

## Algorithms

In the previous lesson the calculations were fairly simple. One or two values were input, a calculation was made and the result is output. Most programs have this sequence of input, calculate, output. However, the number of values input, the calculations and the output can be much more complicated.

In this lesson we will look at a slightly more complicated problem. We will start with a problem statement. Next we will determine the desired output, then determine what input and calculations are needed to achieve the desired output.

Initially, we will develop an algorithm (method for solving a problem) for a human to follow. If we can't solve the problem with pencil and paper, we probably can't write a program to solve it either.

**Problem** Your friend just bought a car. You look at the car in amazement. "This made it around the block?" You politely compliment the car then ask, "What kind of gas mileage does it get?" Your friend replies "I have no idea, how can I figure that out?"



**Before you continue with this lesson, try to write down step by step instructions for your friend to follow to calculate the miles per gallon for the car. This is an algorithm for calculating MPG.**

### Hints:

- You do not need to know how many gallons the tank holds.
- You may assume that if you fill the tank until it stops automatically, the full tank will always be the same amount.
- Filling the tank, then driving until it's empty is not "user friendly."

### Miles Per Gallon

The miles per gallon is the number of miles driven divided by the number of gallons used. For example if you drive 100 miles and use 5 gallons, the MPG is 20.

- In order to find the number of miles driven you will need the starting odometer reading and the final odometer reading.
- To find the number of gallons used you need to start with a full tank, then see how many gallons it takes to fill it back up.

### Algorithm for Finding Miles Per Gallon:

1. Go to a gas station and fill the car (until the pump clicks off.)
2. Write down the odometer reading. We will call this the **starting odometer**.
3. Drive until you need gas again.
4. Go to a gas station and fill the car (until the pump clicks off.)
5. Write down the odometer reading. We will call this the **final odometer**.
6. Write down how many gallons you put in. We will call this **gallons**.

We have all of the information we need and can now do the calculations. It is helpful if we make up some values and then do calculations with the sample values.

**starting odometer:** 8,900

**final odometer:** 9,000

**gallons:** 20

Before you continue. Find the MPG on paper for the example above. Write down every arithmetic operation you perform. (You can't use the number 100 unless you show where 100 came from.)

Distance = final odometer – starting odometer (9000-8900 → 100)

MPG = Distance / Gallons (100/20 → 5)

The car in the example gets 5 miles per gallon.

### Pseudocode

Programmers sometimes write pseudocode before they write the actual program. Pseudo means false: it is not really code because it does not follow syntax rules, it just lets us get the basic sequence of the program on paper. At this point however, we switch to using good variable names. That means the spaces in names like “starting odometer” have to be omitted. We will use the variables **start**, **final**, **gallons**, and **mpg**.

### Pseudocode for the MPG problem:

```
Input start
Input final
Input gallons
distance = final - start
mpg = distance/gallons
Output mpg
```

Note the order of the statements. For the most part the statements **must** be in that order.

- It is considered “user friendly” to ask for information in the order that it is gathered, or in the order that it is usually asked. (*Imagine the errors that would occur if a form asked for the zip code first, then the state.*)
- Most programs have a pattern of input, calculate, output. The program is easier to read if we input all of the information and then do all of the calculations, but sometimes a program must do some calculations before asking the user for more information.
- We had to calculate distance before we can calculate mpg because mpg uses distance.
- Obviously, you can't output mpg until it has been calculated.

After writing pseudocode it should be fairly easy to write the program from it. When you write the program you will decide:

- the type to declare each variable;
- an appropriate (*user friendly*) prompt for each input so that the user will know what to enter;
- how to display the result.