What Is Asked of Computer Programmers?

Write a computer program to perform the transformation

- *Input* is information to process.
- *Output* is the result of the transformation.
Programming Process

Problem Solving Phase
- Analysis and Specification
  • General solution
  • Verification

Implementation Phase
- Concrete Solution
- Test
- Identify new features, bugs

Computer Program Development

Ponder → Natural Language

Does anything happen in between?

Compile/Link
From Natural Language to C++

- Organize thoughts
- Clearly identify the input
- Clearly identify the output
- Identify the steps to transform the input to the output.
  - May require refinement

Fundamental Goal in Programming

- **Goal:** Develop an *algorithm* for solving the problem.
- **Definition of algorithm**
  - A finite sequence of precise steps needed to solve a problem or perform a task.
- **Written in natural language**
  - Use unambiguous words to describe each step.
Properties of Algorithms

- **Finiteness**
  - It has to finish at some point
- **Absence of ambiguity**
  - Only one interpretation
- **Sequence**
  - Need to know the order of execution
- **Input/Output**
  - 0 or more inputs and at least 1 output
- **Effectiveness**
  - Must be capable of performing each step

Why are these properties important?

Example 1

- Consider the steps for accelerating in a car:
  1. Move right foot to gas pedal.
  2. Apply pressure to gas pedal with right foot.
  3. If speed is too high, apply less pressure.
  4. If speed is too low, apply more pressure.
- Is this an algorithm? Why or why not?
Example 2

- **Problem**: Count the number of times the name “Mary” appears in a list of names.
  1. Get the list of names.
  2. Set a counter to zero.
  3. For each name in the list perform the following:
     a. Compare the name to “Mary” and if the name is “Mary”, add one to the counter.
  4. Display the value of the counter, which is the number of times “Mary” appears.

Problem Solving Phase

- Analysis and Specification
  - Read problem statement
  - Identify input and output
  - Identify transformation
- General Solution
  - Develop algorithm
- Verify
  - Try your algorithm yourself
Case Study: Miles to Kilometers

- **Problem**: You are responsible for creating a vacation package for tourists from France. The tourists want to know how far they will travel in the United States on their vacation. Because the tourists are French, they would prefer to use metric distances.

Case Study: Miles to Kilometers

- **Analysis**
  - Distances are given in miles in the United States
  - Distances are given in kilometers in France

- **Goal**: Transform a distance in miles to a distance in kilometers.
Case Study: Miles To Kilometers

- **Input**
  - A distance in miles
  - Assign the name `miles` to represent this distance

- **Output**
  - The distance in kilometers
  - Assign the name `kms` to represent this distance

- **Transformation**
  - 1 mile is 1.609 kilometers
    - Sometimes, you have to look up facts to solve a problem.

---

Case Study: Miles To Kilometers

- **Design 1**
  1. Get the distance traveled in miles
  2. Convert the distance to kilometers
  3. Display the distance in kilometers
Case Study: Miles To Kilometers

- **Design 2**
  1. Get the distance in miles
  2. Convert the distance to kilometers
     2.1 The distance in kilometers is 1.609 times the distance in miles.
  3. Display the distance in kilometers

Case Study: Miles to Kilometers

- **Design 3**
  1. Get the distance in miles, using miles to represent that value.
  2. Convert the distance to kilometers
     2.1 Using kms to represent the distance in kilometers,
     \[ \text{kms} = 1.609 \times \text{miles} \]
  3. Display kms, the distance in kilometers
Case Study: Miles to Kilometers

- Need to verify that the algorithm works with a few tests.

<table>
<thead>
<tr>
<th>miles</th>
<th>kms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.609</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>96.54</td>
</tr>
</tbody>
</table>

What happens if miles is negative?

Problem Solving Techniques

- Ask questions
- Look for things that are familiar
- Solve by analogy
- Means-ends analysis
- Divide and conquer
- Building-block approach
- Merging solutions
Exercise

- Problem: You are creating a vending machine that dispenses small toys. The machine returns change to the customer in $1 coins, quarters, dimes, nickels, and pennies. The machine always gives change using the minimum number of coins. Given an amount in change, determine the number and kind of coins to give back to the customer.

Giving Change: Algorithm 1

1. Get the amount of change in cents and use cents to represent that amount.
2. Convert cents to the number of dollars, quarters, dimes, nickels, and pennies to return. Use dollars, quarters, dimes, nickels, and pennies, to represent those values respectively.
Giving Change: Algorithm 1

2. Convert ...

2.1 Set each of dollars, quarters, dimes, nickels, and pennies to 0.

2.2 If the value of change is >= 100, then perform the following steps.

2.2.1 Add 1 to the value of dollars.

2.2.2 Subtract 100 from the value of change.

2.2.3 Repeat step 2.2

2.3 If the value of change is >= 25 then perform the following steps.

2.3.1 Add 1 to quarters

2.3.2 Subtract 25 from change

2.3.3 Repeat step 2.3
Giving Change: Algorithm 1

2. Convert …
   2.1 Set …
   2.2 If the value of `change` is >= 100 …
   2.3 If the value of `change` is >= 25 …
   2.4 If the value of `change` is >= 10 …
   2.5 If the value of `change` is >= 5 …
   2.6 If the value of `change` is >= 1 …
3. Display dollars, quarters, dimes, nickels, and pennies to the customer.

Giving Change: Algorithm 2

- Steps 1 and 3 are the same.
2. Convert …
   2.1 Set dollars to the quotient of `change` / 100 and update `change` to be the remainder of `change` / 100.
   2.2 Set quarters to the quotient of `change` / 25 and update `change` to be the remainder of `change` / 25.
   2.3 Set dimes to the quotient of `change` / 10 and update `change` to be the remainder of `change` / 10.
   2.4 Set nickels to the quotient of `change` / 5 and update `change` to be the remainder of `change` / 5.
   2.5 Set pennies to `change`. 
Programming Assignment Requirement

- Starting with program 2
- Each submission must contain your algorithm
  - In comments at the top of your source file
  - Use outline format for now
- Failing to include an algorithm is a 20 point penalty.