        rectangleSameNumEachRow(3, 8);
}
```

Output:

```
11111111
22222222
33333333
```

It's easier than Python because `printf()` does not automatically add spaces like Python's `print`.

HW (due Weds): finish nested loops, thatsPerfect