Algorithms & Pseudo Code

Due this week

- **Recitation**
  - Install VS Code
  - Tutorials and videos on Canvas, based on the operating system of your computer
- **Syllabus Quiz** due tonight!
- **Homework 0**
  - Submit zip file on Canvas. Check the due date!
- Participation Quiz - to be discussed on Friday
- Start going through the textbook readings and watch the videos
  - Take **Quiz 1**. Check the due date!

Algorithms and Pseudo Code

Topics

1. What is programming?
2. Anatomy of a computer
3. Machine code and programming

Videos

4. Becoming familiar with your programming environment
5. Analyzing your first program
6. Errors

Next time

7. Problem solving: algorithm design ➡ Today

Warm Up Activity

- Program me to eat the number of pistachios I'm hungry for

- https://bit.ly/3cdEBml
    - (Only open this when ready to copy/paste)
    - The last character is a lowercase "L"

Warm Up Activity

- What were the challenges of doing this activity?

- Where did you fail to communicate your intentions to the "computer" (me)? Why?

What am I trying to program?

- I was a human, and came with no documentation

- Fortunately, computers are very simple, and have documentation
  - Watch the videos!!

Algorithms

- Every program is based on an algorithm (or more)
- An algorithm is like a recipe for cooking
  - It tells the ingredients (*inputs*)
  - It tells the sequential steps for processing the inputs
  - It tells the serving size and style (*outputs*)
- The computer acts like a chef, exactly following the algorithm recipe
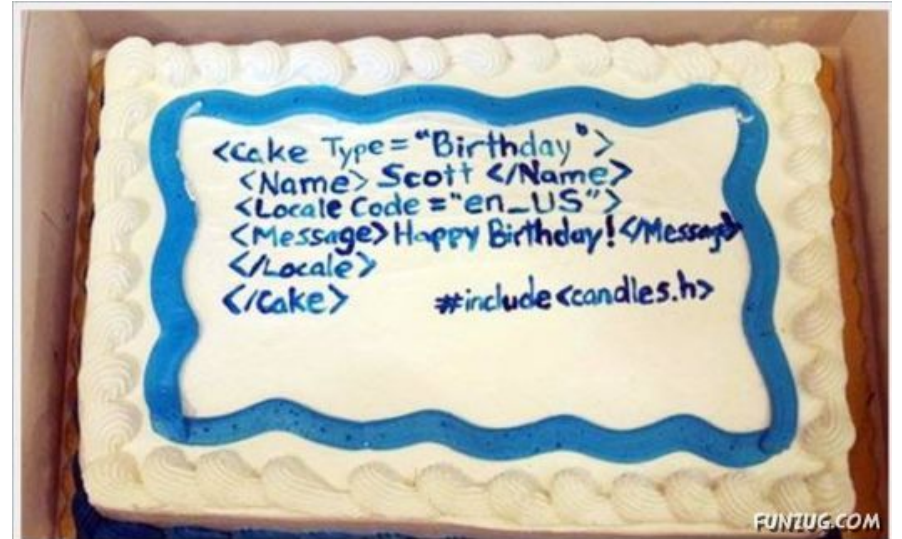
Algorithms

The computer acts like a chef, exactly following the algorithm recipe

- Chef Computer does not know the meaning of "a whole bunch of flour"

- Chef Computer also does not know that of course you don't include the eggshells when the recipe calls for "two eggs"

Making a Cake

- What do we need to make the cake?

- Let's assume you don't have anything needed for making the cake.

High Level Abstraction for Making a Cake

Make a Cake:
- Drive to Store
- Buy Ingredients
- Drive Home
- Bake the Cake
- ... and frost it like a nerd

This is the **algorithm** for making a cake.

It uses high level abstractions to make the algorithm easy to understand.

High Level Abstraction for Making a Cake

Make a Cake:
- Drive to Store
- Buy Ingredients
- Drive Home
- Bake the Cake
- … and frost it like a nerd

→

- Get into and Start the Car
- Drive to King Soopers
- Park the Car
- Turn the Car Off
- Get Out of Car

High Level Abstraction for Making a Cake

Make a Cake:

- Drive to Store
- Buy Ingredients
- Drive Home
- Bake the Cake
- … and frost it like a nerd

- Get into and Start the Car
- Drive to Home
- Park the Car
- Turn the Car Off
- Get Out of Car

High Level Abstraction for Making a Cake

Make a Cake:

- Drive to Store
- Buy Ingredients
- Drive Home
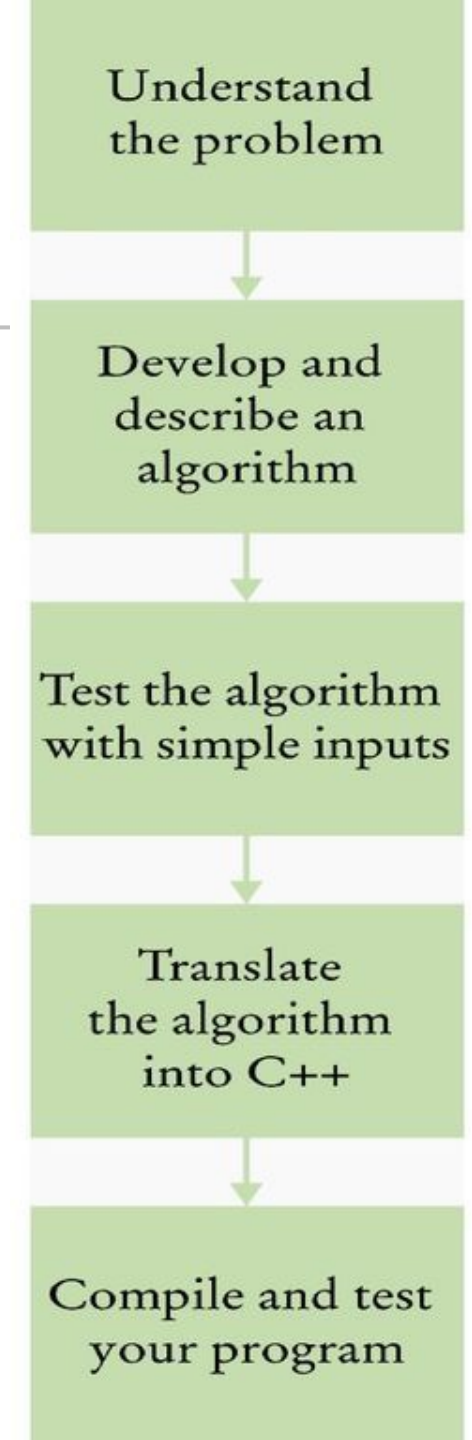- Bake the Cake
- … and frost it like a nerd

→

- Get into and Start the Car
- Drive to DESTINATION
- Park the Car
- Turn the Car Off
- Get Out of Car

- we needed to drive to store and drive to home
- wouldn't it be nice to be able to solve the general problem of driving to … **destination?**

Input: initial location, destination

Output: sequence of steps

The Software Development Process

- For each problem the programmer goes through these steps

- ***You MUST write an algorithm in words, pictures, and/or equations before attempting to translate to C++***

Understand the problem

Develop and describe an algorithm

Test the algorithm with simple inputs

Translate the algorithm into C++

Compile and test your program

6 Building Blocks for Computational Representations

1. Create a variable to store a value for later use
2. Modify the value of a variable
3. Get input or generate output
4. Check if a statement is True or False
5. Repeat a statement or collection of statements
6. *Encapsulating a collection of statements*

6 Building Blocks for Computational Representations

1. Create a variable to store a value for later use

    What is a variable?
    - Have you encountered variables before? Where?

Variables	Values or quantities that change over time
Range of a variable	What are all the possible values it could take?
Variable type	Numeric, text, other

Example story: Alexis is 18 y.o. and her grandma is approaching 80.

6 Building Blocks for Computational Representations

1. Create a variable to store a value for later use

Examples:

lemons = 5

celsius = 15

oranges = 4

fruit = lemons + oranges

6 Building Blocks for Computational Representations

2. Modify the value of a variable

Examples:

lemons = 5

oranges = 4

fruit = lemons + oranges

fruit = fruit + bananas

celsius = 15

fahrenheit = celsius * 9 / 5 + 32

6 Building Blocks for Computational Representations

3. Get input or generate output

Examples:

lemons = 5

oranges = 4

fruit = lemons + oranges

fruit = fruit + bananas

Print out the number of fruits

get the celsius value from user (and save the value entered by the user in variable *celsius*)

fahrenheit = *celsius* * 9 / 5 + 32

Print the fahrenheit value

6 Building Blocks for Computational Representations

4. Check if a statement is True or False

Examples:

lemons = 5
oranges = 4
fruit = lemons + oranges
fruit = fruit + bananas
Print out the number of fruits

if the number of fruits is larger
than 10
        print "lets make a fruit salad"

get the celsius value from user
(and save the value entered by the user
in variable *celsius)*

fahrenheit = *celsius* * 9 / 5 + 32
Print the fahrenheit value

If fahrenheit is less than or equal 32
        display "its freezing in here"

6 Building Blocks for Computational Representations

5. Repeat a statement or collection of statements

Examples:

lemons = 5
oranges = 4
fruit = lemons + oranges
fruit = fruit + bananas
Print out the number of fruits

if the number of fruits is larger than 10
        print "lets make a fruit salad"

for each fruit
        cut fruit into pieces

or

While any piece of fruit is bigger than bite sized
        select largest piece of fruit
        cut selected piece of fruit into two pieces

Algorithms

- Step-by-step procedure for solving a problem or accomplishing some task

- When your algorithm has enough detail (it clearly informs how you will write your code), you are usually writing in *pseudo code*

Pseudo Code

pseu·do·code
ˈso͞odōˌkōd/

A notation resembling a simplified programming language for describing algorithms

- Intended for human readability, not a computer's

- Does not need to be syntactically correct code

- Provides a language independent way to describe the steps of an algorithm

Algorithms

- Step-by-step procedure for solving a problem or accomplishing some task

- When your algorithm has enough detail (it clearly informs how you will write your code), you are usually writing in *pseudo code*

Describing an algorithm with Pseudocode (example 1)

Problem Statement:

You are asked to simulate a postage stamp vending machine. A customer inserts dollar bills into the vending machine, selects the number of stamps needed, and then pushes a "purchase" button. The vending machine gives out as many first-class stamps as the customer requested and can pay for, and returns the change in coins. A first-class stamp costs 55 cents. The machine is broken. The only available coins for change are dollar coins and pennies.

Describing an algorithm with Pseudocode

Step 1 Determine the **inputs** and **outputs**.

Inputs:
- *The amount of money* the customer inserts
- *The number of stamps* wished to purchase

Outputs:
- The number of stamps the machine returns
- The change:
    - The number of dollar coins
    - The number of pennies

Describing an algorithm with Pseudocode

Step 2 Break down the problem into smaller tasks

- Ask the user for input: how much money is inserted and how many stamps they wish to purchase
- Determine the total price
- Compute change value
  - Compute how many dollar coins and how many pennies

Describing an algorithm with Pseudocode

Step 3   Describe each subtask in pseudocode.

You will need to arrange the steps so that any intermediate values are computed before they are needed in other computations.

Describing an algorithm with Pseudocode

Step 3 Describe each subtask in pseudocode.

Ask user to input a whole number for the dollar amount inserted into the machine
> Save in the variable *initial_money*

Ask user to input a whole number for the number of stamps wished to purchase
> Save in the variable *num_stamps*

Compute total purchase price
> *purchase_price_cents = num_stamps * 55*

Describing an algorithm with Pseudocode

Step 3   Describe each subtask in pseudocode.

Compute change needed
        change = initial_money - purchase_price

Give change: ... how do we give change?
Example:
    initial_money = $5
    num_stamps = 5
    *purchase_price = 5 * $0.55 = $2.75*
    change = $2.25

... which is $2 and 25 pennies
How can a C++ program come to the same conclusion?

Describing an algorithm with Pseudocode

Step 3   Describe each subtask in pseudocode.

Compute change needed
change_cents = initial_money *100 - purchase_price_cents

Example:
    initial_money = $5
    num_stamps = 5
    purchase_price_cents = 5 *55 = 275
    change_cents = 5 * 100 – 275 = 225

Describing an algorithm with Pseudocode

Step 3 Describe each subtask in pseudocode.

Give change:
change_dollars = change_cents / 100 (w/o remainder)

change_pennies = change_cents – 100*change_dollars

or

change_pennies = change_cents %100 (remainder),
        where % is the **modulo** operator

Describing an algorithm with Pseudocode

Step 4   Test your pseudocode by working a problem.

Use these sample values:
Example 1:
    initial_money = $5
    num_stamps = 5
    *purchase_price_cents = 5 * 55 = 275*
    *change_cents = 5 * 100 – 275 = 225*
    change_dollars = change_cents/100 = 2
    change_pennies = change_cents%100 = 25

Time check

Describing an algorithm with Pseudocode

Step 4   Test your pseudocode by working a problem.

Use these sample values:
Example 2:
    initial_money = $5
    num_stamps = 7
    *purchase_price_cents = 7 * 55 = 385*
    *change_cents = 5 * 100 – 385 = 115*
    change_dollars = change_cents/100 = 1
    change_pennies = change_cents%100 = 15

Describing an algorithm with Pseudocode

Step 4   Test your pseudocode by working a problem.

Are we ready to implement it into code? Have we thought of all possibilities?

Example 3:
   initial_money = $5
   num_stamps = 17
   *purchase_price_cents = 17 * 55 = 935*
   *change_cents = 5 * 100 – 935 = - 435*
   change_dollars = change_cents/100 = ?
   change_pennies = change_cents%100 = ?

Describing an algorithm with Pseudocode

Step 4   Test your pseudocode by working a problem.

Are we ready to implement it into code? Have we thought of all possibilities?

Example 4:
   initial_money = $5
   num_stamps = -3
   purchase_price_cents = …
   change_cents = …
   change_dollars = change_cents/100 = …
   change_pennies = change_cents%100 = …

Describing an algorithm with Pseudocode

Step 4   Test your pseudocode by working a problem.

Are we ready to implement it into code? Have we thought of all possibilities?

Example 5:
    initial_money = $5
    num_stamps = r
    purchase_price_cents = …
    change_cents = …
    change_dollars = change_cents/100 = …
    change_pennies = change_cents%100 = …

Describing an algorithm with Pseudocode

Step 3   Describe each subtask in pseudocode. **Make changes!**

Ask user to input a whole positive number for the dollar amount inserted into the machine
>   Save in the variable *initial_money*

Ask user to input a whole positive number for the number of stamps wished to purchase
>   Save in the variable *num_stamps*

Compute total purchase price
>   *purchase_price_cents = num_stamps * 55*

Describing an algorithm with Pseudocode

Step 3 Describe each subtask in pseudocode. **Make changes!**

If *purchase_price_cents <= initial_money*100*
      then, Compute change needed
      change_cents = initial_money *100 - purchase_price_cents
Otherwise
      print "Not enough money"

Give change:
change_dollars = change_in_cents / 100 (w/o remainder)
change_pennies = change_in_cents %100 (remainder)

Is this correct?
                  NO!

Describing an algorithm with Pseudocode

Step 3   Describe each subtask in pseudocode. **Make changes!**

If *purchase_price_cents <= initial_money*100*
        then, Compute change needed
        *change_cents = initial_money *100 - purchase_price_cents*

        Give change:
        change_dollars = change_in_cents / 100 (w/o remainder)
        change_pennies = change_in_cents %100 (remainder)

Otherwise
        print "Not enough money"

Questions?