COMP1911 - Computing 1A

Dr Binghao Li
School of Minerals and Energy Resources Engineering
School of Computer Science and Engineering
binghao.li@unsw.edu.au

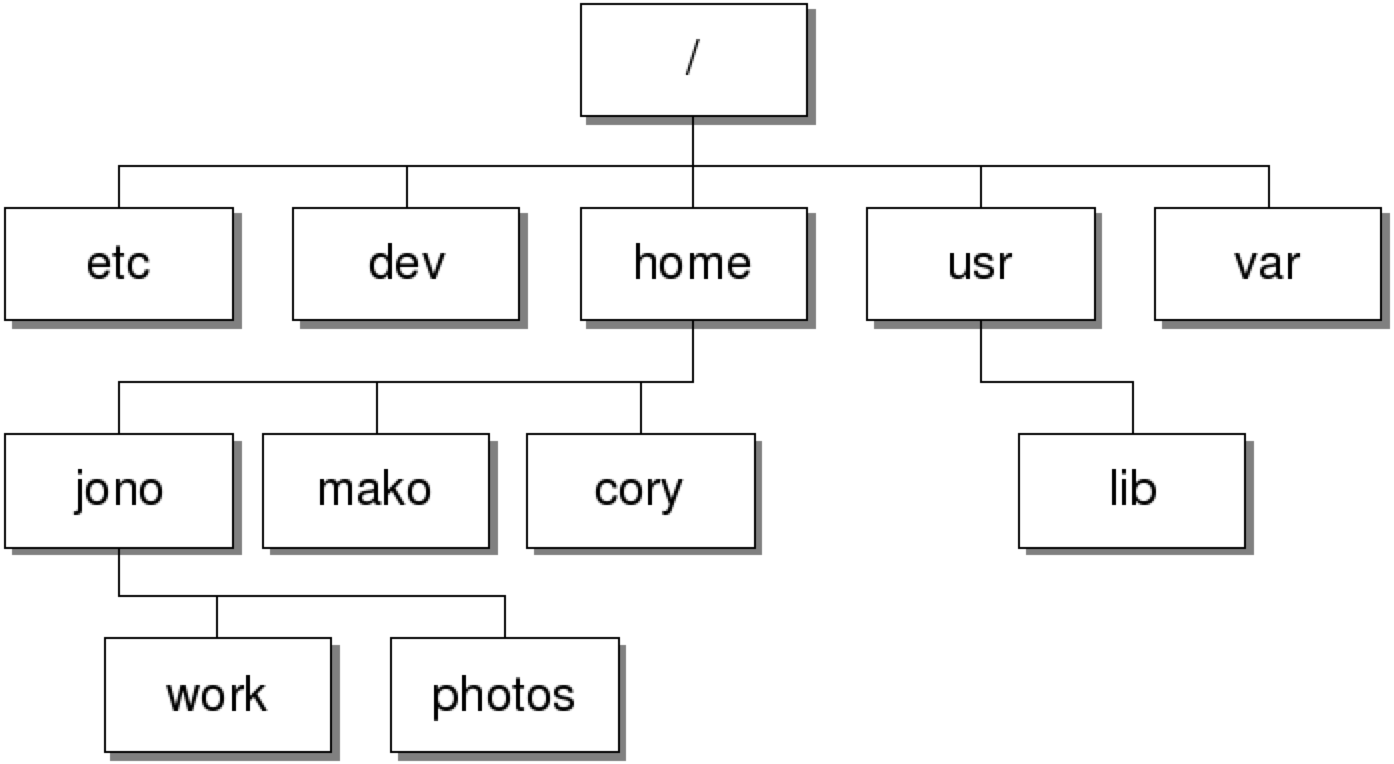
June 2022

3. C Basics

In this lecture we will cover:

- (more) linux terminal commands
- Variables and variable names
- Scanf
- Constants
- Arithmetic operations

Linux File System Structure



Common Linux Commands

- ls, ls -l, ls -a, ls -la
- cd, cd .., cd , cd ~, cd /
- mkdir
- pwd
- mv [options] source destination
  - ➢ Rename or move file(s) or directories.
- rm [options] directory
  - ➢ Remove (delete) file(s) and/or directories.
- who [options]
  - ➢ Display who is logged on

Variables

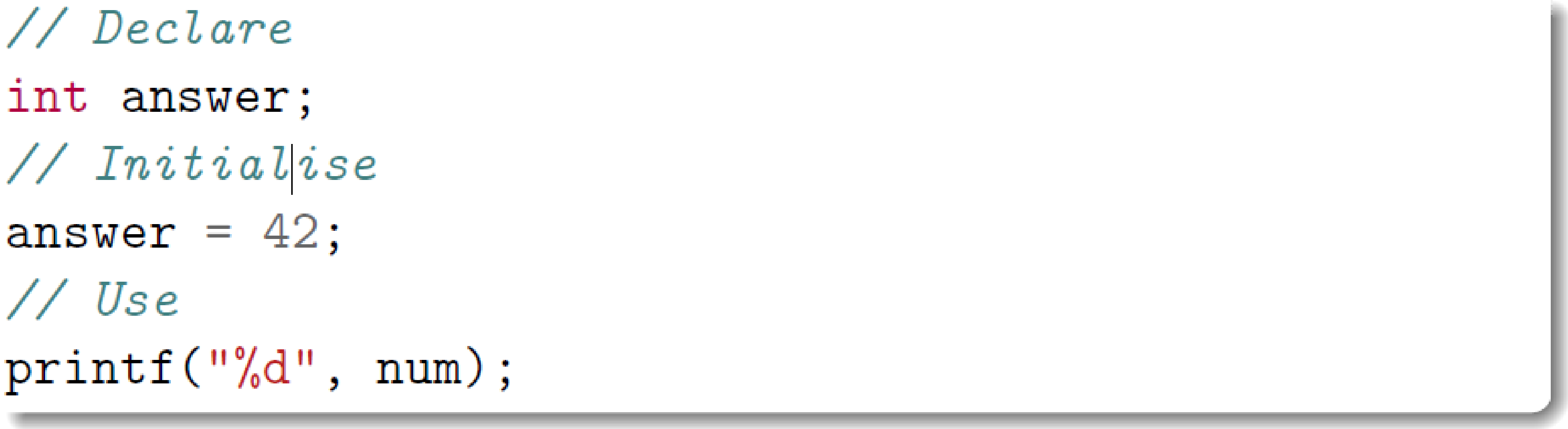
- Variables are used to store a value.
- The value a variable holds may change over its lifetime.
- At any point in time a variable stores one value.
- C variables have a type.
- A variable is stored in a known-part of RAM that is allocated to the program.

To start with, we will only consider 2 types of variables:

- **int** for integer values, e.g.: 42, -1
- **double** for decimal numbers 3.14159, 2.71828

Variables

- **Declare** The first time a variable is mentioned, we need to specify its type. This tells C it needs to set aside a chunk of memory (RAM) for the variable.
- **Initialise** Before using a variable we need to assign it a value. Before we do this, the memory location just contains whatever 'garbage' values that happened to be there before.



```
// Declare
int answer;
// Initialise
answer = 42;
// Use
printf("%d", num);
```

Variable Names (and other Identifiers)

- Variable names can made up of letters, digits and underscores
- Beware variable names are case sensitive,

  e.g. **hello** and **hEllo** are different names
- Beware certain words can't be used as variable names:

  e.g.: **if**, **while**, **return**, **int**, **double**

These **keywords** have special meanings in C programs.

You'll learn what many of them are as we go on.

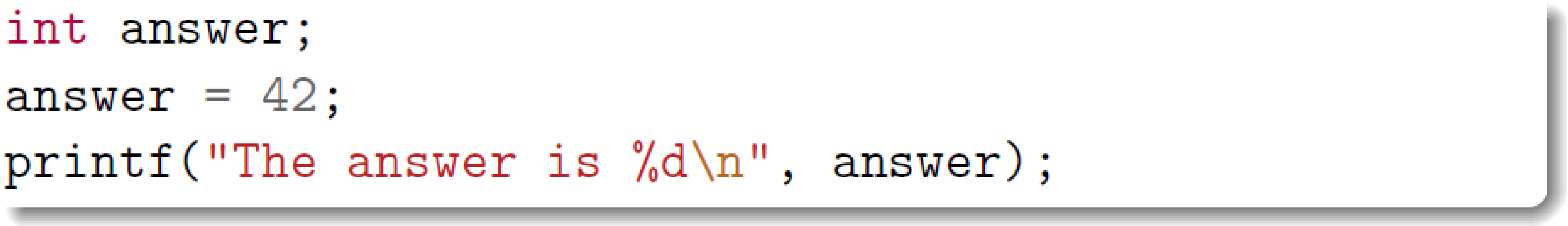
Variable Names (and other Identiers)

In this course we must follow the Style Guide http://cgi.cse.unsw.edu.au/~cs1911/22T2/style-guide/index.html, which is more restrictive:

- They must be valid C identifiers
- They must begin with a lower case letter
- They must not use any underscore characters
- identifier names should be meaningful
- letter variables should be avoided unless they are loop counters or numbers from a maths formula
- where identifier names are composed of several words, the first word should be in lower case and the first letter of each subsequent word should be in upper case
  - eg myFirstVariable
  - We call this camelCase

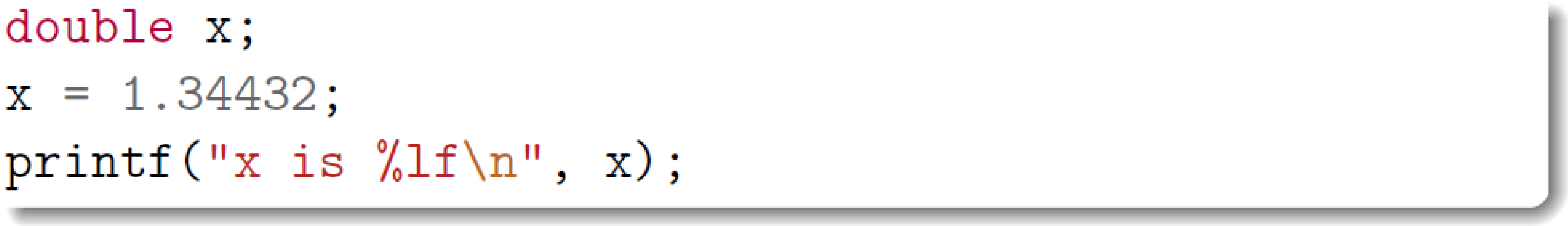
Using values in printf()

- Use conversion specifier **%d** to print an **int** (integer) value



```c
int answer;
answer = 42;
printf("The answer is %d\n", answer);
```

- Use conversion specifier **%lf** or **%f** to print a **double** (floating point) value

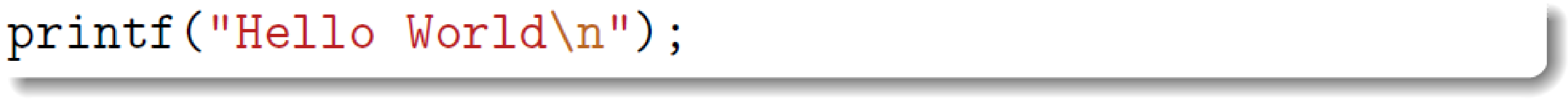


```c
double x;
x = 1.34432;
printf("x is %lf\n", x);
```

In addition, most conversion specifiers have options for finer control, e.g., %2.3lf instructs printf to use a precision of three.

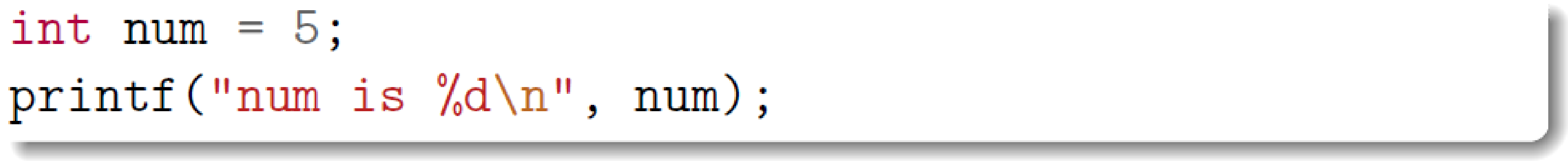
Output using printf()

- No variables:



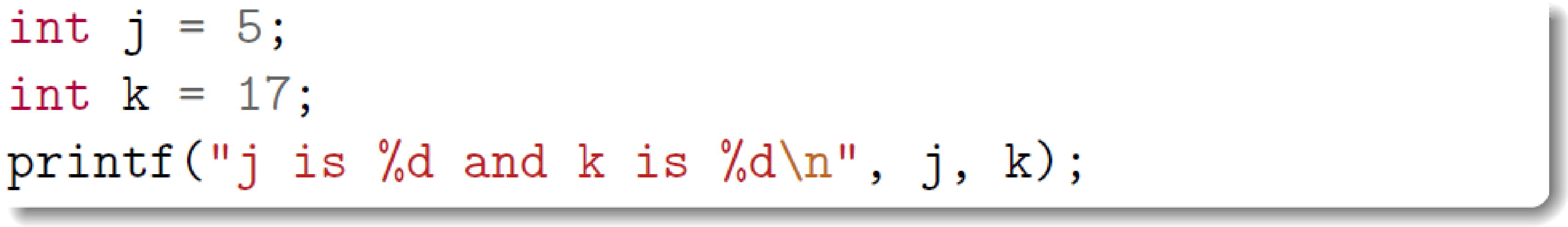
```c
printf("Hello World\n");
```

- A single variable:



```c
int num = 5;
printf("num is %d\n", num);
```

- More than one variable:

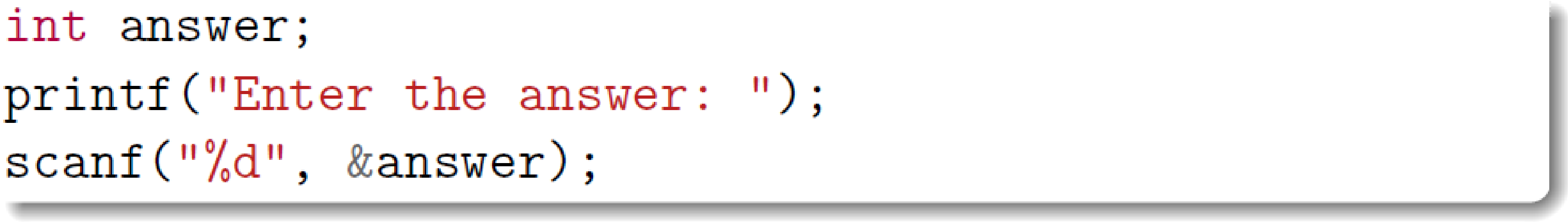


```c
int j = 5;
int k = 17;
printf("j is %d and k is %d\n", j, k);
```

Input using scanf()

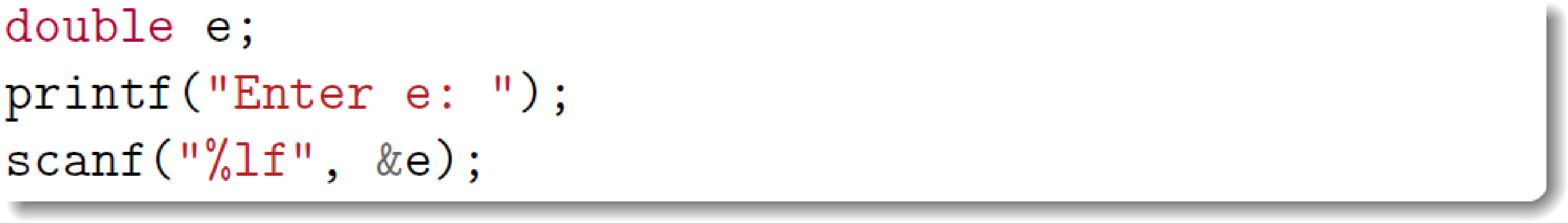
scanf uses a format string like printf.

- Use **%d** to read an **int** (integer) value

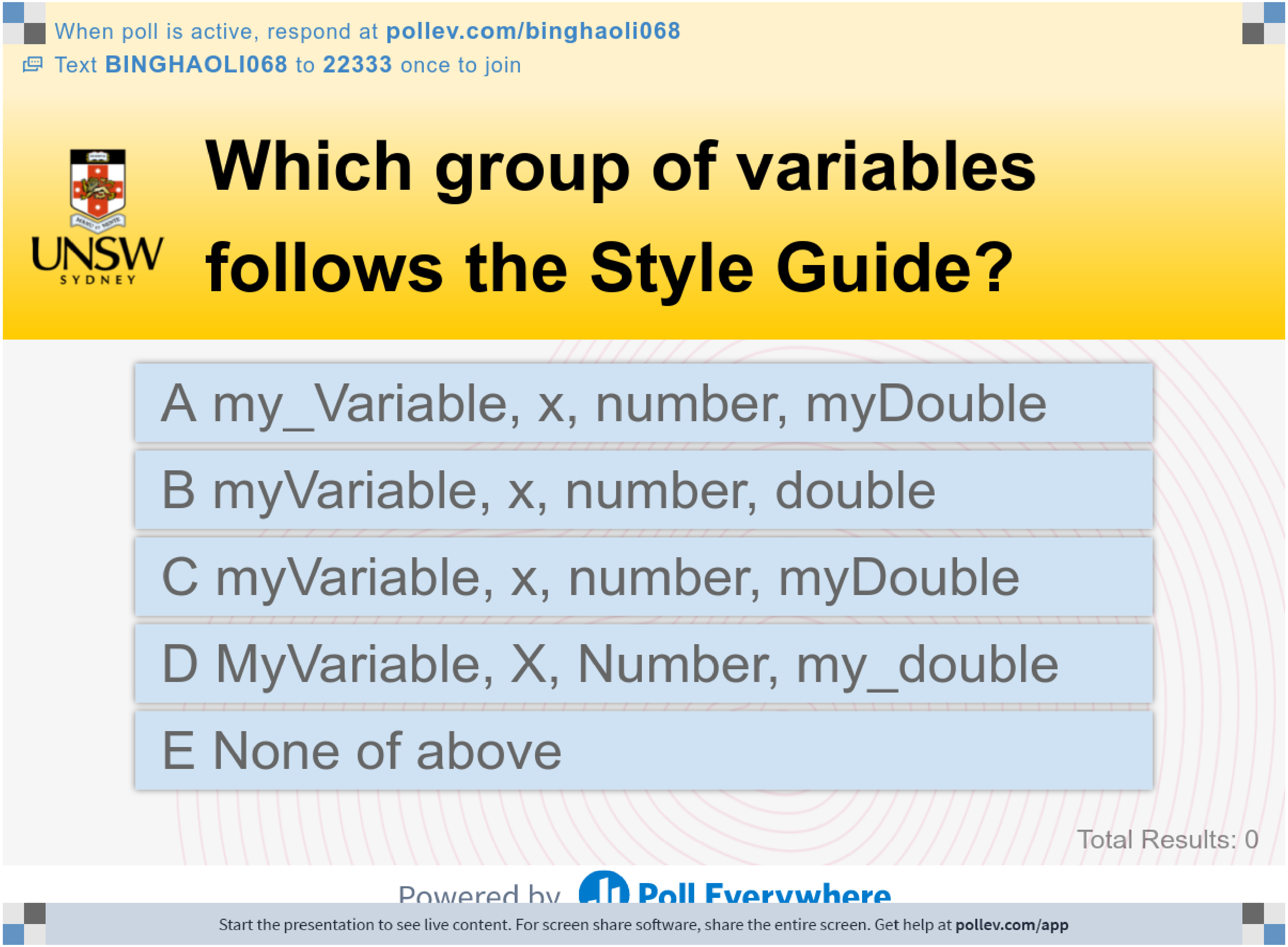


```c
int answer;
printf("Enter the answer: ");
scanf("%d", &answer);
```

- Use **%lf** to read a **double** (floating point) value



```c
double e;
printf("Enter e: ");
scanf("%lf", &e);
```



When poll is active, respond at **pollev.com/binghaoli068**

Text **BINGHAOLI068** to **22333** once to join

Which group of variables follows the Style Guide?

A my_Variable, x, number, myDouble

B myVariable, x, number, double

C myVariable, x, number, myDouble

D MyVariable, X, Number, my_double

E None of above

Total Results: 0

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Numbers and Types

- Numbers in programs have types.
- Numbers with a decimal point are type **double**, e.g. 3.14159 -34.56 42.0
- C also lets write numbers in scientific notation: 2.4e5 => 2.4×10^{5} => 240000.0
- Numbers in scientific notation are also type **double**
- Numbers without decimal point or exponent are type **int**, e.g. 42 0 -24
- Numbers in programs are often called constants (unlike variables they don't change)

Giving Constants Names

- It can be useful to give constants (numbers) a name.
- One method is **#define** statement e.g.

```
#define SPEED_OF_LIGHT 299792458.0
#define MIN_PER_HOUR 60
```

- It often makes your program more readable.
- It also makes it harder for programmers to accidentally change the value
- It can make your program easier to update, particularly if the constant appears in many places
- **#define** statements go at the top of your program after **#include** statements
- For good style, **#define** names should be all capital letters + underscore.

Arithmetic Operators

- C supports the usual maths operations: + - * /
- Precedence is as you would expect from high school, e.g.: a+b*c+d/e => a+(b*c)+(d/e)
- What is the value of the following expression? 1+2*3-2/2
- Associativity (grouping) is as you would expect from high school, e.g.: a-b-c-d => ((a-b)-c)-d
- What is the value of the following expression? 7-4+3
- Use brackets if in doubt about order arithemtic will be evaluated.
- Beware division may not do what you expect.

Division in C

- C division does what you expect if either operand is a **double** the result is a **double.**

  2.6/2 => 1.3

- C division may not do what you expect if both arguments are integers.
- The result of dividing 2 integers in C is an integer.
- The fractional part is discarded (not rounded!).

  5/3 =>1 (not 2)

- C also has the **%** operator (integers only), computes the modulo (remainder after division)

  14 % 3 => 2

Exercise

Discuss with your class mate or think about it yourself

What are the values of the following expressions?

6*7-8*9/10

2*3*4+5*6

5*6/4

3/2

1.0/2.0

1/2.0

What are the values of the following expressions? 6*7-8*9/10, 5*6/4, 3/2, 1.0/2.0, 1/2.0

34.8, 7, 1, 0.5, 0

35, 7, 1, 0.5, 0.5

34, 8, 2, 0, 0

34, 7, 2, 0.5, 0.5

Total Results: 0

Mathematical functions

- Mathematical functions not part of standard library essentially because tiny CPUs may not support them
- A library of mathematical functions is available including: sqrt(), sin(), cos(), log(), exp()
- Above functions take a double as argument (input) and return a double (output)
- //Example usage

```
double result = sqrt(1.5);
```

- Extra include line needed at top of program:

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

- dcc includes maths library by default

  most compilers need extra option:

  gcc needs **-lm** e.g.:

```
gcc -Werror -Wall -O -o circle circle.c -lm
```

GCC Compiler Command Line Options

We have basic C code named main.c

Basic form

```
gcc main.c
```

Specify the output file name for the executable

```
gcc main.c -o main
```

Enable all warnings set

```
gcc -Wall main.c -o main
```

Convert warnings into errors

```
gcc -Werror main.c -o main
```

Optimize, the compiler tries to reduce code size and execution time

```
gcc -Werror -Wall -O main.c -o main
```

Questions