CPS 188

Computer Programming Fundamentals

Prof. Alex Ufkes

Topic 2.2: Input, arithmetic, math.h library

Toronto Metropolitan University

Notice!

Obligatory copyright notice in the age of digital delivery and online classrooms:

Today

Intro to C continued

- User input
- Arithmetic, math library
- File I/O basics

User Input

scanf

Used to read from ***standard input*** (keyboard by default)

```
int number;
scanf("%d", &number);
```

Syntactically: Very similar to **`printf`**

<u>Two key differences:</u>

1) Do **<u>not</u>** use placeholder formatting – provide placeholders *<u>only</u>*
2) Instead of providing a variable to print, we provide the *<u>address</u>* of a variable to *<u>write</u>* to. This is done using the **&** operator.

scanf

These are two of the most common errors:

You may be tempted to do one or more of the following:

```
scanf("Please enter a number: %d", &intVar);
scanf("%d\n", &intVar);
scanf("%8.2lf", &doubleVar);
```

BAD! Use placeholder *only*

```
scanf("%d", intVar);
scanf("%lf", doubleVar);
```

BAD! Don't forget the &

Bad, But Why?

```
scanf("Enter a number: %d", &intVar);
scanf("%d\n", &intVar);
scanf("%8.2lf", &doubleVar);
```

Not a syntax error!

The content inside the quotes tells **`scanf`** what it is going to read in.

If you write **`"Enter a number: %d"`**, scanf will expect to read the statement "Please enter a number: " , followed by an integer.

Instead, do this as follows:

```
printf("Enter a number: ");
scanf("%d", &intVar);
```

GOOD!

Bad, But Why?

```
scanf("%d", intVar);
scanf("%lf", doubleVar);
```

Not a syntax error!

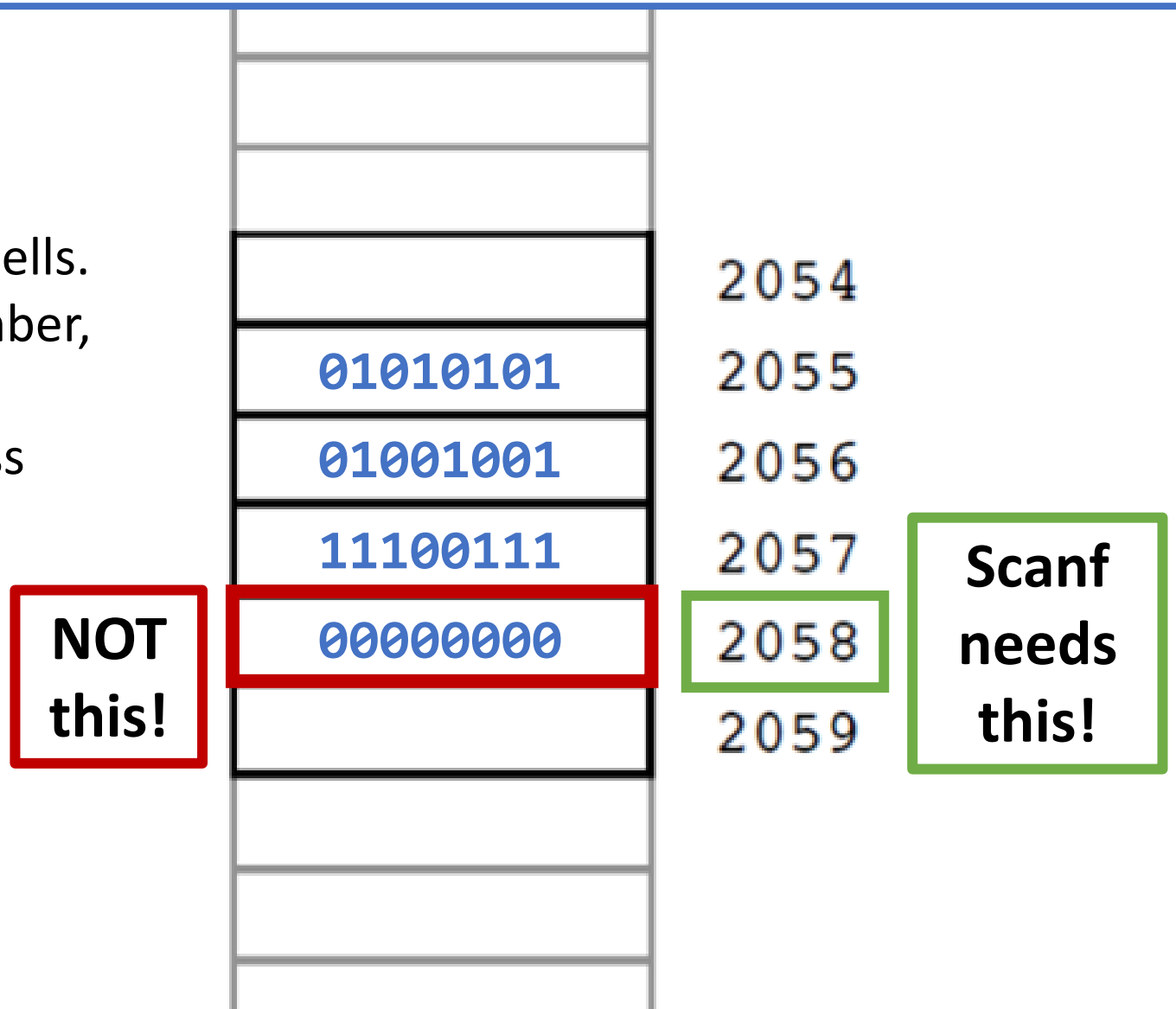
`scanf` needs an address to know where in memory it will write the value. This is what the **&** gets us.

If you do not provide the *address* of the desired variable, but rather just the *value* of the variable, your program will attempt to write the scanned value at the address equal to the value of the provided variable (almost never correct).

Address VS Value

Remember:

- Memory is a sequence of consecutively numbered cells.
- `scanf` needs the cell number, or the ***address***.
- Using **&** gets us the address



Scanning Multiple Variables?

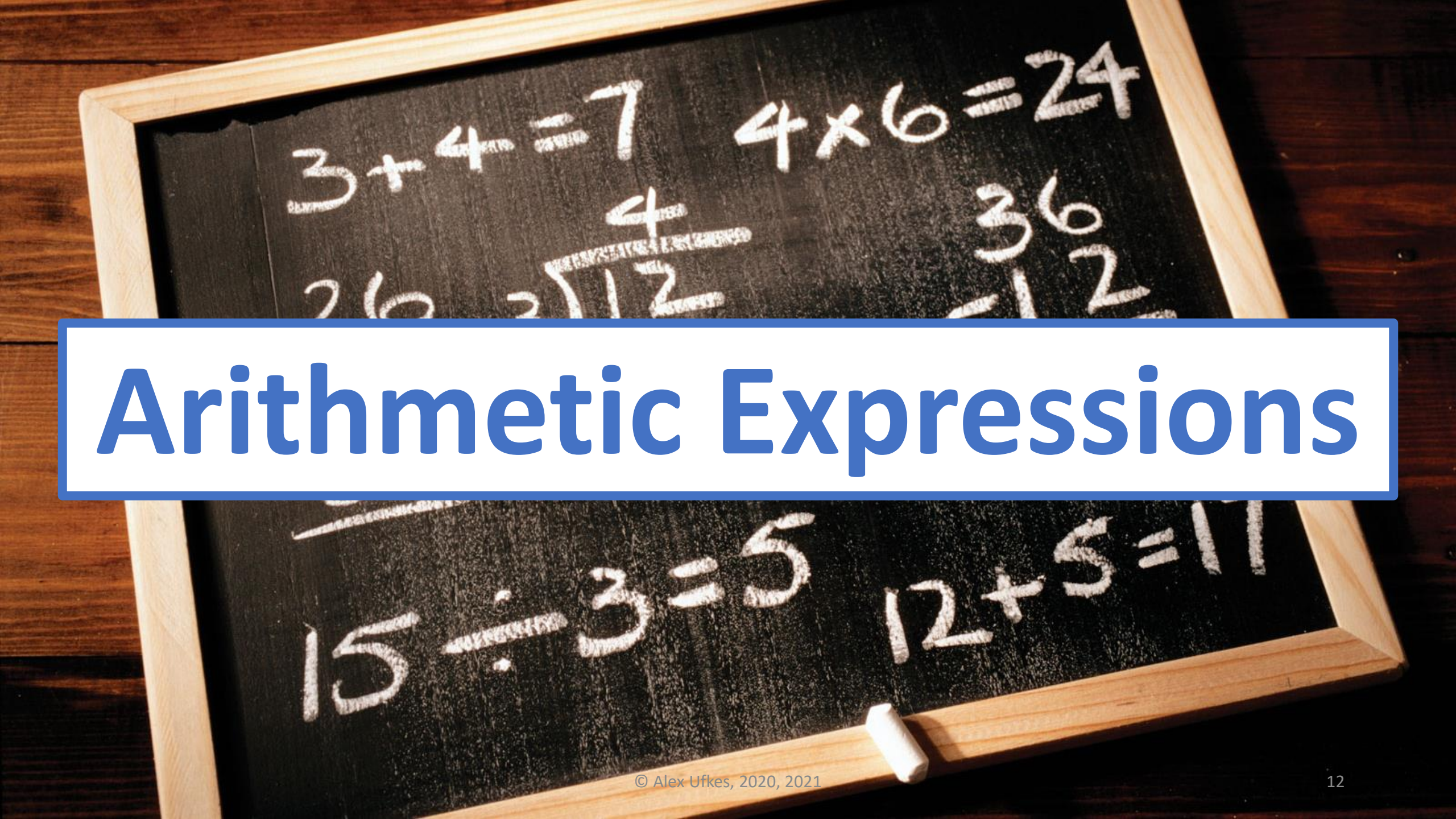
Not ideal, but can be done:

```
#include <stdio.h>

int main(void)
{
  int age;
  double height;

  printf("Enter your age and height: ");
  scanf("%d%lf", &age, &height);

  printf("You are %d years old\n", age);
  printf("You are %lf feet tall\n", height);
  return (0);
}
```

Do This Instead

```c
#include <stdio.h>

int main(void)
{
  int age;
  double height;

  printf("Enter your age: ");
  scanf("%d", &age);

  printf("Enter your height: ");
  scanf("%lf", &height);

  printf("You are %d years old\n", age);
  printf("You are %lf feet tall\n", height);
  return (0);
}
```

Arithmetic Expressions



Unary Operators

Unary: One operand

Unary Minus (-) -(-2) = 2

Important!
Stored value of x doesn't change.

```
int x = -3;
printf("x = %d\n", x);
printf("-x = %d\n", -x);
printf("x = %d\n", x);
```

Output?

```
x = -3
-x = 3
x = -3
```

Binary Operators

Binary as in two operands.

Not binary code.

Addition (+)

`3 + 4` **or** `55.1 + 43.58`

Subtraction (-)

`50 - 20` **or** `45.3 - 0.78`

Multiplication (*)

`5 * 10` **or** `0.6 * 3.4`

Division (/)

`50.0 / 2.0` **or** `45 / 2`

Only works on **integers!**

Remainder (%)

`30 % 7` **or** `45 % 3` **or** `23 % 77`

Integer Expressions

- An arithmetic expression containing only integers.
- The result of an integer expression is **always integer.**
- Non-integer results are truncated.

```
int x = 99, y = 100;
double result;
result = x/y;
printf("x/y = %lf\n", result);
```

Two integers!
Truncation occurs ***before*** being stored in result!

Output? `x/y = 0.000000`

Floating-Point Expressions

- An arithmetic expression containing only floating-point types.
- The result of a double expression is **always double.**

```
double x = 99, y = 100, result;
result = x/y;
printf("x/y = %lf\n", result);
```

Two doubles!
Result will be double.

Output? `x/y = 0.990000`

Mixed Expressions

- An arithmetic expression containing both floating-point and integers.
- The result of a mixed expression yields **floating point.**

```
int x = 5;
double y = 2.0, r1, r2;
r1 = x*y;
r2 = x/y;
printf("x*y = %lf\n", r1);
printf("x/y = %lf\n", r2);
```

Output?

```
x*y = 10.000000
x/y = 2.500000
```

Typecasting

Convert a value of one type to another:

```
int x = 99, y = 100;
double result;

result = x/y;
printf("x/y = %lf\n", result);

result = (double)x/y;
printf("x/y = %lf\n", result);
```

Output?

```
x/y = 0.000000

x/y = 0.990000
```

Be Careful!

```
int x = 99, y = 100;

printf("%lf\n", x/y);
printf("%lf\n", (double)x/y);
printf("%lf\n", x/(double)y);
printf("%lf\n", (double)(x/y));
```

Output?

```
0.000000
0.990000
0.990000
0.000000
```

- Parentheses are evaluated **first**!
- Result gets truncated ***before*** the cast occurs!

Casting

```
int x = 99, y = 100;
double r;

x = (double)x;
r = x/y; /* is x double? */

printf("%lf\n", r);
```

Output?

0.000000

Casting does **NOT** change the data type, only the value.

The value conforms to the container (data type) it is stored in.

Problem?

Quincy 2005 - [test2.c]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>
#define PI 3.14159

int main(void)
{
    double vol, r, h;

    printf("Enter radius: ");
    scanf("%lf", &r);
    printf("Enter height: ");
    scanf("%lf", &h);

    vol = PI*r*r*(1/3)*h;
    printf("Volume is %lf\n", vol);

    return (0);
}
```

Press F1 for help | Ln 18, Col 1 | NUM

quincy

```
Enter radius: 5
Enter height: 5
Volume is 0.000000

Press Enter to return to Quincy...
```

Why?

Problem?

Quincy 2005 - [test2.c]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>
#define PI 3.14159

int main(void)
{
    double vol, r, h;

    printf("Enter radius: ");
    scanf("%lf", &r);
    printf("Enter height: ");
    scanf("%lf", &r);

    vol = PI*r*r*(1/3)*h;
    printf("Volume is %lf\n", vol);

    return (0);
}
```

Press F1 for help Ln 18, Col 1 NUM

Output? **ALWAYS 0**

1/3 = 0

Possible fixes:

```c
vol =
  PI*r*r*(1.0/3)*h;
  PI*r*r*(1/3.0)*h;
  PI*r*r*(1.0/3.0)*h;
  PI*r*r*((double)1/3)*h;
  PI*r*r*(1/(double)3)*h;
  PI*r*r*(h/3);
```

Order of Operations

1. Parentheses **( )**: Evaluated first (inside out)
2. Operator precedence: **unary** `->` `*,/,%` `->` `+,-`
3. Associativity: When on the same precedence level, binary operators evaluate left to right.

When in doubt, use more parentheses! ( )

Order of Operations

Evaluate the following:

```
z - (a + b / 2) + w * -y
```

1. **Parentheses:** `b/2` **then add** `a`
2. **Unary:** `-y`
3. **Multiplication:** `w * -y`
4. **Subtraction:** `z - (1.)`
5. **Addition:** `4. + 3.`

Shortcut Operators

```
i += 1;      /* i = i + 1; */
a /= 2;      /* a = a / 2; */
x *= 5;      /* x = x * 5; */
   i++;      /* i = i + 1; */
   ++i;      /* i = i + 1; */
   i--;      /* i = i - 1; */
   --i;      /* i = i - 1; */
```

Careful!

`r = ++x;` **NOT** the same as `r = x++;`

```
int x = 2, y = 2, r1, r2;
r1 = ++x;
r2 = y++;
printf("pre = %d\n", r1);
printf("post = %d\n", r2);
```

Output?

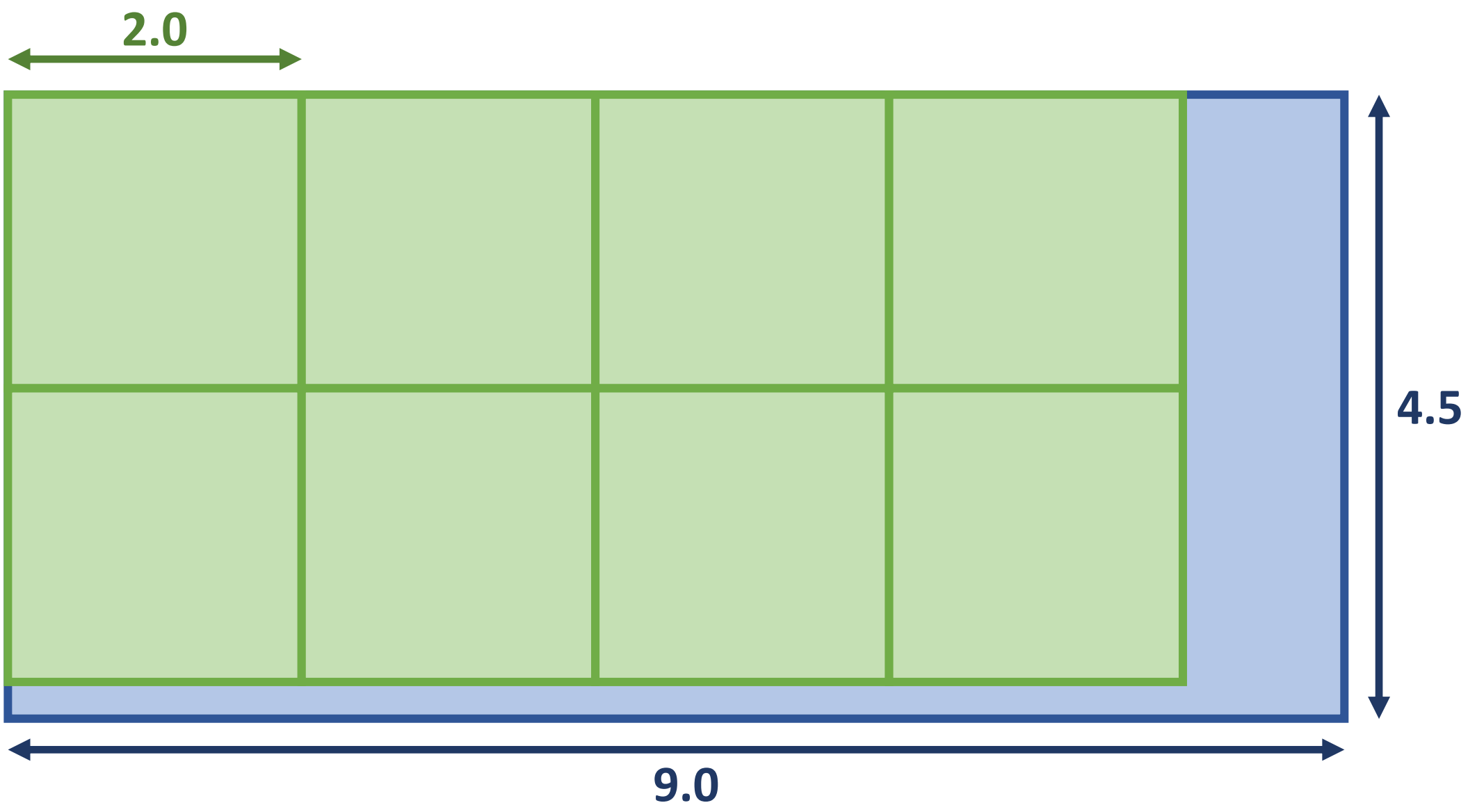
```
pre = 3
post = 2
```

***Pre*-increment:** increment, then assignment.

***Post*-increment:** assignment, then increment.

Practice Example #1

- Write a C program that asks the user to enter two double values representing the length and width of a rectangle.
- Ask the user to enter another double, representing the side length of a square.
- Calculate how many whole squares will fit into the rectangle.



#include <math.h>

math.h

- **`math.h`**, like **`stdio.h`**, is part of the C standard library.
- We include it in the same way using a preprocessor directive:

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

Recall:

- Including **`stdio.h`** lets us use various I/O functions
- We've seen two so far: **`printf, scanf`**

- Including **`math.h`** gets us many useful math functions:
- **`sqrt, log, exp, pow, sin, cos, tan,`** and more.

A Quick Word on Functions

A ***function*** is a programming construct that ***encapsulates*** some code

Every C program must have a main function, as we've seen:

```
#include <stdio.h>
int main (void)
{
    /* your program goes here */
}
```

`printf` & **`scanf`** are also functions!

Functions as Black Boxes

- It's common to use functions as *black boxes*.
- We know what goes in, we know what comes out, we don't know what goes on inside.

`printf? scanf?`

- We know how to ***use*** them, we don't know how they ***work***.
- That's OK! We can use the tools the language gives us.

The `printf` Function

A function can be called inside another function.
We call the **`printf`** function from inside our **`main`** function:

```
#include <stdio.h>
int main (void)
{
?!? int x = printf("Hello, world!\n");
}
```

Function **output**

Function **identifier**

Function **input**

The `printf` Function

Quincy 2005 - [main.c]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>

int main(void)
{
    int x = printf("Hello, world!\n");
    printf("%d chars printed\n", x);
    return (0);
}
```

Press F1 for help | Ln 5, Col 36 | NU

quincy

```
Hello, world!
14 chars printed

Press Enter to return to Quincy...
```

- We don't typically care about the return value of **`printf`**
- It is almost never stored (omit the `int x =`)
- Confused? Don't worry. Lots more on functions later in the course.

Math Functions

```
double a, b, c, x, y, z, n = 3.1415;

a = sin(n);
b = cos(n);
c = tan(n);
```

Output Input

```
x = asin(n);  /* arc sine, inverse of sine */
y = acos(n);  /* arc cosine                */
z = atan(n);  /* arc tangent               */
```

More Math Functions

```
double a, b, c, x, y, z, n = 3.1415, p = 2.5;

a = floor(n);  /* largest whole num <= n   */
b = ceil(n);   /* smallest whole num >= n  */

c = fabs(n);   /* absolute value of n      */

x = sqrt(n);   /* x = the square root of n */
y = log(n);    /* natural log ( ln() )     */
z = log10(n);  /* log base 10              */

x = exp(n);    /* x = e^n, e = 2.71828     */
y = pow(n, p); /* x = n^p                  */
```

Things to Know

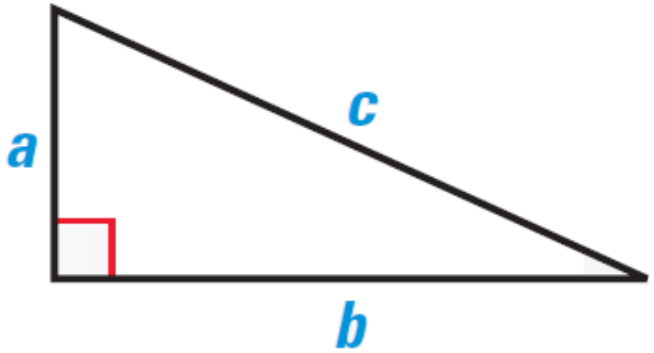
Math functions in `math.h` use and return doubles

```
double a, b, c, x, y, z, n = 3.1415, p = 2.5;

x = sqrt(n);   /* x = the square root of n  */
y = log(n);    /* natural log ( ln() )      */
z = log10(n);  /* log base 10               */
```

This is sensible! Square roots, logarithms, trig functions are almost always going to result in floating-point numbers.

Practice Example #2

- Write a program that asks the user to enter two sides of a right triangle.
- Use the *Pythagorean Theorem* to calculate the Hypotenuse. Print the result.
- Also calculate and print the perimeter and the area of the triangle.

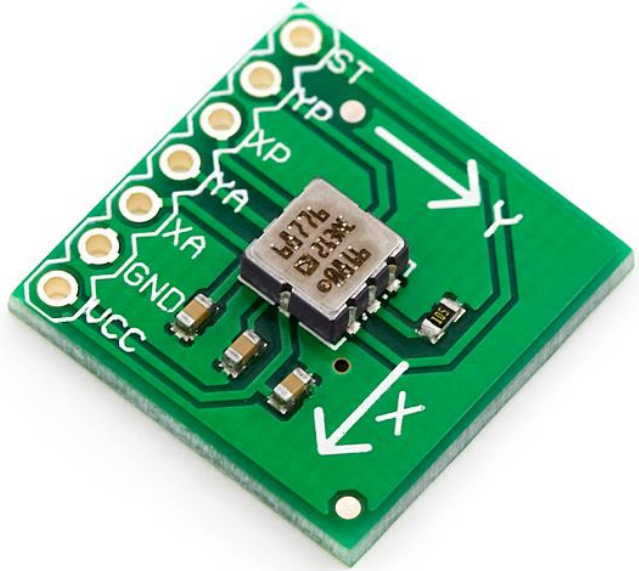


$$c^2 = a^2 + b^2$$

File I/O

It’s hard to do any meaningful data processing if all your input is coming from the keyboard via the user.

In practice, it’s far more common that input data for our programs will come from hardware (cameras, accelerometers, other sensors) or files (data logs recorded previously)



fscanf

Quincy 2005 - [fileIO]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>

int main(void)
{
    int tmp;

    /* file handle variable */
    FILE *in;

    /* open file */
    in = fopen("mydata.txt", "r");

    /* scan an integer from the file */
    fscanf(in, "%d", &tmp);

    /* print value read from file */
    printf("From file: %d\n", tmp);

    /* close the file */
    fclose(in);

    return 0;
}
```

Press F1 for help Ln 8, Col 14

New variable type: FILE *

fopen to open the file. First argument is the filename, second is the mode. **"r"** means ***read***.

fscanf - Like scanf but takes an argument before the string: the file handle variable.

fclose – File should be closed once we're done with it.

Practicalities

Quincy 2005 - [fileIO]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>

int main(void)
{
    int tmp;

    /* file handle variable */
    FILE *in;

    /* open file */
    in = fopen("mydata.txt", "r");

    /* scan an integer from the file */
    fscanf(in, "%d", &tmp);

    /* print value read from file */
    printf("From file: %d\n", tmp);

    /* close the file */
    fclose(in);

    return 0;
}
```

Press F1 for help | Ln 8, Col 1

mydata - Notepad

File Edit Format View Help

```
10 13 7 9 -52
```

Delete Rename New folder Properties Open Edit History Select all Select none Invert selection

Organize New Open Select

Search Code

Date modified	Type	Size
1/18/2017 4:02 PM	C File	1 KB
1/18/2017 4:02 PM	Application	19 KB
1/18/2017 4:02 PM	O File	3 KB
1/18/2017 4:25 PM	C File	1 KB
1/18/2017 4:25 PM	Application	19 KB
1/18/2017 4:25 PM	O File	3 KB
1/18/2017 11:54 AM	C File	1 KB
1/18/2017 9:33 AM	Application	18 KB
1/18/2017 4:20 PM	TXT File	0 KB
11/16/2016 11:04 ...	C File	3 KB
11/16/2016 11:04 ...	Application	21 KB
11/16/2016 11:04 ...	O File	5 KB

Downloads, CKRE110, Code, Code, CPS125, OneDrive, This PC, Desktop, Documents, Downloads

Declarations, declarations.o, fileIO, fileIO, fileio.o, HelloWorld, HelloWorld, mydata, Text1, Text1, text1.o

12 items | 1 item selected 0 bytes

fprintf

Quincy 2005 - [fileIO]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>

int main(void)
{
    int tmp;

    /* file handle variable */
    FILE *in;

    /* open file */
    in = fopen("mydata.txt", "w");

    /* scan an integer from the file */
    fprintf(in, "%d %d %d", 37, 52, -9);

    /* close the file */
    fclose(in);

    return 0;
}
```

Press F1 for help Ln 16, Col 5

Must first open the file:

- This time we're ***writing***. **"w"** means write.
- ***If the file already exists, it will be overwritten!***

fprintf - Like printf but takes an argument before the string: the file handle variable.

fclose – File should be closed once we're done with it.

Quincy 2005 - [fileIO]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>

int main(void)
{
    int tmp;

    /* file handle variable */
    FILE *in;

    /* open file */
    in = fopen("mydata.txt", "w");

    /* scan an integer from the file */
    fprintf(in, "%d %d %d", 37, 52, -9);

    /* close the file */
    fclose(in);

    return 0;
}
```

Press F1 for help Ln 16, Col

mydata - Notepad

File Edit Format View Help

37 52 -9

Open Edit History Properties Open

Search Code

Quick access
GoogleDrive
Downloads
CKRE110
Code
Code
CPS125
OneDrive
This PC
Desktop
Documents
Downloads

Name	Date modified	Type
Declarations	1/18/2017 4:02 PM	C File
Declarations	1/18/2017 4:02 PM	Application
declarations.o	1/18/2017 4:02 PM	O File
fileIO	1/18/2017 4:25 PM	C File
fileIO	1/18/2017 4:25 PM	Application
fileio.o	1/18/2017 4:25 PM	O File
HelloWorld	1/18/2017 11:54 AM	C File
HelloWorld	1/18/2017 9:33 AM	Application
mydata	1/18/2017 4:20 PM	TXT File
Text1	11/16/2016 11:04 ...	C File
Text1	11/16/2016 11:04 ...	Application
text1.o	11/16/2016 11:04 ...	O File

12 items 1 item selected 0 bytes

File I/O

- There are more interesting programs we could write that use file I/O.
- But we need more C knowledge and experience first.


YOU'RE DOING
IT WRONG
Spot the error

```
#include <stdio.h>

int main (void)
{
      printf ("Hello world.\n");

      int y = 0;      /* variable declarations
      y = y + 1;      typically go at the top */

      return (0);
};
    /* semi-colon not needed
    here, but not illegal */
```

```
#include <stdio.h>

#define X          /* Empty Macro definition.
                   Weird, but not an error */

int main(void)
{
    printf("Hello world.\n");

    return (0);
}

X   /* X is replaced
    with nothing. */
```

```
#define PI 3.141516;

int main (void)
{
        double x, y = PI;

        x = x;

        z = y + PI;        /* Undeclared
                              variable */

        return (0);
}
```

```
#include <stdio.h>

int main (void)
{
      ;
      /* Nested /*comment */ */

      printf ("Hello world.\n");

      return (1);
}
```

```
#include <stdio.h>

int main (void)
{
      int a     = 1 ;

      printf("What is your age? ")
;
      A=37;                    /* Undeclared
                               variables */
      y =          a;

      printf("%d\n", a);

      return 0;
}
```

```c
#include <stdio.h>

int main (void)
{
      int x;

      1 = x;   /* Illegal assignment */

      fprintf("Hello world.\n");

      return (0);  /* Improper use
                      of fprintf */
}
```

```
#include <stdio.h>

int main (void)
{
        int a, b, c, d;      /* Redeclaration
        float d, e, f, g;    error */

        printf("Hello world.\n");

        return (0);
}
```

```
#include <stdio.h>

int main (void)
{
      int x,

                        y = 2;

    y+1;  /* Should be double quotes */

      printf('What is your age?\n');
      scanf('%f', x);/* Missing & */

     /* Type mismatch */

      return (0);
}
```

Questions?