How to THINK like a Programmer

Problem Solving for the Bewildered

paul vickers

chapter 2

a strategy for solving problems

Purpose

- This chapter is all about problems and how we solve them
  - Problem-solving is the heart of programming
  - What is a problem?
  - The problem with problem solving
  - A strategy for solving problems
    - Understanding the problem
    - Devising and carrying out the plan
    - Assessing the results
    - Describing what we have learned
    - 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"

- We need a better way...

What is a problem?

- Before we write **any** code we must understand and solve the underlying problem
- A problem exists when
  - We have an (as yet) unreached goal and
  - Several possible ways of attaining that goal
- A programming problem might be something like:
  - “Calculate the cost (including tax and delivery) of a shopping basket of items at an online store and email the customer a receipt”
  - “Whenever the main character in a game is in the castle grounds send the nearest zombie character to fight him”
  - “Rip a track from a CD and save it as an MP3 file”
- The trick is to understand what is required

The problem with problems

- 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 **miles** can you travel on 10 **gallons** of fuel using the same car?

- n.b. 1 US gallon = 128 fluid ounces, 1 UK gallon = 160 fluid ounces, 1 US fluid ounce = 1.04 UK fluid ounces, so 1 UK gallon = 1.20 US gallons!

Problem domain vs. language domain

- The first problem in the activity was straightforward. The 2nd and 3rd were refinements of the first problem
- For each, can we write an algorithm to solve all general problems of this type where the tank capacity and fuel consumption vary?
  - range = capacity × fuel_consumption
  - fuel_required = distance ÷ fuel_consumption
- range = 60 × 14 = 840 km
- fuel for 650 km = 650 ÷ 14 = 46.428571
  - or 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 problem

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. Describe what we have learnt from the process
6. Explain our results (and document our program)

- See Table 2.2 in the book (also at the end of the book in the end papers/inside back cover)

Understand the problem

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

Devise a plan

- Identify all sub-problems
  - Treat each sub-problem as a problem in its own right
  - What are some of the sub-problems of the larger problem “Describe how to get up and ready in the morning”?
- Solve the easy sub-problems first
- If too difficult, try simplifying the problem and solve the simpler version first
- Make a drawing or sketch
- Retell the problem in your own words to someone else

ACTIVITY

Try partitioning (separating out) the problem of getting up in the morning into a few simpler sub-problems (or tasks).

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?
- Does the order rely on certain conditions to be true?
  - Have these conditions come from the problem statement or are they assumptions you have made?
  - Is it possible to verify the assumptions?
- If you are finding understanding a problem too complicated try removing 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 getting up in the morning should be carried out

- Do all actions need to carried out in all circumstances?
- Are some actions dependent on certain conditions being met? (e.g. putting up an umbrella while walking to the corner)
- Do some actions need to be repeated? (e.g. emptying the grass box on a lawnmower)

Are there any optional or conditional tasks involved in getting up in the morning?

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 bugs (defects) in our programs
  - Did they manage to complete the task without asking for help or clarification?
  - Did they misinterpret any of the instructions? If so, did they really misinterpret them or were they correct and you were wrong in the way you wrote the instruction?
  - If you and your friend used the solution, did you both get the same answers? If not, who was wrong, and why? If you did get the same results, were they the correct ones?

ACTIVITY

Get your friend to evaluate/criticize your steps for getting up in the morning that you wrote down earlier

- Did your solution give the right outcome or the correct results?
- If you change some of the values and conditions do you still get the correct answers? If you did get the right answers, did any parts of the solution seem cumbersome? With hindsight, is there a better way of doing things?
- If you got the wrong answers, do you know why? Trace through your solution step by step to see if you can find where it is going wrong

This step of assessing the result is very important as it will highlight errors in the solution and areas that could be improved. You should repeat steps 2, 3, and 4 until you are as confident as you can reasonably be that your solution is the best one you can achieve. Only then should you move on to…

Describing what we have learned

- Having gone through several cycles of devising a plan, carrying out the plan, and assessing the results you should have a list of things you have learnt
- Write down these lessons to build a repertoire of good practice and mistakes
  - You will find this repertoire useful when you solve similar problems in the future

Documenting the solution

- In addition to your solution write down
  - A general summary of the solution
  - Assumptions that have been made which affect the way the solution works
  - Conditions that must be met before the solution can be applied (e.g. before you can apply the solution to getting up in the morning a supply of clothes must be available)

end of chapter 2