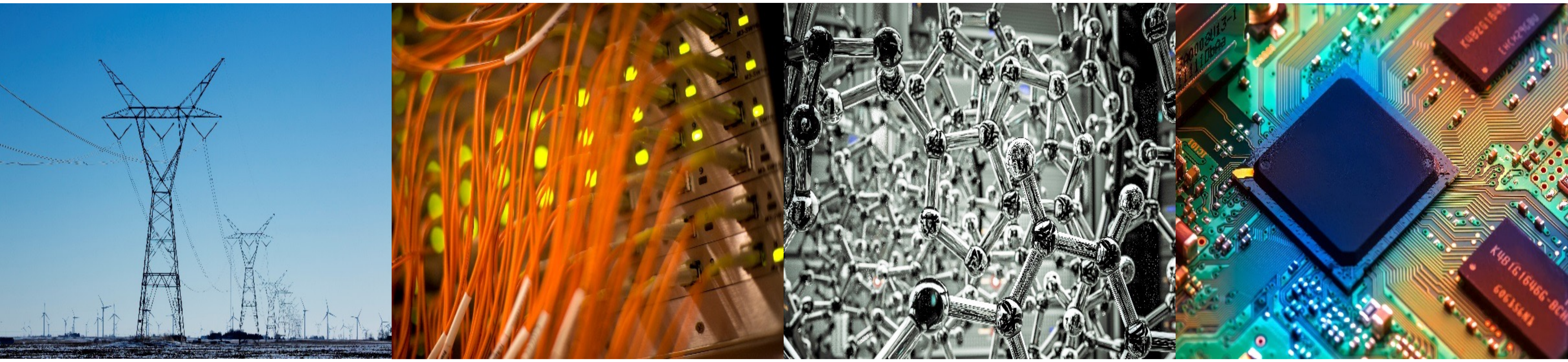
ECE 220 Computer Systems & Programming

Lecture 5 – Introduction to C

February 3, 2026



- Mock quiz (extra-credit) should be taken at CBTF this week
- Quiz1 (LC-3 concepts & programming) is next week

ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

C – Higher Level Language

Gives symbolic names to values

- don't need to know which register or memory location

Provides abstraction of underlying hardware

- operations do not depend on instruction set
- example: can write "`a = b * c;`", even though LC-3 doesn't have a multiply instruction

Provides expressiveness

- use meaningful symbols that convey meaning
- simple expressions for common control patterns (if-then-else)

Enhances code readability

Safeguards against bugs

- can enforce rules or conditions at compile-time or run-time

Basic C Program

```
/*
 * My first program in C. It will print the value of PI
 * and then exit.
 */
#include <stdio.h>
#define PI 3.1416f
int main(){
    float pi = PI;
    printf("pi=%f\n", pi);
    return 0;
}
```

a. Comment
b. Preprocessor directives
c. Main function
d. Variable declaration (type, identifier, scope)
e. I/O
f. Return value
g. Statement termination

Characteristics of C

C is a **procedural language**

- the program specifies an explicit sequence of steps to follow to produce a result; program is composed of functions (aka subroutines)

C programs are **compiled** rather that interpreted

- a compiler translates a C program into machine code that is directly executable on hardware
- interpreted programs (e.g. MATLAB) are executed by another program, called interpreter

C programs are **statically typed**

- the type of each expression is checked at compile time for type inconsistencies (e.g., `int x = 3.411;`)
    - ➢ What is the value of x in this case?

Compiling a C Program

Preprocessor

- macro substitution
- conditional compilation
- "source-level" transformations
  - output is still C

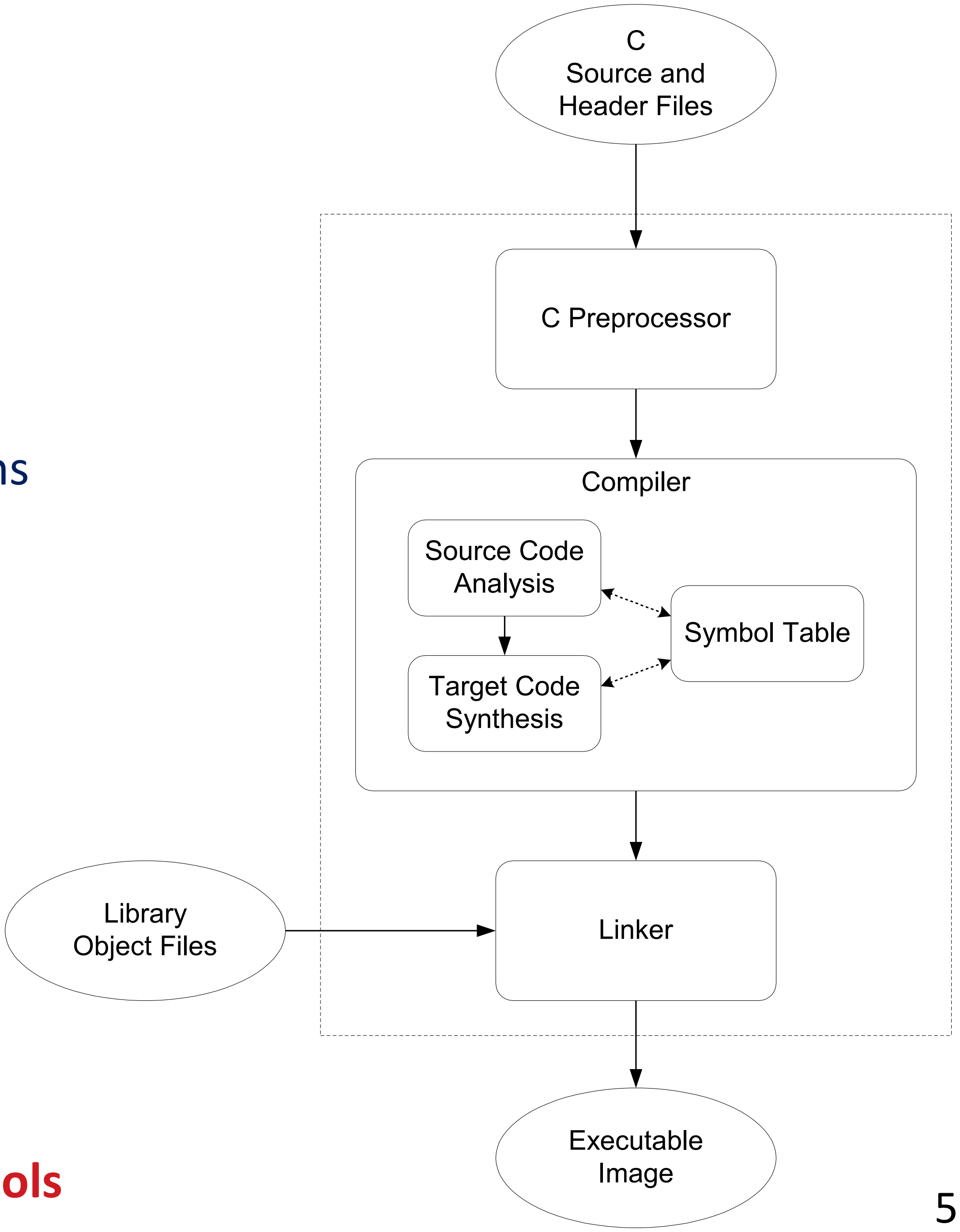
Compiler

- generates object file
  - machine instructions

Linker

- combine object files (including libraries) into executable image

✓ **gcc compiler – invoke all these tools**



ECE ILLINOIS

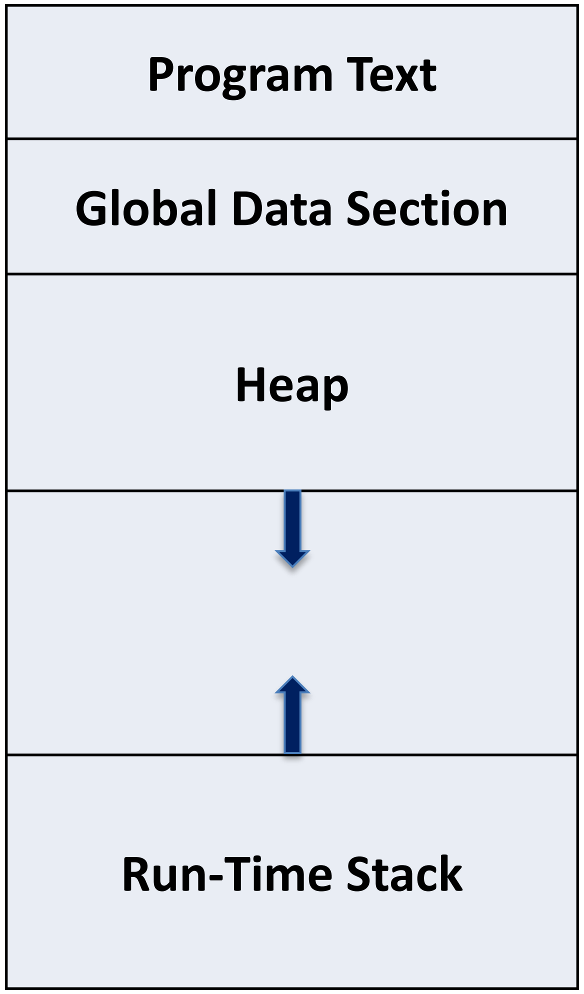
Variables in C

- **int** (long, long long, unsigned), can also use hex representation 0xD
- **float, double**
- **char**

- ❑ ***const*** (qualifier)
- ❑ ***static*** (qualifier)

Scope: local vs. global

Storage class: static vs. automatic



Operators in C

- Expression vs. Statement
- The Assignment Operator (=):
- ‘=‘ vs. ‘==‘
- Arithmetic Operators:
- Order of evaluation:

  precedence `x = 2+3*4;`

  associativity `x = 2+3-4+5;`

  parentheses `x = a*(b + c)*d/2;`
- Logical Operators: ______________________________
- Bitwise Operators: ______________________________
- Relational Operators: ___________________________

Operators in C (continued)

- Increment/Decrement Operators: ++, -- (pre vs. post)

```
example 1: x = 4; y = ++x;
example 2: x = 4; y = x++;
```

  - ➢ What is the value of x and y after increment?

- Special operator (conditional):

  variable = condition ? value_if_true : value_if_false;

```
example: x = (y<z) ? 5 : 7;
/* if y<z, x=____; otherwise, x=____ */
```

- Compound Assignment Operators:

  `a += b;` ⟺ `a = a + b;`

❖ Expression with multiple operators → see Table 12.5 of textbook

Basic I/O

```
#include <stdio.h>          /* header file for standard I/O */
```

```
printf                      printf("%d is a prime number", 43);
/*print to screen*/         printf("43 + 59 in decimal is %d\n", 43+59);
                            printf("a+b=%f\n", a+b);
                            printf("%d+%d=%d\n", a, b, a+b);
```

```
scanf                       scanf("%c", &nextchar);
/*get user input*/          scanf("%f", &radius);
                            scanf(%d %d", &length, &height);
```

- Formatting option: %d, %x, %c, %s, %f, %lf, \n
- Use "**man**" to look up library functions

C Programming Exercise 1

```
#include <stdio.h>
int main(){
/* declare integer variables x, y and z */


/* set x to 3, set y to x² */


/* left shift y by x number of bits */


/* perform bitwise OR on x and y, store the result to z */


/* print z */

return 0;
}
```

C Programming Exercise 2

```
/*
 * Write a C program to convert Fahrenheit to Celsius.
 * C = (F-32)*5/9
 */

/* preprocessor directives */



int main(){
   /* declare variables (as float) for input and output */


   /* prompt user to enter an input value for conversion */
```

```
    /* get user input in Fahrenheit */

    /* calculate the output value in Celsius */

    /* print the result */

    /* return out */

}
```