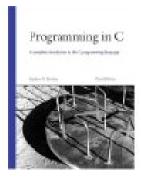# FIRST C PROGRAM

Announcements

- Exam Thursday

- Optional Intro to MATLAB session
  - Thursday hours 7-8 or hours 9-10 in C111
  - Mechanical/Biomedical Engineers who haven't taken ME123/BE100 are **required** to attend

- Homework due Friday 5:00 pm:
  - Fill out team evaluation survey in ANGEL

- Homework due Session 22 (Monday)
  - Reading in C textbook and ANGEL quiz

Reminder: C Textbook

- Kochan's "Programming in C"

- Very readable, like Zelle.

- Recommended highly by two non-CSSE Rose professors

- First assignment and quiz Due before Session 22

```python
from math import *

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

def main():
    printRootTable(10)

main()
```

Parallel examples in Python and C.

```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;
}
```

Getting started

- Create a folder directly on your C: drive, with no spaces in it, like:
  - C:\CProjects
- File→Switch Workspace. Browse to your new folder.   Go to C/C++ perspective.
- New→C Project, Hello World ANSI C Project.
- Call it RootTable.
- Add your name to the .c file it created
- Right click in margin to show line numbers
- Run the project by right-clicking the PROJECT, not the file.

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 //.

The inclusion of header files

`#include` is somewhat like Python's **`from … import *`**

The most commonly included files are *header* files, whose names end with **.h**

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

angle brackets mean that it is a standard C header

If we include a file from our own project, surround it's name with quotes, as in **#include "myFile.h"**

A header file usually contains definitions of constants, and function signatures (without their bodies)

Two lines from **math.h** (we'll explain later):
```
#define M_PI 3.14159265358979323846
double sqrt (double);
```

Other headers: http://www.utas.edu.au/infosys/info/documentation/C/CStdLib.html

Focus on the `main()` Function

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

Every C program must have a function named **main()**

`main`'s return value (In this case 0) is the exit status of the program. Usually, we return 0 to indicate successful completion of the program

This **main( )** function has an empty formal parameter list

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

In a function definition, we must indicate its return type before the name of a function, - In this case, the return type is **int**

The body of a function definition is enclosed in curly braces { … }

Every simple C statement must be followed by a semicolon

The two statements in the body are just like corresponding Python statements

By looking at **main**, how can we tell that **printRootTable** doesn't have to return a value?

`printRootTable()`'s interface

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

void printRootTable(int n) {



}

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

What is the name of the "return type" of the printRootTable() function?
What does that mean?

The formal parameter is called **n**, its type is **int**

Note that this function has no **return** statement. In that case, the return type **must** be declared to be **void**

The type of every formal parameter must be declared

As in Python, if there are multiple formal parameters, they are separated by commas

As in Python, when printRootTable is called, the value of the actual parameter (10) is used to initialize the formal parameter (n)

Notice that we do not provide the type of the actual parameter. Its type is the type of whatever value we pass in. It must "match" the type of the formal parameter

(local) variable declaration

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

void printRootTable(int n) {
    int i;



}



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

> **i** is a local (to the function) variable of the **printRootTable** function

> Its type is **int**

> Unlike in Python, each C variable's and formal parameter's type must be declared before the variable can be used

> Variable declarations must include a type. An optional initialization is allowed, such as `int i = 17;` or `int i = n + 5;`

> A local variable cannot have the same name as a formal parameter of the same function

> Because the variables **i** and **n** are local to printRootTable, you cannot refer to them from anywhere else in the program

i++

- **`i++`** is an abbreviation for **`i = i + 1`**
  - which can also be written **`i += 1`**
- **`i--`** is an abbreviation for **`i = i - 1`**
  - which can also be written **`i -= 1`**
- Some C-programmers write i++ or i-- as part of a more complicated expression.
  - We suggest that you avoid doing that for now.

C's for **loop**

```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));
    }
}
```

Basic syntax is

```c
for (<init>; < test>; <update>) {
    body
}
```

- **init**: usually initializes variables used by the loop

- **test**: if the value of the test is true, the loop body executes

- **update**: After execution of the loop body, this code is executed. Then the **test** code is evaluated again, and if true …