2. Basic Program Structure
ENEE 140

Prof. Tudor Dumitraș
Associate Professor, ECE
University of Maryland, College Park

http://ter.ps/enee140

1

What is Programming (Review)

- Becoming fluent in the language that computers understand
  - Humans are better than computers at doing certain things
  - Computers are better than humans at other things
  - **If you can program, you can do both!**
- Programming stimulates a **way of thinking**
  - Helps you acquire aptitudes and skills applicable in many situations
  - Examples: top-down problem solving, thinking at multiple levels of abstraction, thinking of worst-case scenarios to avoid failures
- Programming is a **creative** process
  - Within the bounds of what computers and programming languages can do

2

2

ENEE 140 Focuses on Programming Principles

- The lectures will discuss important programming principles
  - Most of these are applicable to any programming language
  - C examples will be provided for illustration
- To learn all the details about the C concepts discussed, you must read additional materials
  - The relevant chapters in the textbook
  - Many Internet resources on C programming (Google is your friend)
  - Quick documentation: press F1 in CLion

3

3

First Principles: Code Quality

- Learning objective: write high-quality code
  - **Correctness**: the code should do what it's supposed to do (and nothing else!)
  - **Maintainability**: other programmers should find the code easy to read and to modify
- Other code-quality attributes that we will not emphasize in ENEE 140
  - Efficiency
  - Robustness
  - Security

4

4

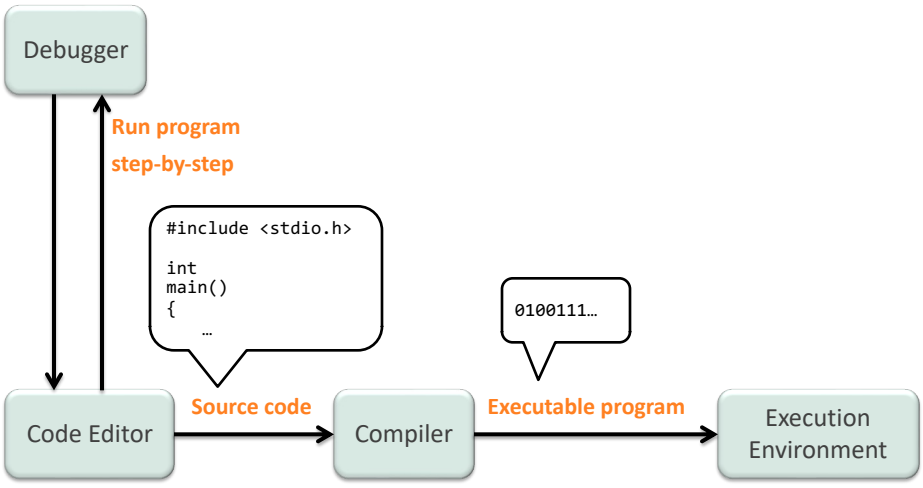
The C Language

- A low-level language
  - No operations for manipulating composite types (e.g. strings, lists, arrays), no memory management, no input/output facilities
    - The standard library provides some of these facilities
  - A small language
    - Can be learned quickly
- Topics **covered** in ENEE 140:
  - Data types, type conversions
  - Operators (arithmetic, relational, logic, bitwise, etc.)
  - Flow control (loops, branches)
  - Functions
  - Multi-dimensional arrays
- Topics **not covered** in 140:
  - Recursion
  - Pointers

5

5

The Programming Toolchain



6

6

Getting Started in C

`int main() { … }`	each program must have one main() function
`return …`	exit the function
`;`	end each statement with a semicolon
`#include <stdio.h>`	use functions from the standard library
`printf(…)`	print something
`// …` or `/* … */`	comments (ignored by the compiler)

Use comments to explain what your program is trying to do

7

7

We've Seen: Programming Concepts

- Write programs to compute quantities difficult to work out in your head
  - Programming languages provide **variables** and **arithmetic operations**
- Come up with step-by-step procedure for arriving at the result
  - Programming languages provide **loops** and **branching statements**
- Break down a problem into simpler steps
  - Programming languages provide **functions**
  - Helpful for solving problems from the top down to the small details
- Communicate with the user
  - Programming languages provide **input/output** mechanisms

8

8

Variables

- Correspond to memory locations that hold data and that may be manipulated in your program
- Must be **declared**:

`int a;`	integer variable
`float b;`	floating-point variable (has fractional part)

- Must be **assigned** a value

`a = 1;`	assignments change the value
`b = 1.5;`	stored in the variable

- May be used in **expressions**

`a < 10`	comparison test
`b = a + 1;`	value of arithmetic operation used in assignment

9

9

Assignment vs. Equality Testing

`a = a + 1;`	assignment (increment a by 1)
`a == a + 1`	equality testing (result is false)

10

10

Arithmetic Operations

`+ - * /`

- **Integer** arithmetic
  - **Division truncates**: the fractional part is discarded

```
int a = 1 / 2;
```
value of a is 0

- **Floating-point** arithmetic
  - **Division does not truncate**

```
float b = 1.0 / 2.0;
```
value of b is 0.5

11

11

Relational Operators

- Used for making comparisons

`==`	Equal	`>`	Greater Than
`!=`	Not Equal	`<=`	Less Than or Equal
`<`	Less Than	`>=`	Greater Than or Equal

- Work on both integers and floats
- Good programming practice: **avoid (in)equality tests with floats**!
  - Example:

```
b != 0
```
if b is a float, try to use <= or >= instead

  - Results of floating-point operations are imprecise (more on this later)

12

12

Combining `int`s and `float`s in Expressions

- If an arithmetic operator has integer operands
  - Integer arithmetic is used

```
int a = 1;
int b = a / 2;          value of b is 0
```

- If an arithmetic operator has at least one floating-point operand
  - Floating-point arithmetic is used

```
float a = 1;
float b = a / 2;        value of b is 0.5
```

- Expression type is evaluated before assignment

```
float b = 1 / 2;        value of b is 0
float b = 1.0 / 2.0;    value of b is 0.5
```

13

13

Symbolic Constants

- Good programming practice: if you have **constants** in your program, give them a **symbolic name**
- Declaring constants
  - Modern constant declarations

```
const float pi = 3.14;
```

  - Old-school constant declarations (traditionally uppercase)

```
#define PI 3.14          no type, no semicolon
```

- Using constants

```
float radius = 1;
float circumference = 2 * PI * radius;
```

14

14

while loops

- Repeating program statements while a condition holds

```
while (condition) {
    …
}
```

condition is tested first

- Example: print "Hello world" 10 times
  - You need a variable to count the number of iterations. Let's call it `i`

```
int i = 0;
while (i < 10) {
    printf ("Hello World\n");
    i = i + 1;
}
```

initialize i

iterate while i is less than 10

increment i

If you find yourself copy-pasting code several times, think about rewriting your program to use a loop!

15

15

Review of Lecture

- What did we learn?

16

16

Next Steps

- Next lecture
  - Character Input/Output
- Assignments for this week
  - Review **K&R 1.2** and make sure you understand how while loops and arithmetic operations work
  - Read **K&R Chapters 1.3, 1.5, 2.1, 2.6, 3.1, 3.2**
  - **Syllabus Quiz**, due on Friday at 11:59 pm
    - Tests your understanding of the ENEE 140 syllabus
  - **Quiz 2**, due on Sunday at 11:59 pm
    - Tests reading assignment
  - Weekly challenge: `word_per_line.c`
  - Homework: `lab02.pdf`, due on Friday at 11:59 pm

17

17