COP 3223H: Introduction to C Programming

Fall 2023

University of Central Florida

Dr. Kevin Moran

Week 2- Class 1: Arithmetic Expressions & Library Functions

Administrivia

- Eustis assignment out
  - Due on Sunday (Sept 3rd)
- Syllabus Quiz has been posted to Webcourses
  - Due on today (Sept 3rd) - should only take a few mins
- Small Programming Assignment 1
  - Out Today
  - Due next Friday (September 8th)
- *No Class on Monday!! (Labor Day)*

Today's Agenda

1. 15 mins for Working on Connecting to Eustis
2. Discuss Arithmetic Expressions
3. Discuss Library Functions (if we have time)

15 Mins - Eusits Setup Assignment

Arithmetic Expressions

Recap of Last Class

- Common Statements executed in C
  - printf()
  - scanf()
  - return
  - Assignment Statements
- Operators
  - Assignment (=)

So now that we learned to write common C statements, what else can we do besides collect/ display values stored in memory?

Arithmetic Expressions

- Lots of problems that programmers solve involve the use of formulating mathematical expressions.
- In this course we will only focus on arithmetic operations (think algebra level)
  - Addition
  - Subtraction
  - Multiplication
  - Division
  - Modular (modulus)

Modulus Operator

- You may not have heard about the modulus operator (remainder operator).
- The modulus operator returns the remainder value of a division result.
- Example: $\frac{4}{3}$ would result with the remainder 1
- The symbol denoted in C uses **%** to represent the modulus operator.
  - In mathematics (such as discrete mathematics) the notation *mod* also represents the modulus operator. In this course, we will only use the notation **%**.

```
int result = 4 & 3;
printf ("4 & 3 = %d\n”, result);
```

Arithmetic Expressions

Arithmetic Operator	Meaning	Examples
+	addition	`5 + 2 = 7` `5.0 + 2.0 = 7.0`
-	subtraction	`5 - 2 = 3` `5.0 - 2.0 = 3.0`
*	multiplication	`5 * 2 = 10` `5.0 * 2.0 = 10.0`
/	division	`5.0 / 2.0 = 2.5` `5 / 2 = 2`
%	remainder	`5 % 2 = 1`

Casting

- Casting is converting an expression to a different type by writing the desired type in parentheses in front of the expression.

```
double n;
double x = 0.5;

n = (int)(9 * 0.5); //casting
```

What value does n *hold?*

a) 4
b) 4.0
c) 4.5
d) 5

Writing Mathematical Formulas in C

Mathematical Formula	C Expression
$b^2 - 4ac$	`b * b - 4 * a * c;`
$a + b - c$	`a + b - c;`
$\frac{a+b}{c+d}$	`1 / (1 + x * x);`
$\frac{1}{1+x^2}$	`1 / (1 + x * x);`
$a * -(b+c)$	`a * - (b + c);`

Formatting Output

- C allows you to format output of numbers for consistency.
  - You can control the number of spaces
  - Text automatically aligns to the right

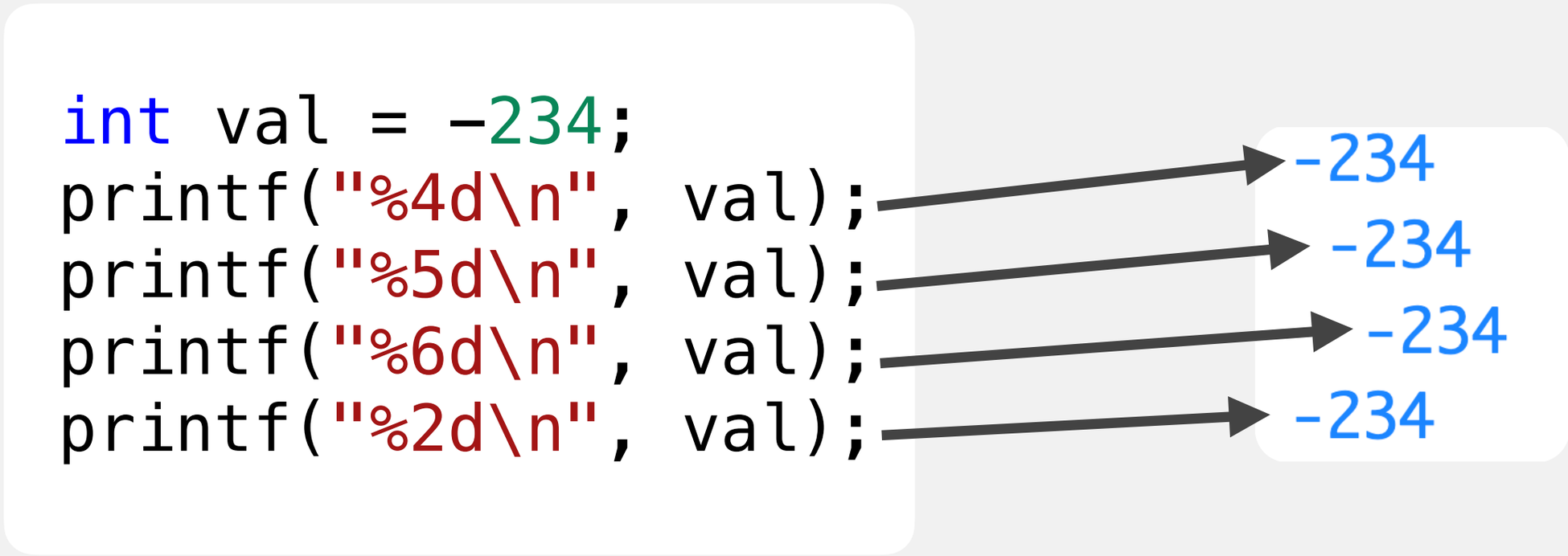
```
int val = 234;
printf("%d\n", val);
printf("%4d\n", val);
printf("%5d\n", val);
printf("%6d\n", val);
printf("%1d\n", val);
```

```
234
234
234
234
234
```

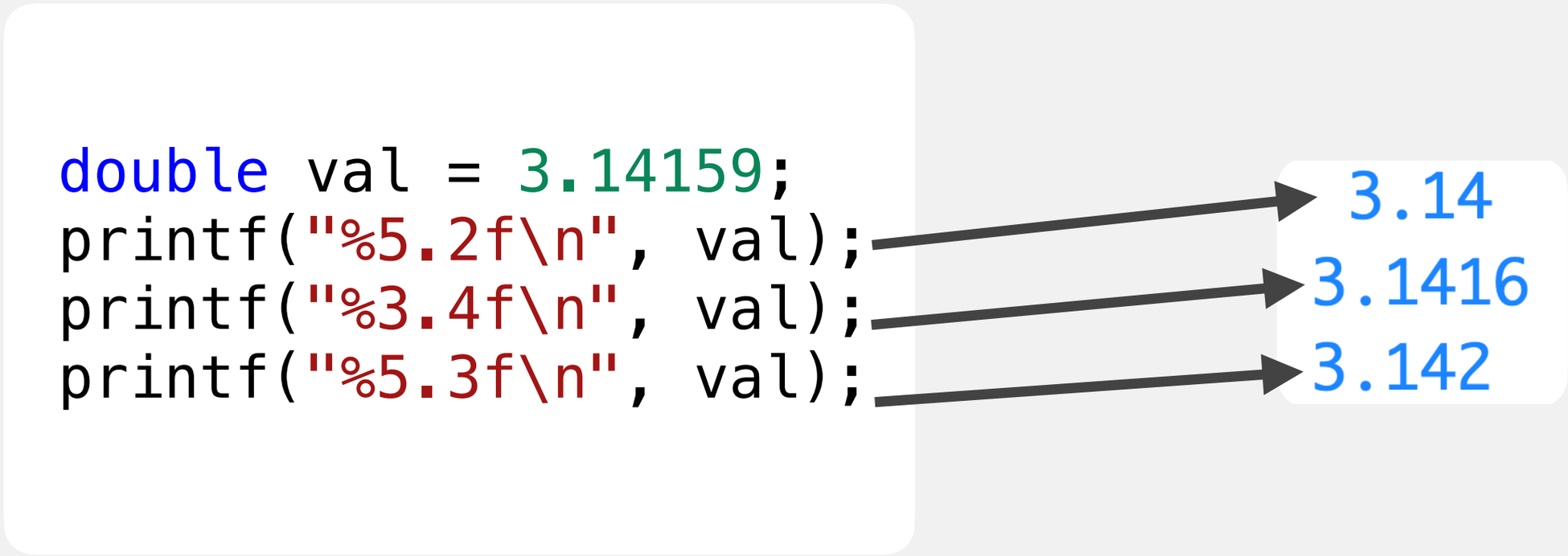
Code File

Output

Formatting Output (another example)



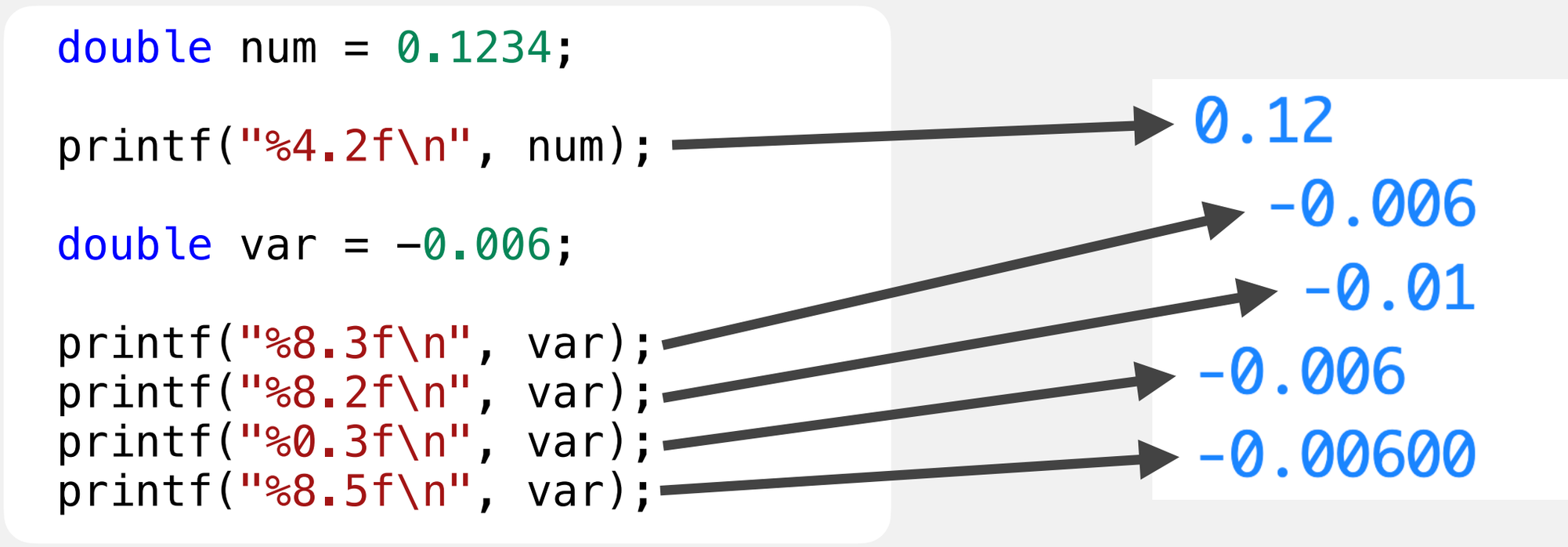
Formatting Output (more examples)



Code File

Output

Formatting Output (more examples)



```
double num = 0.1234;

printf("%4.2f\n", num);

double var = -0.006;

printf("%8.3f\n", var);
printf("%8.2f\n", var);
printf("%0.3f\n", var);
printf("%8.5f\n", var);
```

```
0.12
  -0.006
   -0.01
-0.006
-0.00600
```

Code File

Output

Types of Errors in Programming

- Compile Time Errors: Compiler cannot build code — errors in C syntax.
- Run Time Errors: Code compiles, but crashes or halts while running.
- Logical Errors: Code compiles and runs, however, does not perform the task correctly.
  - WARNING! This type of errors is the worst to have!!

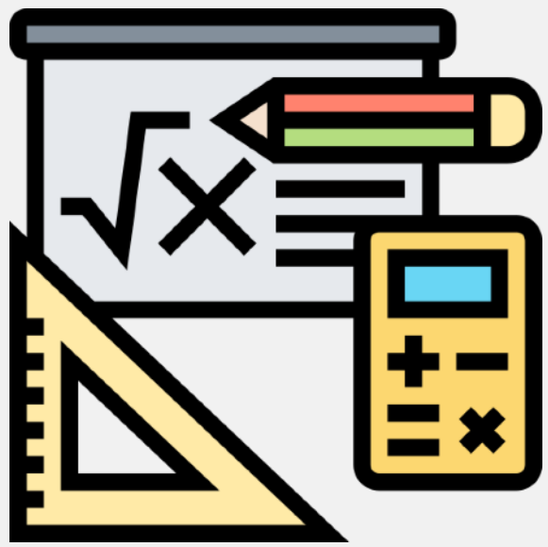
Library Functions

Predefined Functions (Library Functions)

- Code reuse - reusing program fragments that have already been written and tested whenever possible.
- Header files in C contain functions that can reused.
- e.g., `<stdio.h>` has `printf()` and `scanf()`.

The Math Library

- The C language has a math library with predefined functions that perform certain mathematical tasks.
- Task Examples: square root, Trigonometry, etc…
- `#include <math.h>` imports all reusable math functions



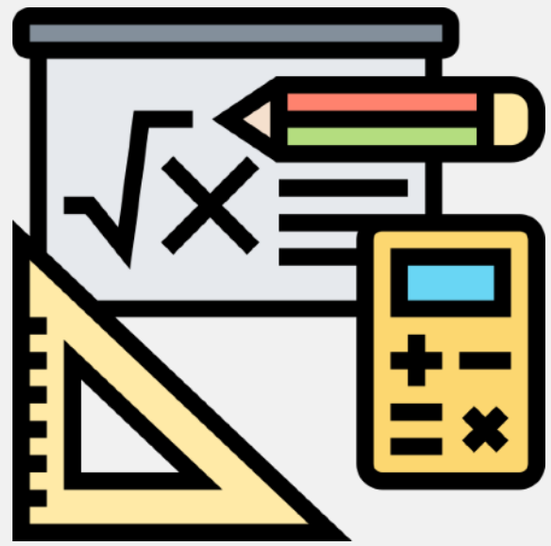
Math Library Functions

Function	Header File	Purpose	Argument(s)	Result
`abs(x)`	`<stdlib.h>`	Absolute Value	`int`	`int`
`ceil(x)`	`<math.h>`	Round Up	`double`	`double`
`cos(x)`	`<math.h>`	Cosine	`double (radians)`	`double`
`exp(x)`	`<math.h>`	Natural Exponent	`double`	`double`
`floor(x)`	`<math.h>`	Round Down	`double`	`double`
`log(x)`	`<math.h>`	Natural Logarithm	`double`	`double`
`log10(x)`	`<math.h>`	Base 10 Logarithm	`double`	`double`
`pow(x,y)`	`<math.h>`	x^y	`double`	`double`
`sin(x)`	`<math.h>`	Sine	`double`	`double`
`sqrt(x)`	`<math.h>`	Square Root	`double`	`double`
`tan(x)`	`<math.h>`	Tangent	`double`	`double`

Class Demo

- Write a program that computes the quadratic function.
  - This is defined as follows:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



The Standard Library

- `#include <stdlib.h>`
- Contains a variety of pre-defined functions for various purposes.

Function from <stdlib.h>	What it does…
`rand()`	Generates a random number from 0-MAX
`srand()`	Seeds the random number generator for `rand()`

- The seed allows for a sequence to be generated for rand().
  - To generate higher quality random numbers you can seed `srand()` with `time()`.
  - Time returns the time in seconds since January 1, 1970.

The ctype Library

- `#include <ctype.h>`
- ctype is short for “character type”
- This is a special library that contains a variety of functions that deal with the char type.

Function	What it does…
`islower()`	Checks to see if character is lower case
`isupper()`	Checks to see if character is upper case
`tolower()`	Converts a character to lowercase version
`toupper()`	Converts a character to uppercase version
`isalpha()`	Checks to see if a character is an alphabet
`isdigit()`	Checks to see if a character is a digit
`isspace()`	Checks to see if a character is a whitespace
`isalnum()`	Checks to see whether a character is alphanumeric
`ispunct()`	Checks to see if character is punctuation

Revisiting the `scanf()` Function

- We know that functions return a value
- What does `scanf()` return?

```c
#include <stdio.h>

int main()
{
    int var1;
    double var2;
    int var3;

    printf("Enter 3 values:");
    int result = scanf("%d%lf%d", &var1, &var2, &var3);
    printf("result = %d\n", result);

    printf("Enter 2 values:");
    result = scanf("%d%d", &var1, &var3);
    printf("result = %d\n", result);

    return 0;
}
```

Acknowledgements

Slides adapted from Dr. Andrew Steinberg's COP 3223H course