CS620c Structured Programming Lesson 2

Joe Duffin

Email: Joseph.Duffin@nuim.ie

A Program is..

- A computer program is **a collection of instructions** that describes a task, or set of tasks, to be carried out by a computer.
- A programming language is used to write a computer program.
- The program can then be converted into a format that the computer can understand.
- Computer programs can be categorized along functional lines:
  - e.g. application software, operating systems, video games etc.

A good quality program is…

- Readable

  code is clearly formatted, indented and commented and uses good coding conventions

- Modular

  code is structured appropriately with clear interfaces.

- Robust

  code is able to deal with all possible input data situations without crashing.

In more detail…

- Readable
  - Other programmers as well as the original programmer must be able to understand it since the maintenance phase of a computer project can actually be more expensive than the original development process.
  - Comments should be provided to explain what the program does. Other external documentation can also be useful.
- Modular
  - Programs should be broken down into component parts, where each part is subdivided as necessary. This is the key to developing good algorithms.
- Robust
  - A program should gracefully handle cases when the input to it is not as expected or some other error has occurred. A program should never crash.

Importance of planning

- You need to plan what your program needs to be able to do.
- You need to determine what the component parts are and how to best order them.
- Writing an algorithm helps you to do this.
- ***An algorithm is a detailed sequence of steps that is required to solve a problem.***
- Examples of algorithms include:
  - a set of instructions telling you how to put a kit together e.g. IKEA flatpack
  - a recipe in a cook book

Algorithm

- Features:
  - Input and starting conditions of the system is defined
  - Advance through a finite number of steps
  - Output a result
- Finish

Tip for writing algorithms

- A good rule-of-thumb is to think about
  - What you need to solve the problem (materials)
  - The sequence or order of steps (instructions)

- Write an algorithm to boil an egg
- *Assumption:* You all know how to boil an egg!!!

Boiling an Egg

- Materials:
  - An egg
  - Water
  - A saucepan
  - A stove
  - A match
- Steps:
  1. Put water in the saucepan
  2. Turn on the stove
  3. Boil the water
  4. Put the egg in the water
  5. Wait 3 minutes
  6. Take out the egg from the water

More detailed steps..

1. Open kitchen press door.
2. Remove saucepan.
3. Close press door.
4. Place pot under sink tap.
5. Turn on the cold water.
6. Wait until the pot is ¾ full.
7. Turn off the cold water.
8. Turn burner on high heat.
9. Place pot on stove.
10. Open refrigerator door.
11. Take out the egg carton.
12. Open the egg carton.
13. Remove an egg from the carton.
14. Close the egg carton.

More detailed steps..

15. Place the egg carton back in the refrigerator.
16. Close refrigerator door.
17. Open the cutlery drawer.
18. Remove a large spoon.
19. Close the cutlery drawer.
20. Wait until the water is boiling.
21. When the water is boiling, use the spoon to place the egg in the pot.
22. Wait three minutes until the egg is cooked.
23. Turn off the stove.
24. Remove the egg from the pot.

How much detail?

- Too little detail will leave the algorithm vague and ambiguous
- Too much detail will clutter up the algorithm and is unnecessary

- The key: ***try to break the problem up into fundamental blocks that are simple enough not to require further detail***

Refine and structure

Top down design –> decomposition until you reach primitives

This process is also called “stepwise refinement”

1. Get a Saucepan
   - 1.1 Open kitchen press door.
   - 1.2 Remove saucepan.
   - 1.3 Close press door.
2. Fill with water
   - 2.1 Place pot under sink tap.
   - 2.2 Turn on the cold water.
   - 2.3 Wait until the pot is ¾ full.
   - 2.4 Turn off the cold water.
3. Start stove
   - 3.1 Turn burner on high heat.
   - 3.2 Place pot on stove.

Bottom up design

Start with all the primitives and then start assembling them. Although intuitive it is not always the best approach for most problems (OK sometimes).

Refine and Structure

4. Get an egg
   - 4.1 Open refrigerator.
   - 4.2 Take out the egg carton.
   - 4.3 Open the egg carton.
   - 4.4 Remove an egg from the carton.
   - 4.5 Close the egg carton.
   - 4.6 Place the egg carton back in the refrigerator.
   - 4.7 Close refrigerator door.
5. Get a spoon
   - 5.1 Open the cutlery drawer.
   - 5.2 Remove a large spoon.
   - 5.3 Close the cutlery drawer.
6. Place egg in saucepan
   - 6.1 Wait until the water is boiling.
   - 6.2 When the water is boiling, use the spoon to place the egg in the pot.

Refine and Structure

7. Wait for egg to boil
   7.1 Wait three minutes until the egg is cooked.
8. Finish up!
   8.1 Shut off the stove burner.
   8.2 Remove the egg from the pot.

Your first algorithm

- Write an algorithm to make a cup of tea
  - What materials do you need to carry out the task?
  - What steps are required?

  - Try to refine and structure

Ideas.. ?

- What materials do we need?
- What are the instructions?

Make a cup of Tea

- Materials

Make a cup of Tea

- Instructions

Try to refine this example!

- Try to refine this algorithm into its functional parts

Washing the dishes

- What materials do you need to carry out the task?
  - Dishes
  - Sink
  - Stopper
  - Water
  - Draining board
  - Washing-up liquid
  - A cloth

- What steps are required?

1. Fill sink
   1.1 Put stopper in sink
   1.2 Turn on water
   1.3 Stop filling sink when water is ¾ full
   1.4 Add 2 squirts of washing up liquid
   1.5 Swirl water with your hand

Washing the dishes

2. Wash dishes
   2.1 Wash dishes until all dishes are clean
      2.1.1 Place dishes in sink until no more dishes can fit
      2.1.2 Wash dishes until no more dishes in sink
         2.1.2.1 Use cloth to rub each dish clean
         2.1.2.2 Place dish on draining board
      2.1.3 Check to see if there are any more dishes to wash
         2.1.3.1 If there are more dishes to wash repeat steps 2.1.1 and 2.1.2
         2.1.3.2 Otherwise move on
3. Finish up!
   3.1 Remove stopper from sink
   3.2 When there is no more water in the sink clean the sink with a cloth

Something to think about

- Exercise for this evening which will be graded in tomorrow's lab, think about an algorithm to search a phone book for a person's number. Write down the answers to the following.
  - What materials do you need?
  - What steps would you take?

Summary

- In order to write a good program you need to be able to break up the problem into manageable modular pieces
- You need to know what things you need to carry out the task and you need to know the steps that you need to follow
- A computer is only as good as the instructions it has!
- Throughout the course you should start all your assignments by writing an algorithm
- Give yourself a pat on the back! Although you don’t know it yet, so far we have dealt with very important programming constructs!
- Well done!

Can you answer any of these questions yet?

1. What is an algorithm?
2. What does JVM stand for?
3. What does ‘platform independent’ mean? (research this)
4. How do I create a new Java file using the a text editor?
5. How do I compile it?
6. How do I run it?
7. Are filenames case sensitive?
8. Can I have spaces in file names?
9. What line of code do I write to print the words ‘Java Programming’ to the screen?
10. Name three errors often made when writing programs.

Java – Important Points

- Java is case sensitive
- All statements in Java end with a semi-colon
- There are several types of comments in Java
  - single line comments - //

```
// This is a single line comment
```

  - multiline comments /* ……*/

```
/* This is a multi-line comments, use this type of
   comment when your comment is longer than one
   line */
```

- Indentation is very important. It makes your program easier to read

Dealing with compilation syntax errors

- Check your spelling
- Check case (Java is case sensitive)
- Check brackets (did you close all opened brackets)
- Check semi-colons
- Make use of the debugger (later when we get to use BlueJ)