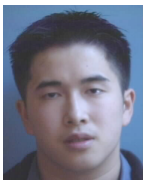
CS107
Introduction to Computer Science

Lecture 2

An Introduction to Algorithms:
Basic instructions and Conditionals

Administrativia

- **Lab access**
  - Searles 128:
    - Mon-Friday 8-5pm (unless class in progress) and 6-10pm
    - Sat, Sun noon-10pm
  - Searles 117: 6-10pm, Sat-Sun 12-10pm
- **Study group**
  - Leader: Richard Hoang'05
  - Time: TBD (Mondays ??)
  - Location: Searles 128



How do people think???????

- Puzzle:
  - Before A, B, C and D ran a race they made the following predictions:
    - A predicted that B would win
    - B predicted that D would be last
    - C predicted that A would be third
    - D predicted that A's prediction would be correct.
  - Only one of these predictions was true, and this was the prediction made by the winner.

  In what order did A, B, C, D finish the race?

Readings

- *A techie, absolutely, and no more*
- *Faster supercomputer aiding weather forecasts*
- *Space station gets HAL-like computer*

Algorithms

- How to come up with an algorithm?
  - Part of it its routine
  - Part it's problem solving
- So far we've talked about how to solve a problem (e.g. sorting) at a high level of abstraction
- If we are to write down in detail an algorithm, what is a good way to express it?
  - In English??
  - In a programming language??

Expressing algorithms

- Is natural language good?
  - For daily life, yes…but for CS is lacks structure and would be hard to follow
  - Too rich, ambiguous, depends on context
- How about a programming language?
  - Good, but not when we try to solve a problem..we want to think at an abstract level
  - It shifts the emphasis from how to solve the problem to tedious details of syntax and grammar.

Pseudocode

- Pseudocode = English but looks like programming
- Good compromise
  - Simple, readable, no rules, don't worry about punctuation.
  - Lets you think at an abstract level about the problem.
  - Contains only instructions that have a well-defined structure and resemble programming languages

Building algorithms

- **Basic (primitive) instructions**
  - Read the input from user
  - Print the output to the user
  - Cary out basic arithmetical computations
- **Conditional instructions**
  - Execute an instruction (or a block of instructions) if a condition is true
- **Repeat instructions**
  - Execute a block of instructions (multiple) times until a certain condition is met
- and…**variables**

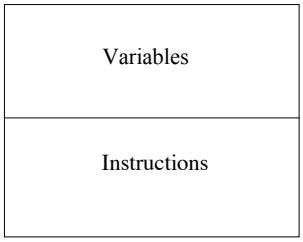
Variables

Variable
- A memory location that can store a value; we address it by its name
- Think of it as a box into which you can store a value, and from which you can retrieve a value

Examples: i ☐ M ☐

A model for visualizing an algorithm

Algorithm



- An algorithm consists of instructions
- Instructions involve variables

Basic instructions

1. Assign values to variables using basic arithmetic operations

- Set the value of x to 3
- Set the value of y to x/10
- Set the value of z to x +25

2. Get input from user
- Get the value of x from user

3. Print to user (screen)
- Print the value of y, z to the user

Basic instructions

1. Assign values to variables using basic arithmetic operations

Example: Let's assume we have two variable, a and b. The current value of a is 10 and the current value of b is 31.
- We execute:
  - Set the value of a to 3

  This instruction changes the value of a to 3.
- Now we execute
  - Set the value of b to a*3 + 12

  This instruction changes the value of b to 3+3 + 12 = 21

Basic instructions

2. Read input from user

Example:

get the value of [variable] a [from user]

- When this instruction is executed, the algorithm stops, waits for the user to type in an answer, and the value typed by the user is stored into variable a. Whatever the value of a was before, after this instruction it is whatever the user entered.

Basic instructions

3. Print to user (screen)

Print variables and/or text

Example:

print the value of variable x

- When a print instruction is executed, the algorithm writes to screen the value of variable x

print "Hello world!"

- This instruction prints to screen the text "Hello world!"

- Why do we need quotation signs?
  - Print hello
  - Print "hello"

Example

Write an algorithm that asks the user for three numbers, computes their sum and average and outputs them.

One possible algorithm for this problem:

Variables: a,b,c, sum, avg
- Get the values of a, b, c from user
- Set avg to (a+b+c)/3
- Set sum to (a+b+c)
- Print sum, avg

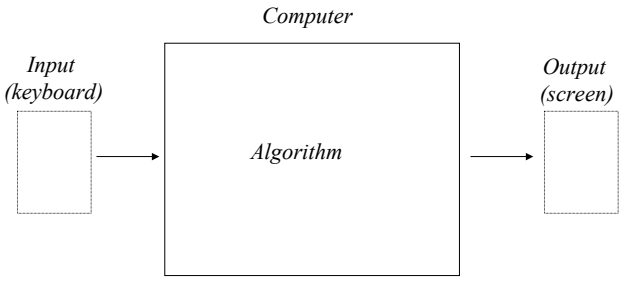
Example

Write an algorithm that reads the value of a circle radius from the user, and prints the circumference of a circle with that radius.

One possible algorithm for this problem:

variables: r, c
1. Get the value of r from user
2. Set c to 2 * pi * r
3. Print "The circumference of your circle is " c

A model for visualizing an algorithm's behavior



Exercise

Write an algorithm that asks the user for his year of birth, then prints it out, says thank you and good bye. It should look like this:

Hi. Please tell me your year of birth: 1920
You said 1920. Congratulations. Goodbye.

Conditional Instructions

Specifies an instruction (or a block of instructions) that may or may not be done, depending whether the condition is true or false. The else part is optional.

if <condition> then

<instruction(s) to be done>

[else <instruction(s) to be done otherwise>]

Example

if the value of x is 0 then set the value of a to 0, else set the vale of a to a+1

If the value of b is equal to the value of a than print "equal numbers", else print "bad luck.."

Conditional instructions

- Variants:
  - Both then and else cases
  - Only then case (no else)
  - One instruction or a block of instructions (use indentation)
- Examples:
  - If the value of a is equal to the value of b then print "The numbers are equal" else print "The numbers are not equal"
  - If the value of a is greater than10 then substract 10 from a
  - If the value of x is >10 then set b to x - 10
    set c to x+10
    print b,

    Else set x to x + 10
    print x

Example

Consider the following algorithm:

print "Enter two values"
get a, get b
if the value of a is larger then the value of b then print a
else print b

What does the program print if the user enter 10, 3?
Describe in English what the algorithm accomplishes.

Exercise

Write an algorithm that asks the user for two numbers that represent the scores to a soccer game Bowdoin vs. Colby, and prints out either "You won", "You tied" or "You lost". For example:

Enter Bowdoin score: 4
Enter Colby score: 2
You won.

Pseudocode

- This is correct
  - Get the value of a from user
  - Get the value of b from user
  - If the value of a is equal to the value of b then print the value of a, else set the value of c to the value of a + the value of b and print c
- But, we'll slowly converge towards a more compact, math-like style
  - Get a, get b
  - If (a==b) then print a
    Else set c = a + b
    print c
- Note:
  - use == to test for equality
  - use = for assignment

Pseudocode examples

Equivalent:

- Set the value of a to 1
- Set a to 1
- **a=1**

Equivalent

- Add 1 to count
- Set count to count + 1
- Increment the value of count by 1
- **count = count + 1**

Writing in pseudocode gives you the freedom to choose any of these. In general people prefer the shorter form.

- **Incorrect**
  - Set 1 to a
  - Add a + b (what do you do with the result?)
  - Set a+b +3
- **NOT the same**
  - set a to b
  - set b to a
- Example: what is the output of the following algorithms?

set a to 2	set a to 2
set b to 4	set b to 4
set a to b	set b to a
print a, b	print a, b

Example

Write an algorithm to compute the distance traveled and the average miles per gallon on a trip when given as input the number of gallons used and the starting and ending mileage readings on the odometer.

Variables: gallons, start, end, distance, mpg
get gallons, start, end
distance = end - start
mpg = distance / gallons
print mpg
if mpg > 25.0 then print "You are getting good gas mileage"
else print "You are NOT getting good gas mileage"

Conditions

- If (condition) then .... else ...
- Condition must evaluate to true or false
- Building conditions
  - Use >, >=, <, <=, ==, remainder
- Examples:
  - If (a >= b) ...
  - If (a + b <= c)...
  - If (x == y)...
  - If (the remainder of x divided by y == 0) ...
    - Note: the operator for "remainder" is denoted %
  - If (x%2 == 0) print "even number"

Combining Conditions

- Use logical operators: and, or, not
  - If (condition 1 and condition 2)
    - True if BOTH are true
  - If (condition 1 or condition 2)
    - True if at least one is true
  - If (not condition 1)
    - True if condition 1 is not true
- Examples
  - If (score > 85 and score < 95) then print "grade is A-"
  - If (not (score < 2)) print "you passed"

Exercise

Write an algorithm that asks the user for an exam score, which is expected to be a number between 0 and 100. If the score does not fit in this range, display an error message. Otherwise assign a letter grade A, B, C, D or F. The breaks are 90, 80, 70, 60. For example:

Please enter a score, between 0 and 100: 110.
Sorry, that's impossible.

Please enter a score between 0 and 100: 85
The grade is B.