How to THINK like a Programmer

Problem Solving for the Bewildered

paul vickers

Problem Solving 2

a strategy for solving problems

John Brosnan,
Computing Dept, ITT

Aim of Lesson

- This lesson is all about problems and how we solve them
  - Problem-solving is the heart of programming
  - The problem with problems
  - A strategy for solving problems
    - Understanding the problem
    - Devising and carrying out the plan
    - Assessing the results
    - Documenting the solution
  - Applying the strategy

The heart of programming

- When you get a new gadget do you:
  A. Fully read the instructions before switching it on and using it?
  B. Start playing with it, learning how it works by trial and error and using the instructions only when you get stuck?
- Too many beginning programmers get a programming assignment and immediately start writing code **before they have thoroughly understood the problem**
- This leads to **frustration** and **bewilderment**

"Hours spent playing with a new software package can save minutes reading the manual"

The problem with problems

- **Before we write any code we must understand and solve the underlying problem**
- Problems can be **difficult**
- They may require intuitive leaps (e.g. cryptic crossword puzzles)
- Even the most systematic problem-solving technique still requires you to **think** for yourself
- In programming, the chosen programming language may introduce its own set of sub-problems to solve. Consider the following activity, a nice problem of arithmetic...

ACTIVITY

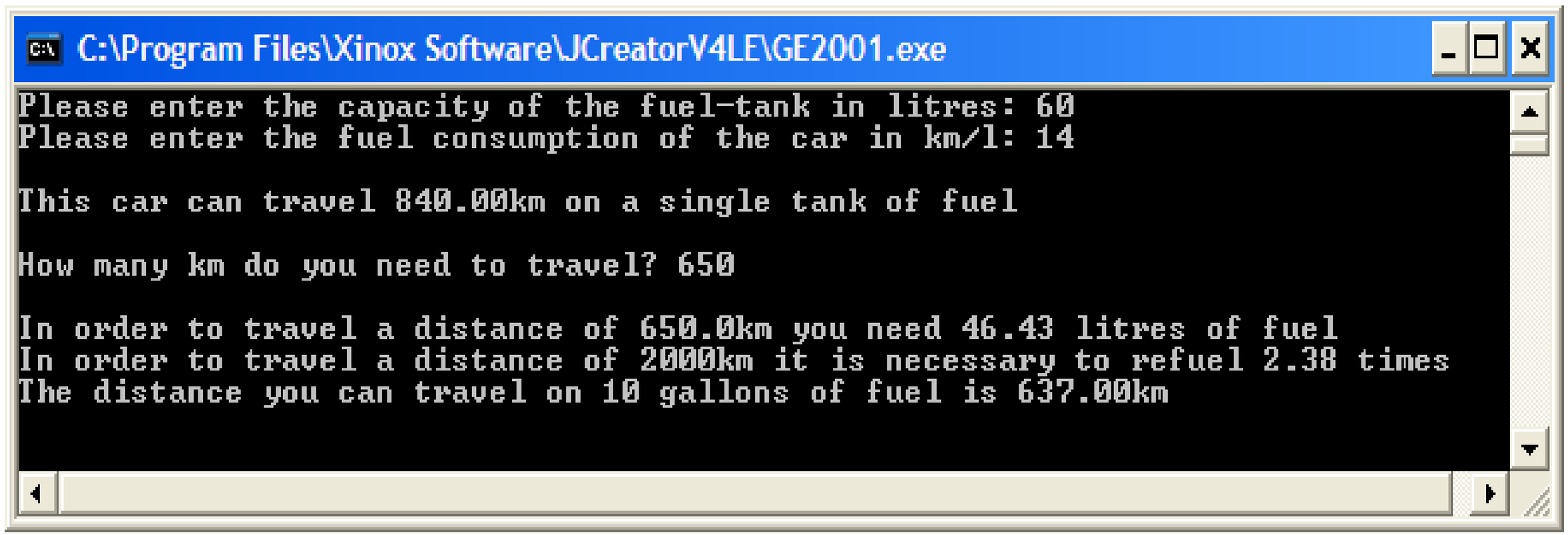
1. A certain car has a fuel tank with a capacity of 60 litres and an average fuel consumption of 14 km/l. How far can the car travel on one tank of fuel, and how many litres are needed to travel 650 km?

2. How many times must you refuel to travel 2000 km?

3. How many **km** can you travel on 10 **gallons** of fuel using the same car?

N.B. 4.55 litres = 1 gallon

See if you can come up with the solutions to these problems on paper now
and then write the Java code for the solution - your program might run as indicated in the following screenshot


C:\Program Files\Xinox Software\JCreatorV4LE\GE2001.exe
Please enter the capacity of the fuel-tank in litres: 60
Please enter the fuel consumption of the car in km/l: 14
This car can travel 840.00km on a single tank of fuel
How many km do you need to travel? 650
In order to travel a distance of 650.0km you need 46.43 litres of fuel
In order to travel a distance of 2000km it is necessary to refuel 2.38 times
The distance you can travel on 10 gallons of fuel is 637.00km

Solutions

Q1 in the activity was fairly straightforward.

- For each part, can we write an **algorithm** to solve all general problems of this type where the volume of fuel and fuel consumption vary?

  distance = volume of fuel × fuel consumption

  volume of fuel required = distance ÷ fuel consumption

- distance possible on 1 tank of fuel = 60 × 14 = 840 km
- fuel for 650 km = 650 ÷ 14 = 46.428571 litres

  or is it 46? or 46.4? or 46.43? etc.

  Sub-problem = how many digits to display? Should everything be displayed as a decimal fraction? Different programming languages handle display of numbers differently possibly introducing a tricky sub-problem.

Solutions

- Q2 gives the distance to be traveled as 2000km.
- But we already know from Q1 that the distance that 1 tank of fuel allows is 840km. Therefore to travel 2000km requires 2000 ÷ 840 = 2.38 tanks of fuel.
- In general, the number of tanks of fuel required to travel a given distance is given by the algorithm:
  - tanks of fuel required = total distance ÷ distance traveled on 1 tank

- Q3 gives the volume of fuel as 10 gallons. We already know the fuel consumption of the car is 14km/l.
- But …. the units for volume of fuel must match those for fuel consumption in order to obtain the correct distance.

Solutions

- Therefore we must perform a conversion
- We are told that 1 gallon = 4.55 litres.
  Therefore, 10 gallons = 45.5 litres.
- In general, the number of litres that correspond to a certain number of gallons is given by the algorithm:
  - volume in litres = volume in gallons × 4.55
- Therefore, the number of km that can be traveled in this case is given by the formula we used earlier i.e.
  distance = volume of fuel × fuel consumption
- So the distance here is 45.5 x 14 = 637km

A problem-solving strategy

1. Understand the problem
2. Devise a plan to solve the problem (plan an attack)
3. Carry out the plan/attack
4. Assess the result, i.e. look back — check the result and reflect upon it
5. Documenting the solution

Understand the problem

- Read it through several times
- Identify the principal parts of the problem
- Restate/rephrase the problem if necessary, use a different representation (e.g. draw a diagram if applicable)
- What is the **known**?
- What is the **unknown**? Is finding the unknown part of the problem?
- Does the problem have sub-problems?
- How do the parts of the problem relate to each other?

Devise a plan

- Identify all sub-problems within the larger problem
  - Treat each sub-problem as a problem in its own right
- Is it similar to problems you have solved before?
  Many programming problems fall into well-understood patterns that can be reused.
- Solve the easy sub-problems first
- If too difficult, try simplifying the problem and solve the simpler version first

Carrying out the plan

- Once you think you have a firm grasp on the solution, write down the sequence of actions necessary to solve the problem
- Is the ordering of the actions correct?
- Do the actions need to be executed in all circumstances? Some may only need to be carried out if certain conditions are met – the “umbrella” action.
- Do the actions need to be executed only once? Some may need to be repeated – the “emptying the lawnmower” action
- If you are finding understanding a problem too complicated temporarily remove some of the details to see if the simplification gives you a clearer understanding

ACTIVITY

Write down the sequence in which the various actions associated with making a cup of tea for somebody should be carried out

- Do all actions need to be carried out in all circumstances?
- Are some actions dependent on certain conditions being met?
- Do some actions need to be repeated?

Are there any conditional or repetitive tasks involved in making a cup of tea?

Possible Solution!

1. Check that there is sufficient water in the kettle
2. If there is not sufficient water, switch off the power to the kettle, unplug the kettle and fill it with enough water for one person, while making sure the element is covered
3. Plug the kettle back in and switch the power on as well as the kettle itself.
4. As the water is heating, take out a cup, spoon and tea-bag
5. If you don’t know how the person likes their tea, inquire how much milk and sugar they would like
6. If they require milk or sugar, take these out.
7. When the kettle has boiled and switched off, pour sufficient water into the cup, add the tea-bag and leave it rest for about 30 seconds, then squeeze the bag and bin it.
8. Add the required milk and sugar, stir and serve.

Assumptions and Imperfections

- Note how my solution make several assumptions here. I assume that there is a kettle available, that is working, that there is a supply of electrical power, that the kettle is an electrical rather that a gas-based kettle, that there is a supply of cups, spoons, tea-bags, milk and sugar!

- If there were no electrical power available and the kettle is an electric one, then there is no point in even executing the first step of the solution.

- Likewise, if there is no sugar or milk available, there is no point in waiting to step 6 to find this out. Therefore, assumptions can affect how we should organize our solutions for efficiency.

Assessing the results

- Run through your solution (sequence of actions)
  - Ideally, get someone else to work through them – they may just find instructions that mean something quite different from what you intended. It is the differences between what we intend, and how we actually express our intentions that result in logical errors in our programs
  - Does your solution give the correct outcome? If you change some of the values or conditions, do you still get the correct outcome?
  - If the solution is correct, are there any parts of it that seem cumbersome – is there a better way of doing things?
  - Assessing the result is vital to identify logical errors and areas that could be improved

Documenting the solution

- In the software industry, software maintenance is very expensive and the person who wrote the original solution is often unavailable.
- It is vital that programmers learn the discipline of writing good, clear and useful documentation that will help future programmers to read, understand and maintain their programs.
- In addition to your solution write down
  - A general summary of the solution (we use comments for this)
  - Conditions that must be met before the solution can be applied (e.g. before making the tea, we must have some tea-bags or tea-leaves)