BTE2313

Chapter 10: Basic Problem solving

by
Sulastri Abdul Manap
Faculty of Engineering Technology
sulastri@ump.edu.my

Computer Programming – 3 stages

- Computer programming can be divided into three distinct areas of activity:

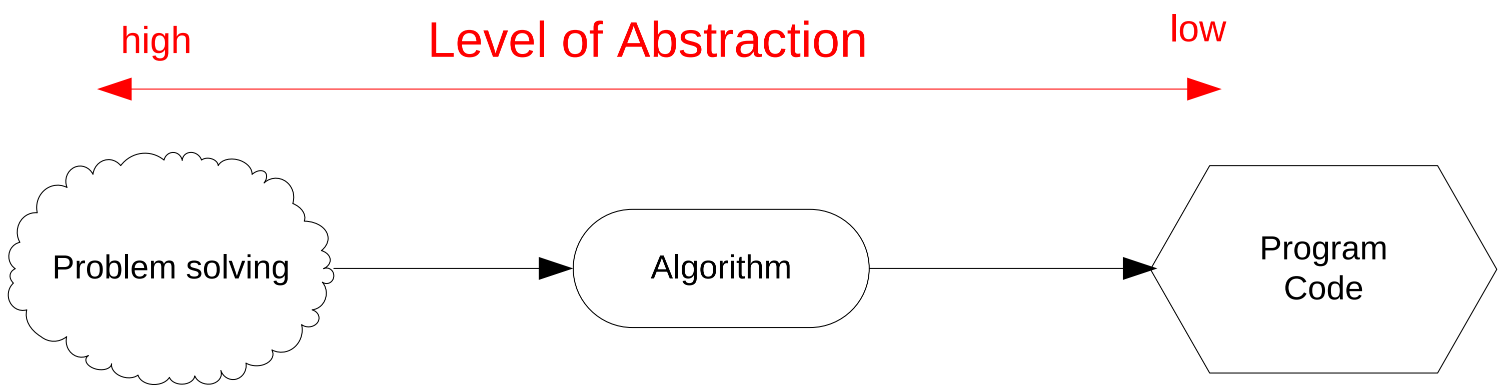
1. Problem definition and problem solving
2. Creating a structured solution (or algorithm)
3. Coding (e.g. Java, Python, C++)

Computer Programming – 3 stages (cont.)

- Example
  - **Specify the problem**
    Customers paying by cash often do not have the correct money. If they pay more than the price of the item, then they need to be given the correct amount of change.
  - **Solve the problem**
    The amount of change returned to the customer will be the amount paid by the customer, minus the price of the goods.
  - **Specify the solution in a structured (algorithm) format**

```
Get Price ;
Get Amount Paid ;
Calculate Change Due (Amount paid – Price) ;
Return Change to Customer;
```

Problem solving, algorithm development and coding



The amount of change returned to the customer will be the amount paid by the customer, minus the price of the goods.

```
Get Price ;
Get Amount Paid ;
Calculate Change Due
(Amount paid - Price) ;
Return Change to
Customer ;
```

```
public static void main(String[]
 args){
Vendor v = new Vendor(400, 10000);
double cashToReturn = v.getChange();
v.returnCash(cashToReturn);
}
. . . . Etc
 . . . . Etc
```

What is a problem?

- There are many different kinds of problems:
  - Why did I break up with my girlfriend?
  - How much tax should my employees be paying?
  - How do I get to UMP Pekan from UMP Gambang?
  - Why is it wrong to lie?
- However, not all kinds of problems can be solved using computers
- For a problem to be soluble through the use of a computer, it must generally:
  - Be technical in nature with **objective** answers that can be solved using **rational** methods
  - Be **well-structured**
  - Contain sufficient information for a solution to be found
  - Contain little, if any, **ambiguity**.

A problem solving framework

- An outline framework for problem solving:
  1. Understand the problem
  2. Devise a plan to solve the problem
  3. Carry out the plan
  4. Assess the result
  5. Describe what has been learned from the process
  6. Document the solution.

(1) Understanding the problem

- To understand a problem
  - We need to read and reread it till we understand every detail
  - We need to dissect the problem into its component parts (e.g. problems and **sub-problems**)
  - We need to remove any **ambiguity**
  - We need to determine our **knowns** and our **unknowns**
  - We need to be aware of any **assumptions** we are making.

(1) Understanding the problem (cont.)

A car has a fuel capacity of 12 litres and a fuel consumption of 45 km per litre. Petrol costs RM2.56 a litre. How far can the car travel on RM16.64 worth of petrol?).

- In the above example,
  - What are the **knowns** and the **unknowns**?
  - Is there any ambiguity?
  - What are the component parts of the problem?
  - What information is extraneous to the problem solution?
  - What assumptions have you made about the problem?

(2) Devise a plan to the solve the problem

- If a problem contains a set of sub-problems, in what order are you going to solve them?
- How are you going to represent the problem:
  - Numerically?
  - Graphically?
  - Tabular data?

(3) Carry out the plan

- Consider the following problem:

In a room with ten people, everyone shakes hands with everyone else exactly once. In total, how many handshakes are there?

- To answer the handshakes problem we could begin by **accounting for all the possibilities** and graphically represent those possibilities in a table.
- This clearly shows us who shakes hand with who (A to J shake hands with everybody except themselves (X)).
- We can then use some basic **mathematical reasoning** to find our solution:

```
number of people = 10
grid size = 10 x 10 = 100
handshakes = (100 - 10)/2 = 45
```

- From here we can abstract a generalizable solution that can easily be turned into an algorithm and eventually into computer code:

$h = ((p^2) - p))/2$

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
A	X									
B		X								
C			X							
D				X						
E					X					
F						X				
G							X			
H								X		
I									X	
J										X

(4) Assessing the results

- It is very unusual when solving complex problems to achieve the correct result first time round. We often need several attempts to get it right.
- To verify our solutions are correct, we need to take a few steps backwards:
  - Was our understanding of the problem correct?
  - Did we overlook anything?
  - Did we choose the correct strategy?
  - Did we employ that strategy correctly?
  - Have we made any incorrect assumptions?
- However, it is often very difficult to spot our own mistakes. It is often better, therefore, to have somebody else verify our solutions for us.

(4) Assessing the results (cont.)

- Sometimes solutions appear correct, but are in fact wrong, due to an initial misunderstanding of a problem
- If you have misunderstood a problem, it does not matter how good a coder you are, your program will not work as it is supposed to.
- Therefore getting the problem-solving part of programming right is absolutely essential if we are to build programs that work as they are supposed to work.

(5) Describing what you have learned

- You can only become a good problem solver by reflecting on your experiences of problem solving.
- Keeping a record of problems you have attempted, your success, failures, the approaches you have used, etc. will:
  - Broaden your problem solving skills/collections
  - Help you to recognize similarities/patterns in problems
  - Help you identify and fix logical or implementation errors
  - Help you to solve problems faster and more effectively

6) Documenting the solution

- Documenting a solution will help you to **generalize** your approach to other similar problems.
- It will also prevent you forgetting how you arrived at your solutions.
- In the long run, it will make you a better problem solver and eventually a better programmer.

EXERCISE

- Write a program that will input kilometers traveled and the hours spent. The program will determine km/hour. This calculation must be done in a function other than main; however , `int main()` will print the result. The function will thus have 2 parameters: kilometers and hours. Output is fixed with 2 decimal point precision.