Software Lesson 1 Outline

What is Software? A Program? Data?

Software, for our purposes, is just a word that means "programs."

A ***program*** is a specific, precise, detailed description of:

- a collection of ***data*** – in RAM, on disk, etc –

and

- a **sequence of actions** on those data.

The actions in a program are known as ***instructions***.

In computing, ***data*** are values stored in ***storage locations***: for example, RAM, disk, etc.

What are Instructions?

The actions in a program are known as ***instructions***.

Examples:

- ***Arithmetic/Logical calculation***: for example, add, subtract, multiply, divide, square root, cosine, etc.
- ***Memory operations***: load from or store into RAM
- ***I/O***: read from or write to secondary storage
- ***Branch***: jump to an instruction that is out of sequence
- ***Repetition***
- ***Allocation*** of resources

… and many more.

What is a Programming Language?

A ***programming language*** is a **well-defined set of rules** for specifying a program's collection of data and sequence of actions.

Examples: C, C++, Fortran 90, Java, Basic, HTML, Perl, SAS, Haskell, Prolog, Pascal, Unix shell, x86 assembly language, etc.

See:

https://en.wikipedia.org/wiki/List_of_programming_languages

What is Source Code? What is a Source File?

Source code is a sequence of instructions, written in a **human-readable** programming language, that constitutes a program, or a piece of a program.

An example is shown on slide #8.

A ***source file*** is a file of source code.

What is an Operating System?

An ***operating system*** is a program that manages interactions between:

- users and hardware;
- users and software;
- hardware and software;

... and so much more.

Operating System Examples

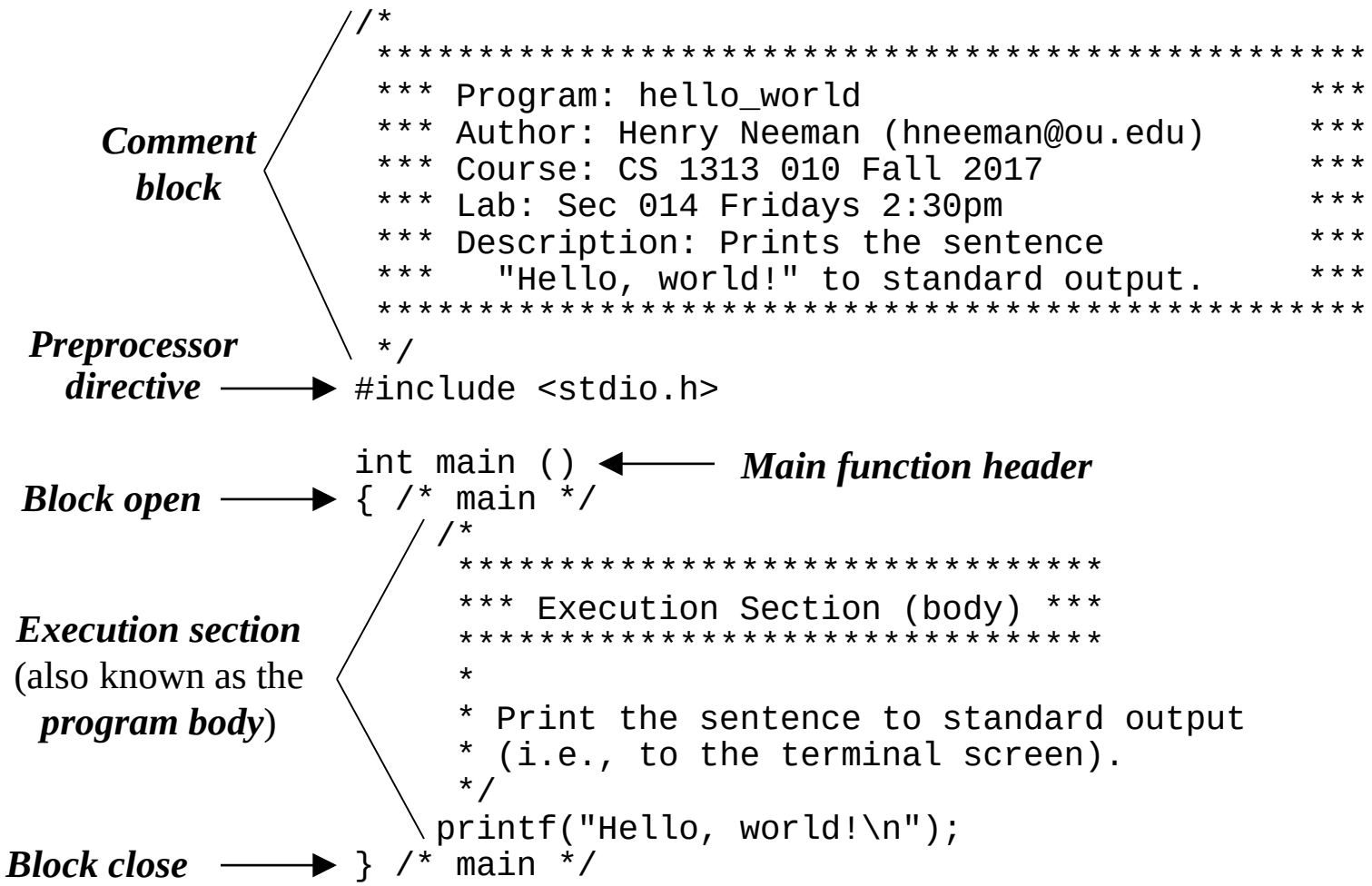
- PC Operating Systems
  - MS Windows/MS-DOS
  - MacOS – built on top of FreeBSD (Unix)
- Phone and Tablet Operating Systems
  - iOS (iPhone, iPad)
  - Android (many cell phones and tablets) – built on top of Linux
  - Windows Mobile
  - BlackberryOS
- Unix
  - Linux (portable)
  - FreeBSD (portable, underlies MacOS X)
  - Solaris (Oracle)
  - AIX (IBM)
  - HP-UX (Hewlett-Packard)
  - Unicos (Cray)

A Simple C Program

```
/*
 ************************************************
 *** Program: hello_world                     ***
 *** Author: Henry Neeman (hneeman@ou.edu)    ***
 *** Course: CS 1313 010 Fall 2017            ***
 *** Lab: Sec 014 Fridays 2:30pm              ***
 *** Description: Prints the sentence         ***
 ***   "Hello, world!" to standard output.    ***
 ************************************************
 */
#include <stdio.h>

int main ()
{ /* main */
    /*
     *********************************
     *** Execution Section (body) ***
     *********************************
     *
     * Print the sentence to standard output
     * (i.e., to the terminal screen).
     */
    printf("Hello, world!\n");
} /* main */
```

Anatomy of a Simple C Program



Block Delimiters

The open curly brace, also known as the left brace,

{

acts as the start of a ***block*** and is known as the

***block open*.**

The close curly brace, also known as the right brace,

}

acts as the end of a ***block*** and is known as the

***block close*.**

The block open and block close are said to ***delimit*** the block:

they indicate where the block begins and

where the block ends.

Delimit: Indicate where something begins and ends.

What Is a Comment? #1

A ***comment*** is a piece of text in a source file that:

- **tells human beings** (for example, programmers) something useful about the program,

BUT

- is **ignored by the compiler**, so it has absolutely no affect on how the program runs.

In C, the start of a comment is indicated by

```
/*
```

and the end of a comment is indicated by

```
*/
```

All text appearing between these ***comment delimiters*** is part of the comment, and therefore is ignored by the compiler.

Delimit: Indicate where something begins and ends.

What Is a Comment? #2

A ***comment*** is a piece of text in a source file that:

- **tells human beings** (for example, programmers) something useful about the program,

BUT

- is **ignored by the compiler**, so it has absolutely no affect on how the program runs.

In C, the start of a comment is indicated by

```
/*
```

and the end of a comment is indicated by

```
*/
```

A comment **can use multiple lines** of text. The delimiters **DON'T** have to be on the same line.

Are Comments Necessary?

Comments are ignored by the compiler, so strictly speaking they aren't needed to compile and run.

But, if you don't put them into one of your CS1313 programming projects, **YOU MAY LOSE A FULL LETTER GRADE OR MORE** on that project.

Why?

Comments tell human beings useful things about your program.

They help **programmers** – including you, a month later when you've forgotten everything about your program – to understand your program.

They also tell **graders** that you know what the heck you're doing.

`hello_world.c` with Comments

```
/*
 ************************************************
 *** Program: hello_world                     ***
 *** Author: Henry Neeman (hneeman@ou.edu)    ***
 *** Course: CS 1313 010 Fall 2017            ***
 *** Lab: Sec 014 Fridays 2:30pm              ***
 *** Description: Prints the sentence         ***
 ***   "Hello, world!" to standard output.    ***
 ************************************************
 */
#include <stdio.h>

int main ()
{ /* main */
    /*
     *********************************
     *** Execution Section (body) ***
     *********************************
     *
     * Print the sentence to standard output
     * (i.e., to the terminal screen).
     */
    printf("Hello, world!\n");
} /* main */
```

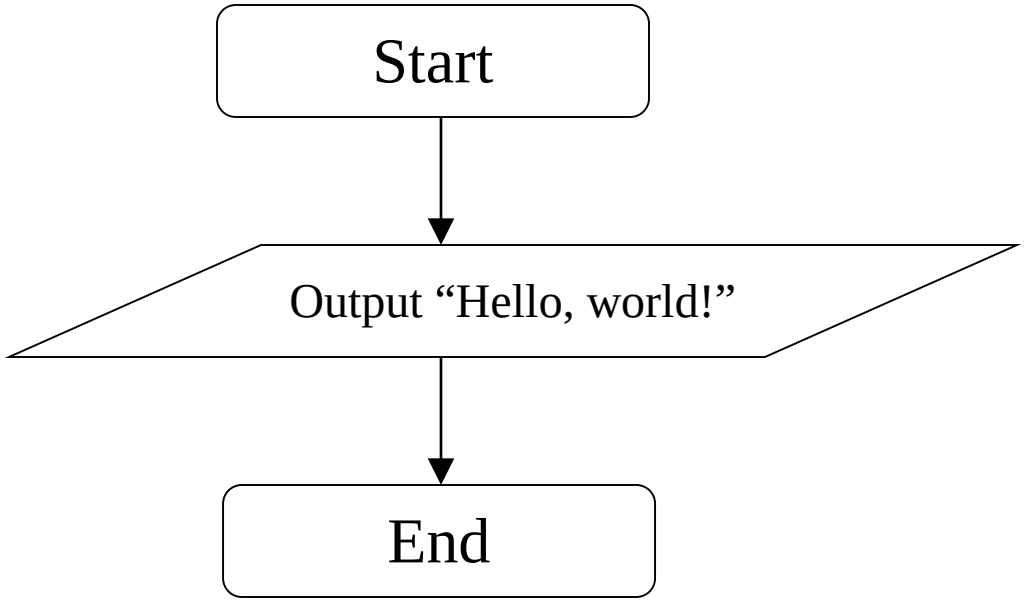
`hello_world.c` without Comments

```
#include <stdio.h>

int main ()
{
    printf("Hello, world!\n");
}
```

Flowchart for `hello_world.c`

```
int main ()
{
    printf("Hello, world!\n");
}
```



An **oval** denotes either the start or the end of the program, or a halt operation within the program (which we’ll learn about later).

A **parallelogram** denotes either an input operation or an output operation.

An **arrow** denotes the flow of the program.

Reference:
http://www.edrawsoft.com/flowchart-symbols.php

Outputting, Compiling and Running a C Program

```
% cat hello_world.c <----
/*
 ************************************************
 *** Program: hello_world                     ***
 *** Author: Henry Neeman (hneeman@ou.edu)    ***
 *** Course: CS 1313 010 Fall 2017            ***
 *** Lab: Sec 014 Fridays 2:30pm              ***
 *** Description: Prints the sentence         ***
 ***   "Hello, world!" to standard output.    ***
 ************************************************
 */
#include <stdio.h>

int main ()
{ /* main */
    /*
     *********************************
     *** Execution Section (body) ***
     *********************************
     *
     * Print the sentence to standard output
     * (i.e., to the terminal screen).
     */
    printf("Hello, world!\n");
} /* main */
% gcc -o hello_world hello_world.c <----
% hello_world <----------------------------
Hello, world! <--------------------------
```

Anatomy of Outputting, Compiling and Running

```
% cat hello_world.c
/*
 ************************************************
 *** Program: hello_world                     ***
 *** Author: Henry Neeman (hneeman@ou.edu)    ***
 *** Course: CS 1313 010 Fall 2017            ***
 *** Lab: Sec 014 Fridays 2:30pm              ***
 *** Description: Prints the sentence         ***
 ***   "Hello, world!" to standard output.    ***
 ************************************************
 */
#include <stdio.h>

int main ()
{ /* main */
    /*
     *********************************
     *** Execution Section (body) ***
     *********************************
     *
     * Print the sentence to standard output
     * (i.e., to the terminal screen).
     */
    printf("Hello, world!\n");
} /* main */
% gcc -o hello_world hello_world.c
% hello_world
Hello, world!
```

Unix command to output to the screen

Unix command to compile

Unix command to run

Program output

A Less Simple C Program #1

```
/*
 ************************************************
 *** Program: my_add                          ***
 *** Author: Henry Neeman (hneeman@ou.edu)    ***
 *** Course: CS 1313 010 Fall 2017            ***
 *** Lab: Sec 014 Fridays 2:30pm              ***
 *** Description: Input two integers, compute ***
 *** their sum and output the result.         ***
 ************************************************
 */
#include <stdio.h>

int main ()
{ /* main */
   /*
    ***************************
    *** Declaration Section ***
    ***************************
    *
    *****************************
    * Named Constant Subsection *
    *****************************
    */
    const int program_success_code =  0;
   /*
    *****************************
    * Local Variable Subsection *
    *****************************
    *
    * addend: The addend value that the user inputs.
    * augend: The augend value that the user inputs.
    * sum:    The sum of the addend and the augend,
    *   which is output.
    */
    int addend, augend, sum;
```

Continued on the next slide.

A Less Simple C Program #2

```
/*
 *************************
 *** Execution Section ***
 *************************
 *
 ************************
 * Greeting Subsection *
 ************************
 *
 * Tell the user what the program does.
 */
 printf("I'll add a pair of integers.\n");
/*
 ********************
 * Input subsection *
 ********************
 *
 * Prompt the user to input the addend & augend.
 */
 printf("What two integers do you want to add?\n");
/*
 * Input the integers to be added.
 */
 scanf("%d %d", &addend, &augend);
```

Continued on the next slide.

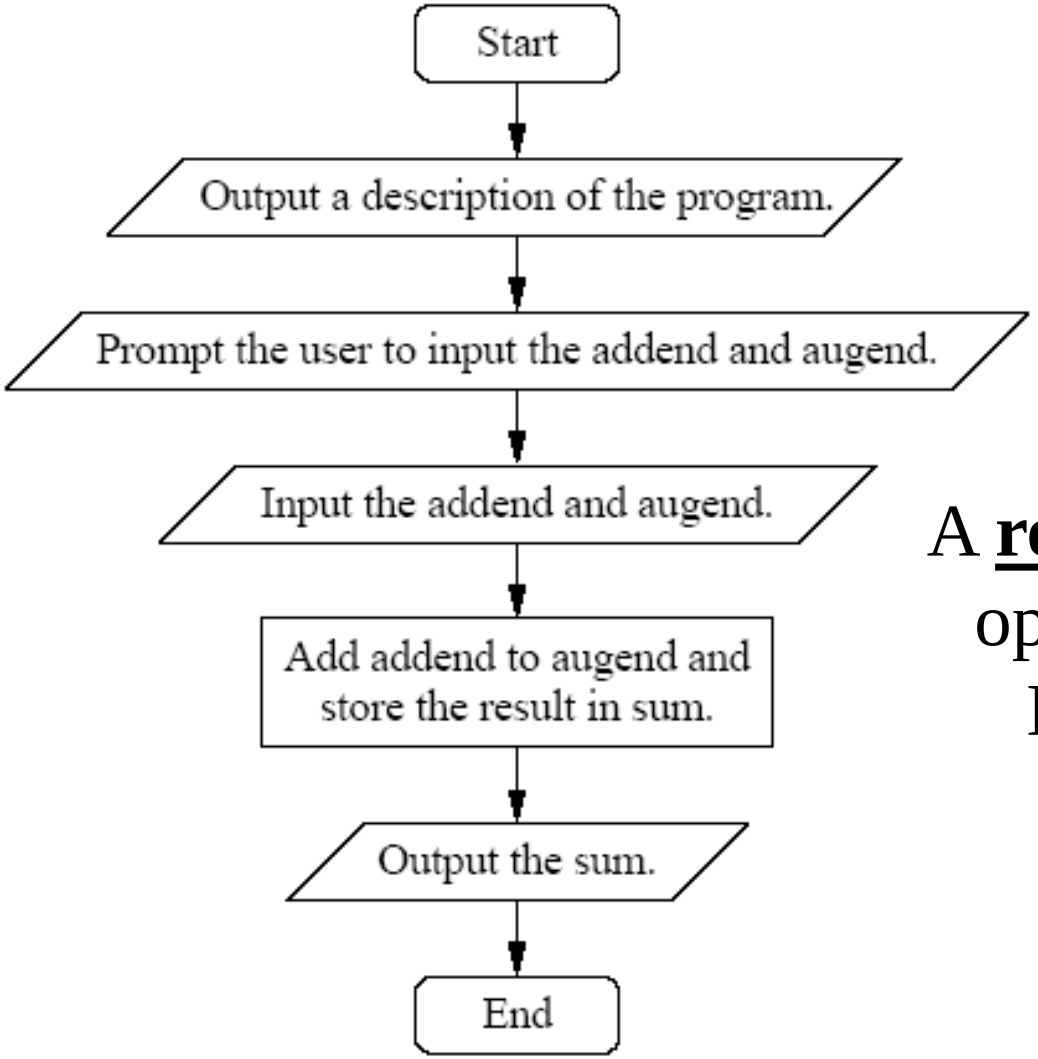
A Less Simple C Program #3

```
    /*
     *************************
     * Calculation Subsection *
     *************************
     *
     * Calculate the sum.
     */
     sum = addend + augend;
    /*
     *********************
     * Output Subsection *
     *********************
     *
     * Output the sum.
     */
     printf("The sum of %d and %d is %d.\n",
         addend, augend, sum);
     return program_success_code;
} /* main */
```

A Less Simple C Program: Compile & Run

```
% gcc -o my_add my_add.c
% my_add
I'll add a pair of integers.
What two integers do you want to add?
5 7
The sum of 5 and 7 is 12.
% my_add
I'll add a pair of integers.
What two integers do you want to add?
1593
09832
The sum of 1593 and 9832 is 11425.
```

Flowchart for `my_add.c`



A **rectangle** denotes an operation other than I/O or branching (for example, calculation).