IT6008 Computer Programming

بوليتكنك البحرين

Bahrain Polytechnic

Unit 2: Algorithms, Variables and Data types

IT6008 Computer Programming

بوليتكنك البحرين

Bahrain Polytechnic

Unit 2 – Part 1: Problem solving and Algorithms

Assumed Knowledge

- Before the beginning of this lecture, students should:
  - be familiar with the terms *compiler*, *program*, and *machine code*;
  - have reviewed some simple Java programs from the Module 1 materials and Lewis et. al. Chapter 1

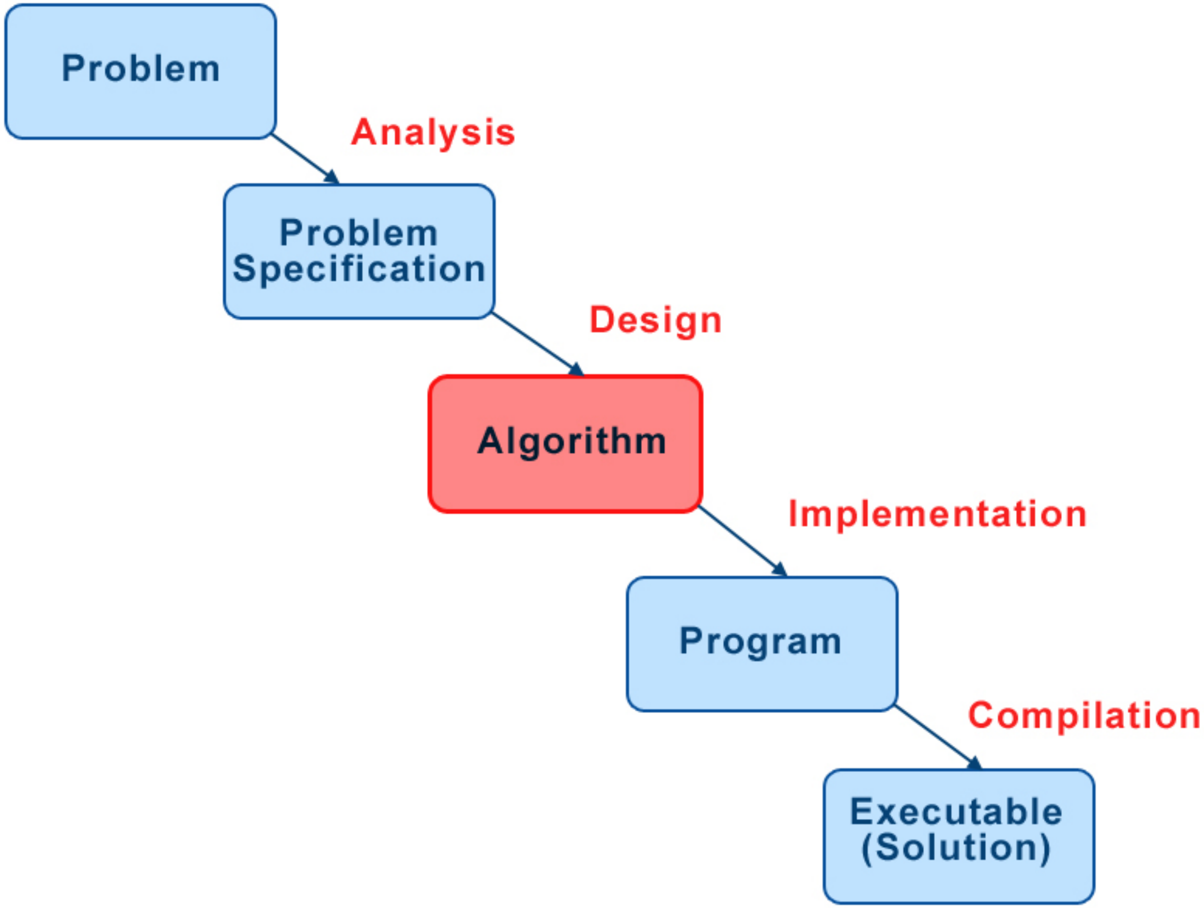
- Before the beginning of this lecture, students should have read over:

  > Chapter 2, sections 2.1 and 2.2 of Lewis et. al.

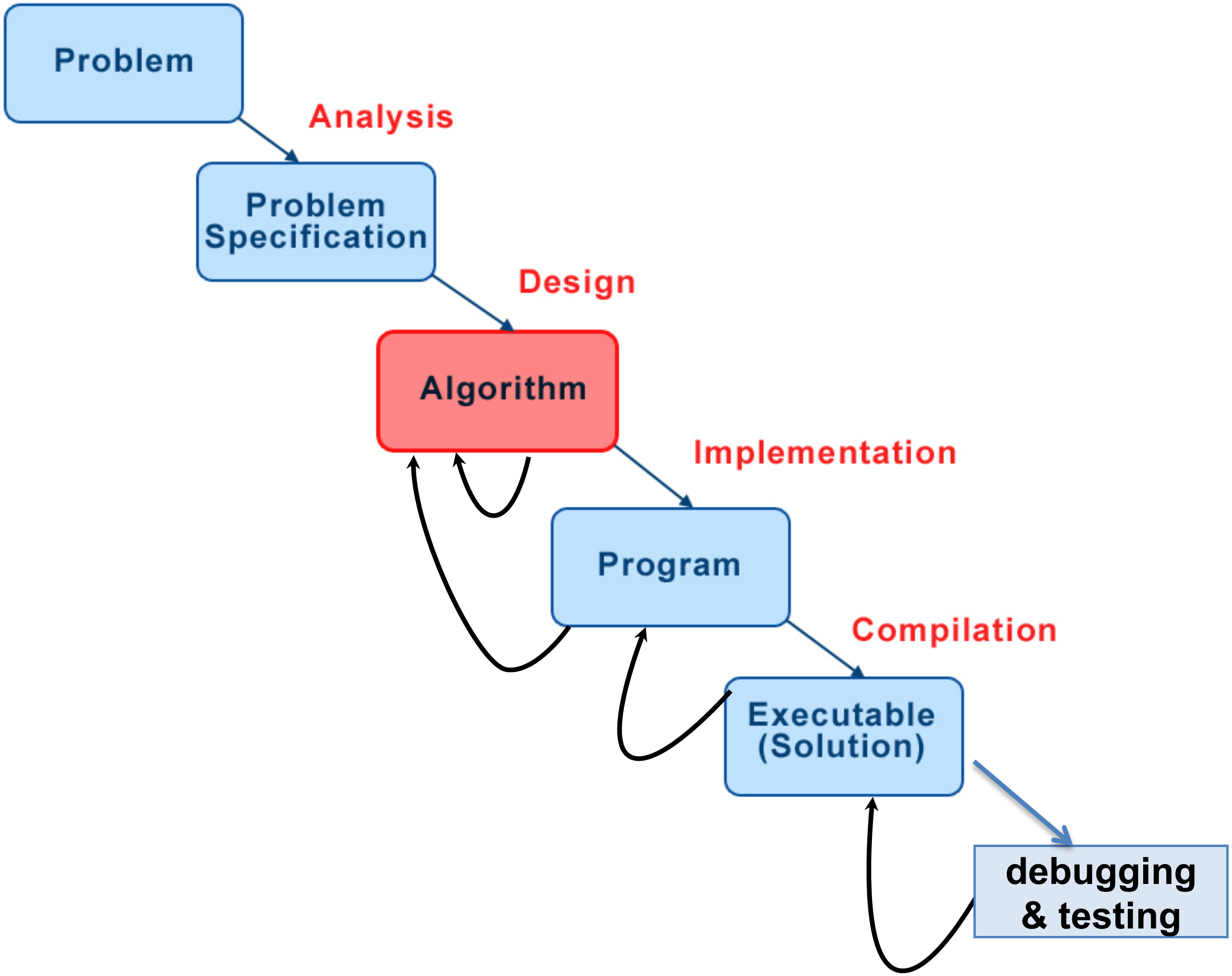
Algorithms: Objectives

- Understand the role programming plays in the problem-solving process
- Understand the concept of an algorithm, and the relationship between algorithms and programs
- Be able to write basic algorithms for everyday tasks
- To understand the purpose of variables
- To understand the difference between variables and constants

The Problem Solving Process



The Problem Solving Process is *not* linear!



Problem

Analysis

Problem Specification

Design

Algorithm

Implementation

Program

Compilation

Executable (Solution)

debugging & testing

Algorithm

algorithm |ˌalgərɪð(ə)m|

noun

a process or set of rules to be followed in calculations or other problem-solving operations, esp. by a computer.

The term is named after *Muhammad ibn Musa* *al-Khwarizmi* of Khowarezm (now Khiva in Uzbekistan), Circa 780-850 C.E.

Algoritmi is latinised *al-Khwarizmi*

lska
land
Беларусь
Belarus
Україна
Ukraine
Romania
Kazakhstan
A
Uzbekistan
Kyrgyzstan
Ελλάς
Greece
Türkiye
Turkey
Turkmenistan
Syria
Iraq
Iran
Afghanistan
Pakistan
Nepal
bya
Egypt
Saudi Arabia
India
Oman
had
Sudan
Yemen
Ethiopia

Algorithm - Examples

- "A list of well-defined instructions for completing a task"
  - A cooking recipe
  - The rules of how to play a game
  - Instructions on how to operate a piece of equipment
  - Directions for driving from A to B
  - A knitting pattern
  - A car repair manual

Algorithms

- An algorithm is a well-defined sequence of unambiguous instructions
- Should terminate (to produce a result)
- Algorithm description must be precise, using terms and language that will be understood exactly by whoever is going to read it (according to convention).
- It may contain English text and mathematical expressions, provided these are used clearly and precisely.

Example: Manual Addition…

Describe the method to your friend... you have just recited an algorithm!

$$\begin{array}{r} 123456 \\ +\ 789001 \\ \hline 912457 \end{array}$$

Almond and Honey Slice

1/2 quantity *Shortcrust Pastry*

185 g unsalted butter

100 g castor sugar

5 tablespoons honey

50 ml cream

50 ml brandy or any other liqueur or spirit

300 g flaked almonds

1. Preheat oven for 200° C
2. Line a 30 cm × 20 cm baking tray with baking paper, and then with pastry
3. *Bake blind* for 20 minutes, then remove weights and foil
4. Turn oven up to 220° C.
5. Bring remaining ingredients to a boil, stirring.
6. Spread evenly over pastry.
7. Bake until topping is bubbling and has caramelised evenly, about 15 minutes.
8. Cool before cutting into fingers or squares.

Instructions are given in the **order** in which they are performed ("executed")

Algorithms are not Programs

An algorithm describes how a computer or human may do a task but,

…it cannot be *executed* by a computer.

... it is for a *human* to read.

... it must be *implemented* (coded), to make a *program*.

Computers can only execute programs.

Always design the algorithm **before** you start to program.

The “idea” comes before the coding.

Algorithms and Languages

- The very same Algorithm may look very different when described in different languages

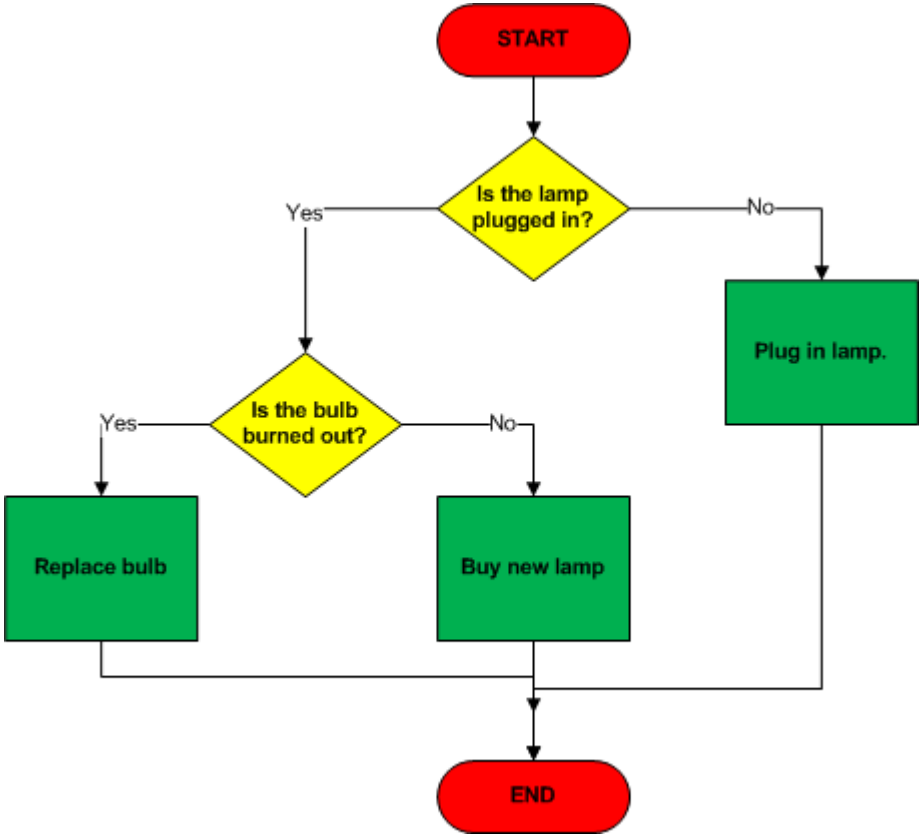
“Sum the numbers between 1 and 10

by adding each of them to a total starting with 0”

s ← +/(ι10)	APL-style
int s:=0; for i = 0 to 10 do s:= s+i; end	Pascal-style

Algorithms

- Can be expressed as a flowchart diagram



Components of an Algorithm

- Instruction (also known as primitive)
- Sequence (of instructions)
- Selection (used to decide which instructions to execute –Unit 4)
- Repetition (of instructions – Unit 5)

Instructions (Primitives)

An instruction must be simple and unambiguous.

The system must be capable of performing it.

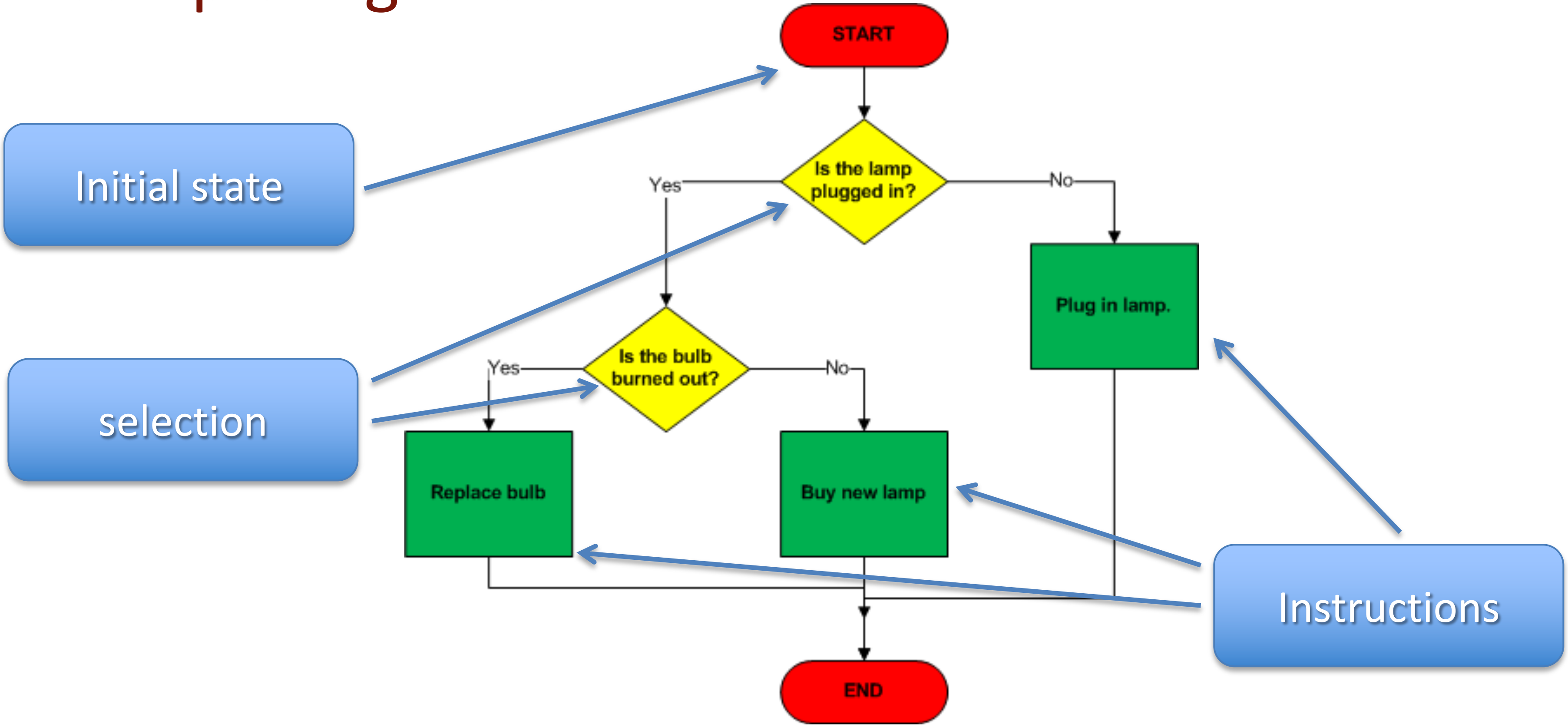
Examples:

Take off your shoes
Count to 10
Cut along dotted line
Knit 1
Purl 2
Pull rip-cord firmly
Sift 10 grams of flour

Directions to perform specific actions

Algorithms

- “A list of well-defined instructions for completing a task”



Initial state

selection

Instructions

Sequence

- A series of instructions
- ...to be carried out one after the other...
- ...without hesitation or question
- For example:
  - How to cook a Gourmet Meal^{TM}

Sequence - Example

- Open freezer door
- Take out Gourmet Meal™
- Close freezer door
- Open microwave door
- Put Gourmet Meal™ on carousel
- Shut microwave door
- Set microwave on high for 5 minutes
- Start microwave
- Wait 5 minutes
- Open microwave door
- Remove Gourmet Meal™
- Close microwave door

A *sequence* is a series of instructions
to be carried out one after the other...
without hesitation or question

Timeout Questions…

What does the algorithm below compute?

```
Begin
    Read n1, n2, n3
    sum = n1+n2+n3
    ave = sum/3
    print ave
End
```

a. the average of three numbers
b. converts a Fahrenheit temperature to Celsius
c. Sorts three numbers from smallest to largest
d. Finds the maximum number

Activity 1: Fox, chicken, and corn

Swako, a Red Indian boy, was going on a journey to meet his future in-laws for the first time. It is an Indian tradition that when the groom meets the bride's parents, he gives them gifts to show his generosity. The gifts Swako chose were a fox, a chicken, and a sack of corn. He had to go across a wide, deep river and his canoe could only carry him and one other thing. If the fox and the chicken are left together, the fox will eat the chicken. If the chicken and the corn are left together, the chicken will eat the corn.

Write an algorithm for Swako to transport his gifts safely to the far side of the river.

The commands available are of the form:

- put x in the canoe
- paddle across the river
- take x from the canoe

Activity 2: Water Jug Problem

A villager is given two jugs- one with a capacity of 5 litres, the other with a capacity of 3 litres. She is asked to fetch exactly four litres of water from the river. Write an algorithm that will allow the villager to do this.

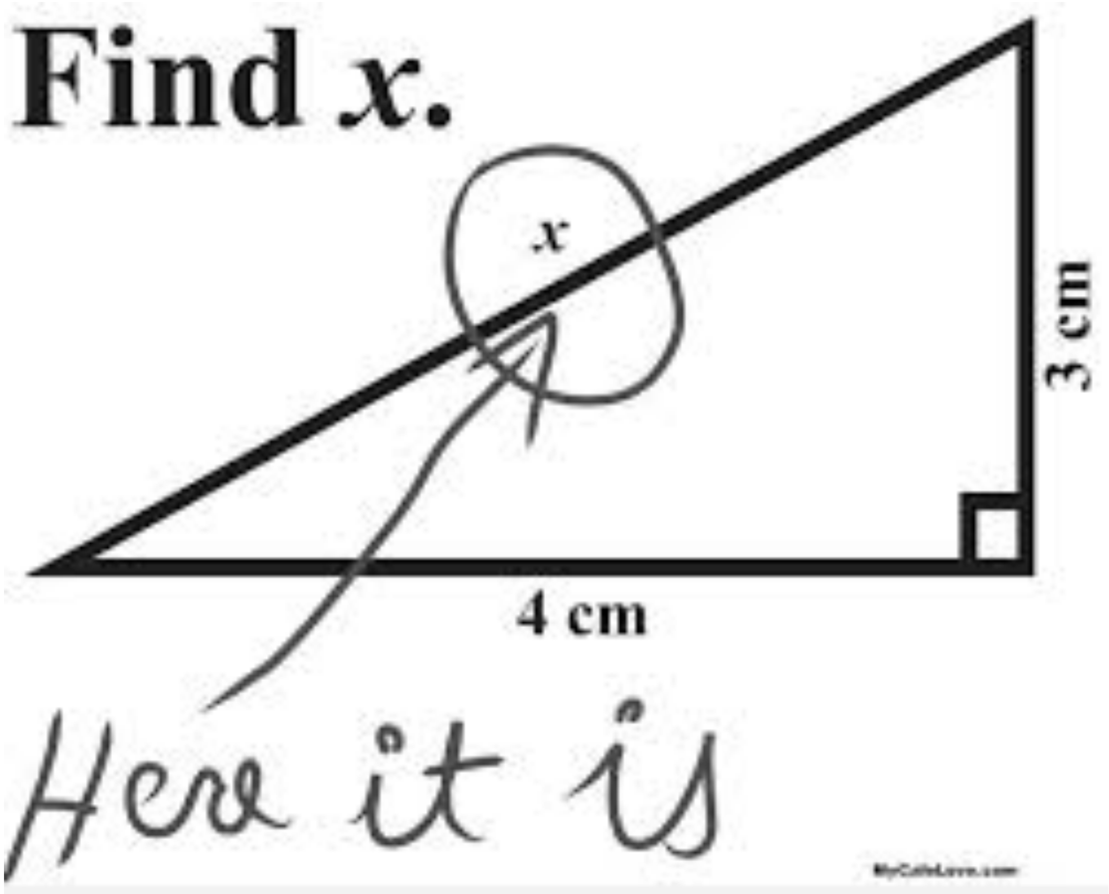
The **commands** available are

- Fill a jug from the river
- Fill a jug from another jug
- Empty a jug

Ask Yourself

- Do I understand the fundamentals of the Problem Solving Process
- Can I list the basic components of an algorithm?
- Can I write a basic algorithm to solve a problem?

Simple Algorithm for finding x



-
- 1) Since we know 2 sides of this triangle, we will use the pythagorean theorem to solve for side x.
- 2)Substitue the two known sides into the pythagorean theorem's formula:
- $A^2 + B^2 = C^2$

بوليتكنك البحرين

Bahrain Polytechnic

Unit 2 - Part 2 : Data, Variables and Constants

Data and Expressions

- This lecture focuses on:
  - the declaration and use of variables
  - primitive data
  - expressions and operator precedence
  - data conversions

Representing Values in Programs

- **Literals:** exact values typed into the source code of a program. Commonly used types are:
  - String literals: often used with println statements.
  - Numbers
- **Variables**
- **Constants**

Variables

- Are containers for values, allowing them to be changed without recompilation of the source code

Variable

Values

This jar can contain

10 cookies

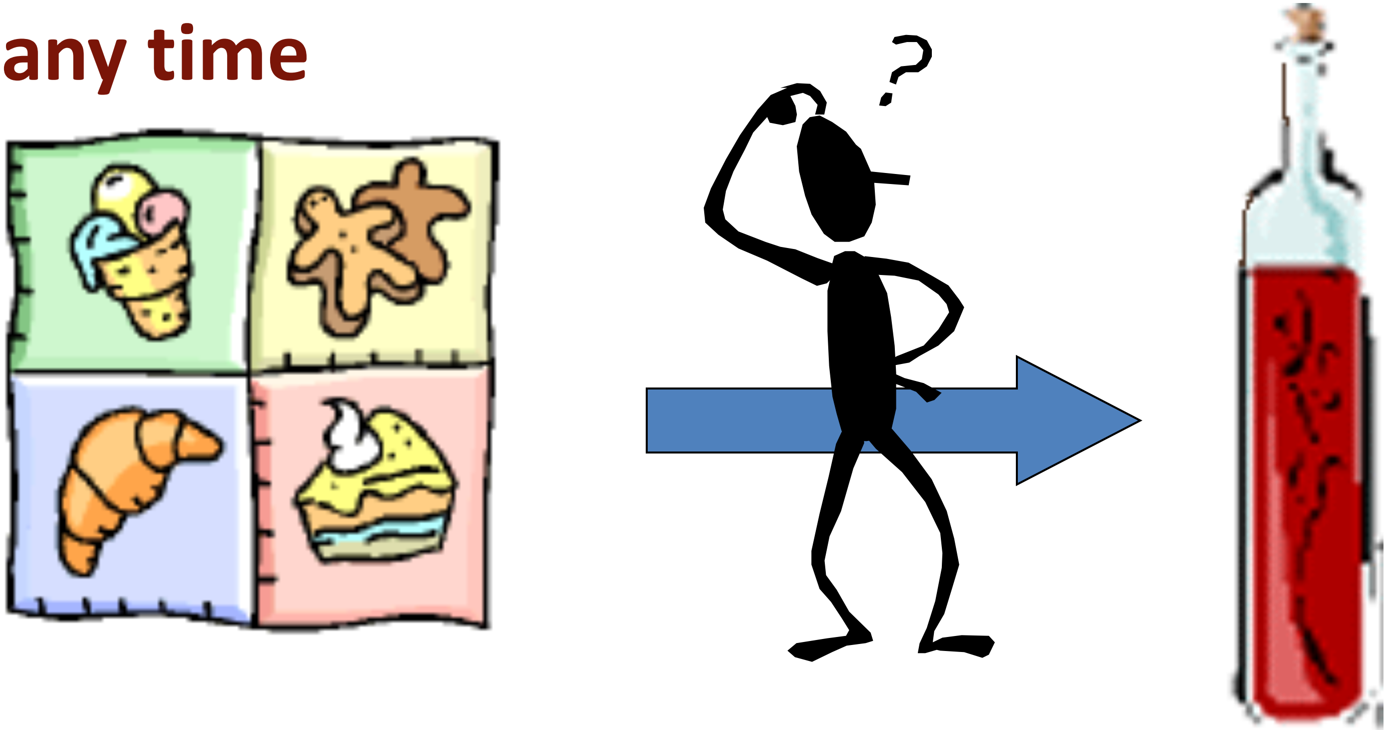
50 grams of sugar

3 slices of cake

etc.

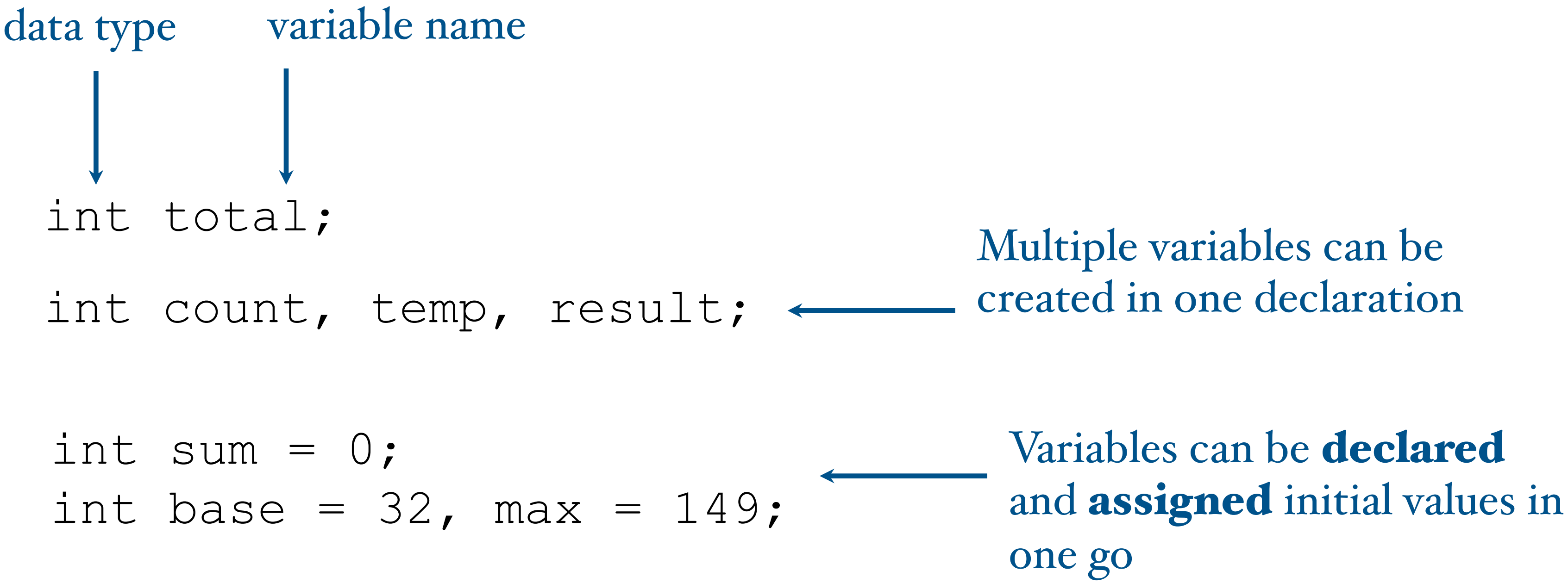
Variables and Data Type

- Variables are restricted to contain **a specific type** of value and can only hold **one value at any time**



Variables

- A *variable* is a name for a location in memory
- A variable must be ***declared*** by specifying the variable's **name (identifier)** and the **type** of information that it will hold



data type

variable name

```
int total;

int count, temp, result;


int sum = 0;
int base = 32, max = 149;
```

Multiple variables can be created in one declaration

Variables can be **declared** and **assigned** initial values in one go

PianoKeys.java (Listing 2.5, p42 Lewis et al.)

```
//********************************************************************
//  PianoKeys.java       Java Foundations
//
//  Demonstrates the declaration, initialization, and use of an
//  integer variable.
//********************************************************************

public class PianoKeys
{
   //-----------------------------------------------------------------
   //  Prints the number of keys on a piano.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int keys = 88;

      System.out.println("A piano has " + keys + " keys.");
   }
}
```

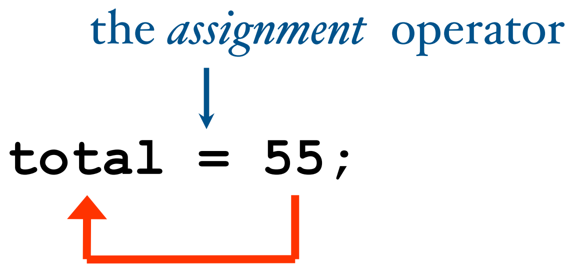
the variable name

a string (designated by “quotes”)

Timeout: Write the output generated from executing this code.

Assignment

- An ***assignment statement*** changes the value of a variable



the *assignment* operator

```
total = 55;
```

- The expression on the right is evaluated and the result is stored in the variable on the left
- The value that was previously in `total` is overwritten
- You can only assign a value to a variable that is consistent with the variable's declared type

Geometry.java (Listing 2.6, p43 Lewis)

```java
//********************************************************************
//  Geometry.java       Java Foundations
//
//  Demonstrates the use of an assignment statement to change the
//  value stored in a variable.
//********************************************************************

public class Geometry
{
   //-----------------------------------------------------------------
   //  Prints the number of sides of several geometric shapes.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int sides = 7;  // declaration with initialization
      System.out.println ("A heptagon has " + sides + " sides.");

      sides = 10;  // assignment statement
      System.out.println ("A decagon has " + sides + " sides.");

      sides = 12;
      System.out.println ("A dodecagon has " + sides + " sides.");
   }
}
```

Constants

- A constant is an identifier that is similar to a variable except that it holds the same value during its entire existence
- As the name implies, it is constant, not variable
- The compiler will issue an error if you write code that tries to change the value of a constant
- In Java, we use the **`final`** modifier to declare a constant

```
final int MIN_HEIGHT = 69;
```

Constants

Useful for three important reasons

- They give meaning to otherwise unclear literal values (“magic numbers”).

  E.g. “MAX_LOAD” is more meaningful to humans than the number “250”.

- They facilitate program maintenance.

  E.g. If a constant is used in multiple places, its value only needs to be set once.

- Constants formally establish that a value should not change. This helps to avoid inadvertent errors by (other) programmers.

Timeout Question

How many values can be stored in a variable at one time?

a. 0

b. 1

c. 2

d. 3

Primitive Data Types

- There are eight primitive data types in Java

`byte, short, int, long`	integers 19, -174
`float, double`	floating point numbers 17.95, -102.3333335
`char`	characters A, b, %, +
`boolean`	Boolean values true, false

Numeric Primitive Data

- The difference between the various numeric primitive types is their size, and therefore the values they can store:

Type	Storage	Min Value	Max Value
`byte`	8 bits	-128	127
`short`	16 bits	-32,768	32,767
`int`	32 bits	-2,147,483,648	2,147,483,647
`long`	64 bits	$< -9 \times 10^{18}$	$> 9 \times 10^{18}$
`float`	32 bits	+/- 3.4 x 10^{38} with 7 significant digits	
`double`	64 bits	+/- 1.7 x 10^{308} with 15 significant digits	

Assigning a value to a float variable

- To assign a value to a variable of type **float,** you need to add an **f** to the value:

```
float salary = 1221.45f;
```

Omission of the f will generate a syntax error.

Characters

- A char variable stores a single character
- Character **literals** are delimited by single quotes:
- Example declarations
  - char topGrade = ‘A’;
  - char terminator = ‘;’, separator = ‘ ’;
  - ‘a’ ‘X’ ‘7’ ‘$’ ‘,’ ‘\n’
- **Note:** in addition to the primitive char variable, which holds only one character, Java has a **String** object, which can hold multiple characters

Boolean

- A boolean value represents a true or false condition
- The **reserved words** *true* and *false* are the only valid values for a boolean type
- **Example: boolean done = false;**
- A boolean variable can also be used to represent any two states, such as a light bulb being on or off.

Timeout

- What are the four integer data types? How are they different?
- What are the two floating point data types? How are they different?
- What is a character set?

Identifiers

- *Identifiers* are the words a programmer uses in a program
  - can be made up of letters, digits, the underscore character (_), and the dollar sign
  - cannot begin with a digit
- Java is *case sensitive* - `Total`, `total`, and `TOTAL` are different identifiers
- By convention, programmers use different case styles for different types of identifiers, such as
  - *title case* for class names - `Lincoln`
  - *upper case* for constants - `MAXIMUM`

Identifiers

- Sometimes we choose identifiers ourselves when writing a program (such as `Lincoln`)
- Sometimes we are using another programmer's code, so we use the identifiers that he or she chose (such as `println`)
- Often we use special identifiers called *reserved words* that already have a predefined meaning in the language
- A reserved word cannot be used in any other way

Reserved Words

- The Java reserved words

```
abstract
assert
boolean
break
byte
case
catch
char
class
const
continue
default
do
double
else
enum
extends
false
final
finally
float
for
goto
if
implements
import
instanceof
int
interface
long
native
new
null
package
private
protected
public
return
short
static
strictfp
super
switch
synchronized
this
throw
throws
transient
true
try
void
volatile
while
```

White Space

- Spaces, blank lines, and tabs are called *white space*
- White space is used to separate words and symbols in a program
- Extra white space is ignored
- A valid Java program can be formatted many ways
- Programs should be formatted to enhance readability, using consistent indentation

بوليتكنك البحرين

Bahrain Polytechnic

Unit 2 - Part 3: Operators and Expressions

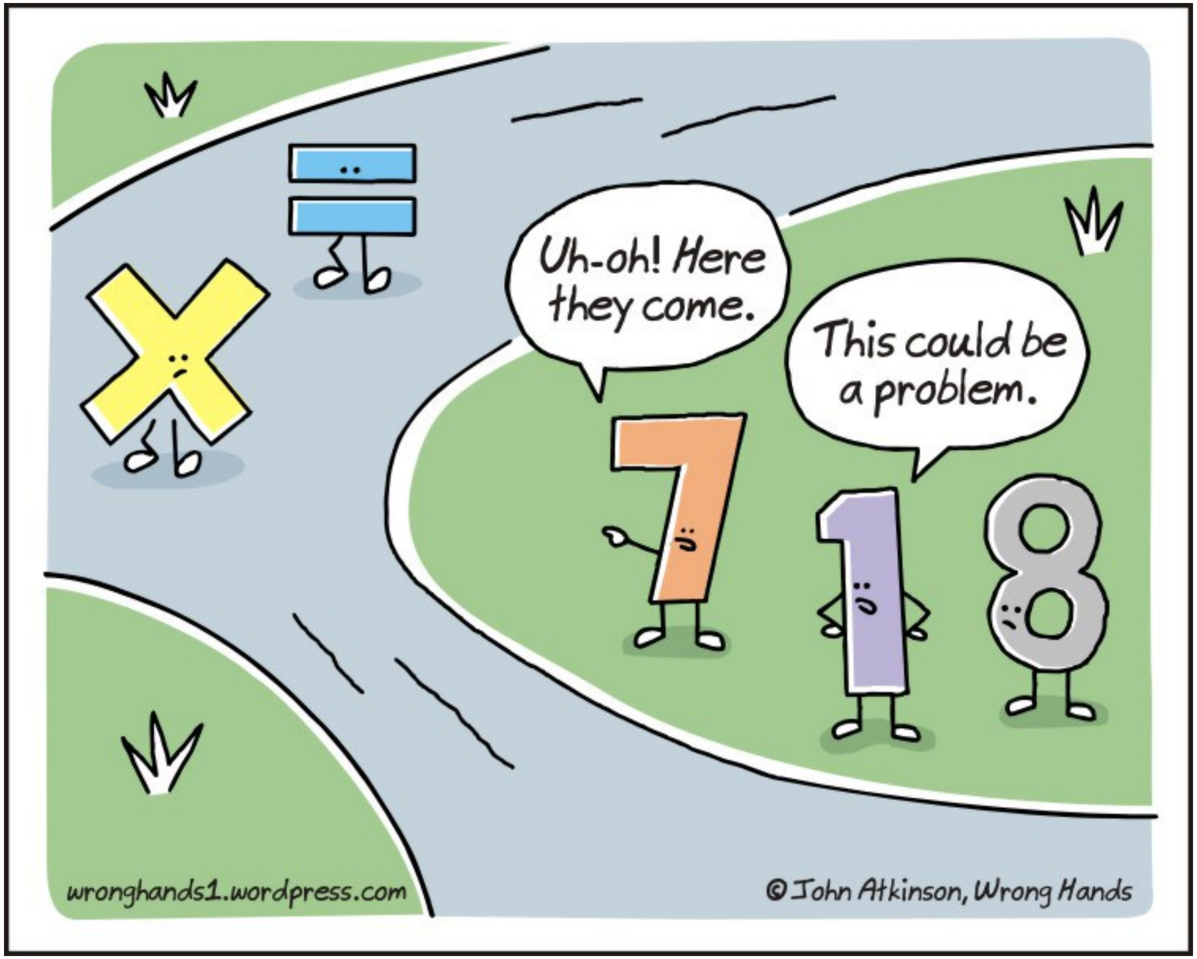
Expressions

- An *expression* is a combination of one or more **operators** and **operands**
- ***Arithmetic expressions*** compute numeric results and make use of the following **arithmetic operators:**

Addition	**+**
Subtraction	**-**
Multiplication	*****
Division	**/**
Remainder	**%**

- If either or both operands used by an arithmetic operator are floating point, then the result is a floating point

Operators and Operands



Division and Remainder

- If both operands to the division operator (`/`) are integers, the result is an integer (the fractional part is discarded)

`14 / 3`	**equals**	`4`
`8 / 12`	**equals**	`0`

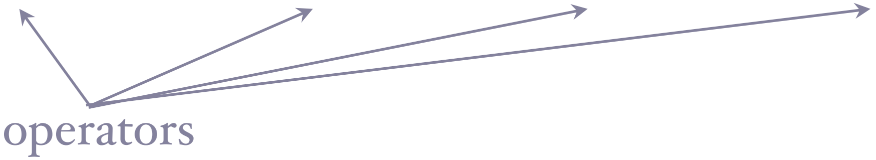
- The remainder operator (`%`) returns the remainder after dividing the second operand into the first

`14 % 3`	**equals**	`2`
`8 % 12`	**equals**	`8`

Operator Precedence

- Operators can be combined into complex expressions

```
result  =  total + count / max - offset;
```



operators

- Operator precedence (order of evaluation):
  - Operators in parentheses (innermost first)
  - multiplication, division, and remainder
  - addition, subtraction, and string concatenation
  - operators with the same precedence are evaluated from left to right
  - assignment operator

Operator Precedence

- What is the order of evaluation in the following expressions?

```
a + b + c + d + e
  1   2   3   4
```

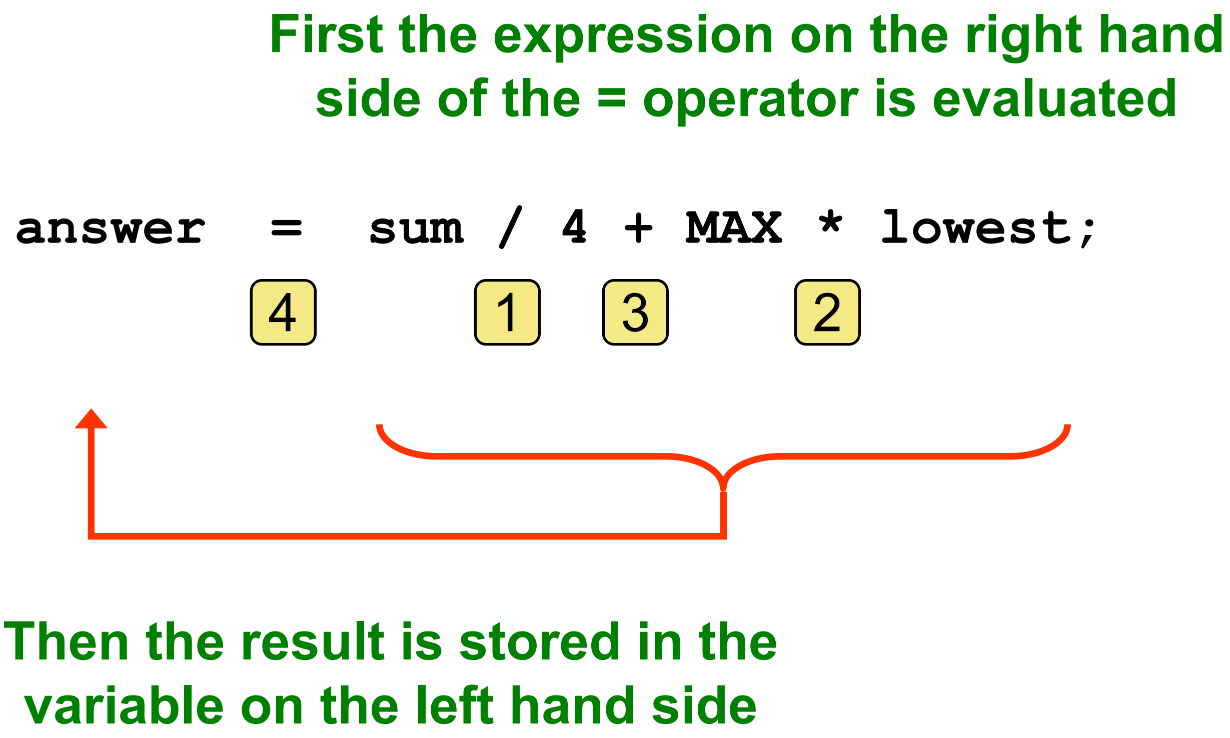
```
a + b * c - d / e
  3   1   4   2
```

```
a / (b + c) - d % e
  2    1    4   3
```

```
a / (b * (c + (d - e)))
  4    3    2    1
```

Assignment Revisited

- The assignment operator has a lower precedence than the arithmetic operators



First the expression on the right hand side of the = operator is evaluated

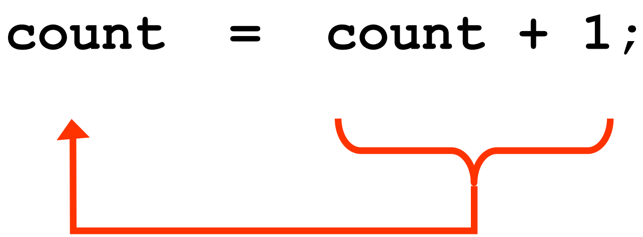
```
answer  =  sum / 4 + MAX * lowest;
```

Then the result is stored in the variable on the left hand side

Assignment Revisited

- The right and left hand sides of an assignment statement can contain the same variable

First, one is added to the original value of count



```
count  =  count + 1;
```

Then the result is stored back into count (overwriting the original value)

Increment and Decrement

- The increment and decrement operators use only one operand.
- The *increment operator* (++) adds one to its operand
- The *decrement operator* (−−) subtracts one from its operand
- The statement

  ```
  count++;
  ```

  is functionally equivalent to

  ```
  count = count + 1;
  ```

Timeout question

- What is the order in which the following expression is evaluated?

i) (a+b) * c + d * e

ii) a + (b-c) * d - e

Data Conversion

- Sometimes it is convenient to convert data from one type to another.
- In a particular situation we may want to treat an integer as a floating point value
- Such conversions do *not* change the type of a variable or the value that's stored in it – they only temporarily convert the variable’s *value* as part of a computation

Data Conversion

- Conversions must be handled carefully to avoid losing information
- ***Widening*** *conversions* are safest because they tend to go from a small data type to a larger one (such as a `short` to an `int`)
- ***Narrowing*** *conversions* can lose information because they tend to go from a large data type to a smaller one.
- In Java, data conversions can occur in three ways
  1. assignment conversion
  2. promotion
  3. casting

Assignment Conversion

- *Assignment conversion* occurs when a value of one type is assigned to a variable of another.

- E.g. the following assignment converts the value in **`dollars`** to a **`float`.**
    - `float money;`
    - `int dollars = 3;`
    - `money = dollars;`

  Note: the value or type of dollars did *not* change

- Only widening conversions can happen via assignment

Promotion

- *Promotion* happens automatically when operators in expressions convert their operands

- Eg:

```
result = sum / count;
```

if `sum` is a `float` and `count` is an `int`, the value of `count` is converted to a floating point value to perform the calculation

Questions

- Calculate the following:
  - A = 3 * 3
  - B = 3 / 2
  - C = 3.0 * 3.0
  - D = 3.0 / 2.0
  - E = 3.0 * 3
  - F = 3.0 / 2
  - G = 3 / 2.0

Casting

- Casts are helpful where we need to treat a value temporarily as another type. If `total` and `count` are integers and we want a floating point result we need to cast the values.
- Eg:
  - average = (float) total/count;
- If the cast had not been included, the operation would perform an integer division and truncate the answer before assigning it to result.

Timeout question

What is the type of value returned by the expression 7.0 + 5 / 3?

a. char

b. int

c. double

d. String

Ask Yourself

- Do I know the Java *primitive* data types?
- Can I *declare* a variable of a primitive data type and assign a value?
- Am I able to write *basic expressions*?
- Can I evaluate an expression using *operator precedence*?
- Do I understand the importance of *Data Conversion*?

YOUNGER SIBLINGS WHO A

COLLEGE OR HIGH SCH

NUMBER ONE PIECE OF ADV

SHOULD LEARN HOW TO

- MARK ZUCKERBERG
CEO, FACEBOOK