Edinburgh Napier University

Programming Fundamentals

Tutorial Material

Dr Neil Urquhart

Contents

Unit 1

Introduction

These tutorials are designed to teach you a number of techniques which will improve your coding and scripting skills and build your technical confidence. These are all tried and tested techniques, some of which have been in use for a long time. They are not a substitute for the analysis and design techniques which will be taught in later modules, but they are a means to help you translate your design into lines of working code.

The principle techniques that we will cover are:

- Flowcharts/activity diagrams
- Functional decomposition
- Pseudo code
- Code review

You may or may not use all of these techniques in the future, but they will help improve your understanding has to how code is structured and how we can move from a design that specifies logic to lines of code that implement the same logic.

This document only contains rough outlines of what will be covered, you will need to take notes during the sessions. I would encourage you to bring laptops and other devices for note taking and for diagramming, if you dont have a laptop dont worry!

You might find Microsoft Visio useful for drawing diagrams, a copy can be obtained via DreamSpark at https://softcentre.soc.napier.ac.uk/users.cgi .

Yours,

Neil Urquhart

Unit 2

An introduction to flowcharts

Introduction

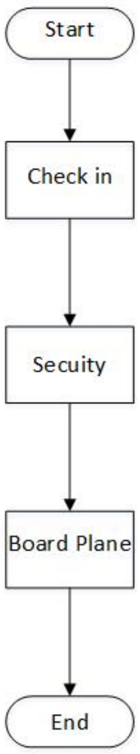
Flowcharts are a means of visually representing work flows, logic or algorithms. A flowchart comprises a number of activities, which are connected in order to determine the order in which the activities take place. As well as processes, decision elements allow us to add elements of selection and repetition.

2.0.1 Flowchart elements

As with program code, three main elements exist within flowcharts, sequence, decision and repetition.

Sequence

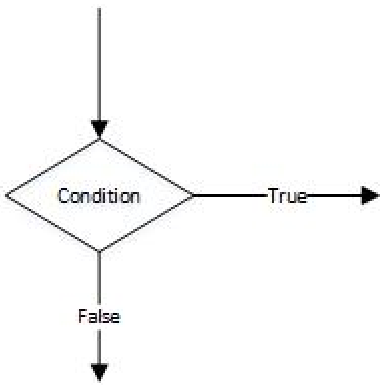
Suppose we wish to model the sequence of activities that take place whilst traveling through an airport. Our analysis suggests that there are three activities that happen in a strict sequence (check in, security and boarding the aircraft). We can represent a sequence of activities as follows:



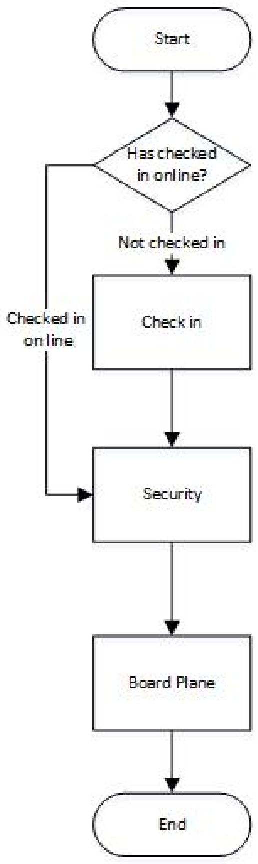
Note the start and end elements which tell us where our logic starts and finishes. In this example the three processes always take place strictly in order, there is only one route through the diagram.

Decision

If we could only represent sequence our models would be very restricted. We can include an element of decision by using the decision symbol, which looks like this:



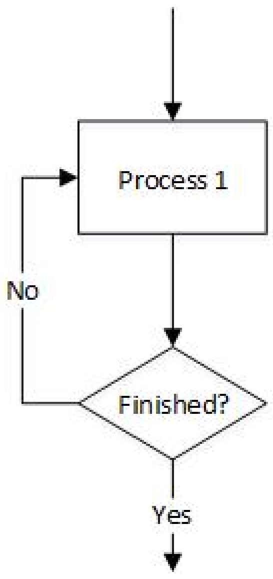
The decision element specifies a condition, which controls which path will be taken. If we revisit our earlier airport example, suppose we added the condition that passengers who have already checked on line in do not need to visit the check in desk:



In our modified diagram we capture the logic that determines that only those who have not checked-in on line go through the check-in process.

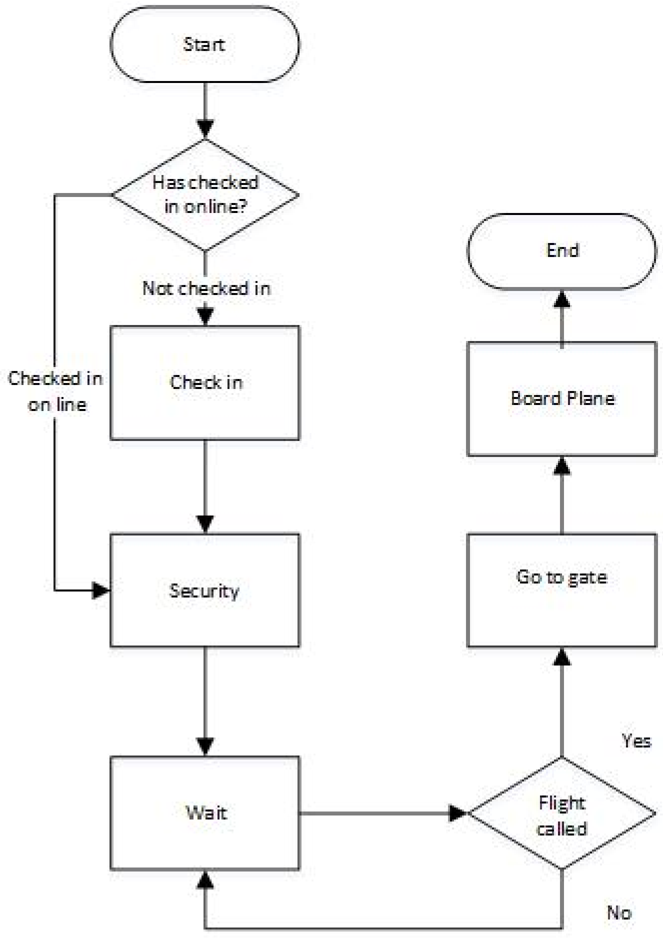
Repetition

There may be some processes that we wish to repeat more than once. We can achieve this through the decision element combined with a process flow that takes us backwards as follows:



We use the the decision element to control the loop, depending on the outcome the logic processes or returns to the previous process.

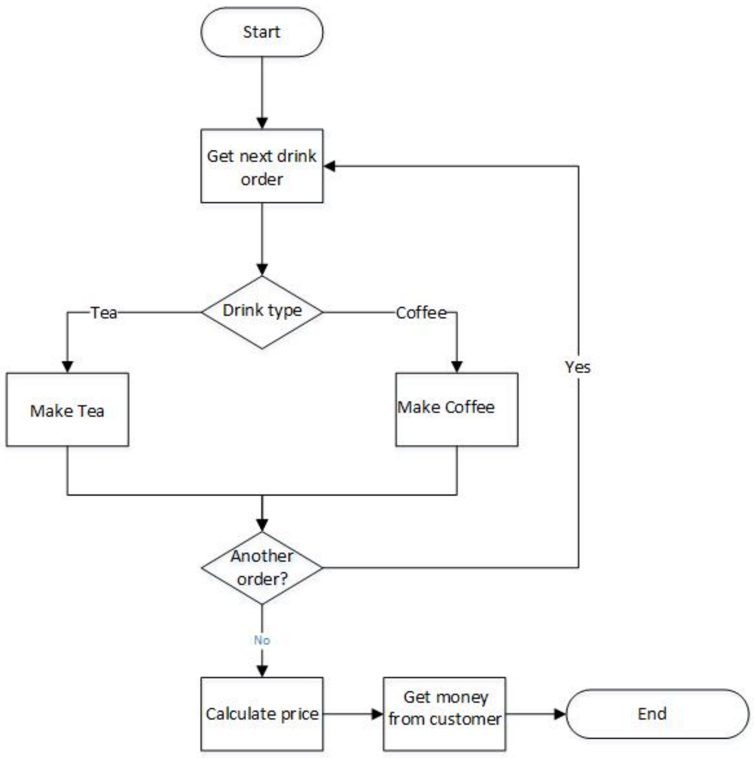
Continuing our airport example, when the passenger has passed through security they must wait until their flight is called, then they should proceed to the appropriate gate and board.



The second decision element is used to show that the wait process is repeated until the flight is called, in this example we can see a decision and loop in use;

2.1 A more complex example

We will now bring together the techniques demonstrated in the previous sections. Consider a canteen that serves teas and coffees. The following diagram illustrates the process by which a customer is served. Each customer may order a number of drinks (repetition), each drink may be either tea or coffee (decision).



2.2 Exercises

Draw a flowchart to illustrate each of the following scenarios:

1. When matriculating a student must have their name and address recorded, should then be added to a class list and finally a matriculation card should be issued.

2. When boarding an airplane each passengers' boarding card must be checked, if the card is not for that flight, they should be denied boarding, otherwise they may proceed. When all of the passengers have been checked a manifest should be printed off.

3. Students sit 6 modules in their 4th year, if any of the marks are less than 50% then the student fails that module and will have a resit letter sent to them. If all of the module marks are greater than or equal to 50% the marks are averaged.

   If the average is greater 70% they should be awarded a first class degree, if the mark is between 60% and 69% they should be awarded a second class degree, if the mark is between 50% and 59% they should be awarded a third class degree.

   All students that are awarded are sent a graduation letter, for their class of degree.

4. Produce a flow chart to illustrate the process by which a call is handled within the following help desk scenario.

All calls to the help desk are logged and allocated a call-id. Calls concerning faults in critical systems are passed to critical systems support team, who allocate an engineer. All other faults are passed to the duty technician who allocates them as follows, printer faults are allocated to the printer technician, applications faults are allocated to a member the applications team and hardware faults are allocated to a member the hardware systems group .

After a call has been allocated, a response will be sent to the user confirming details of the allocation. When the issue has been resolved user will be sent a message asking them to confirm that the issue has been resolved, if it has then call is closed otherwise the call is passed back to the critical systems support team or the duty technician, for re-allocation.

Any call regarding an issue that is not a fault is passed to the help desk supervisor for action.

Unit 3

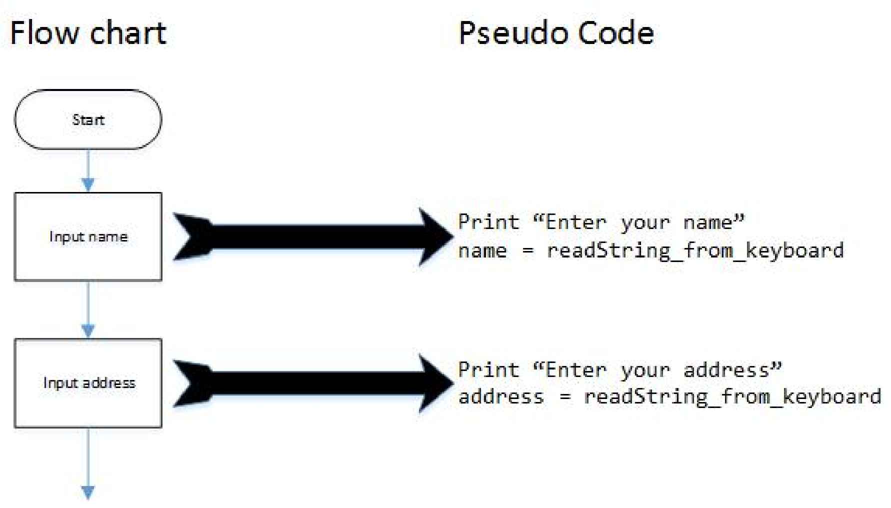
Designing systems using flowcharts

3.1 Physical Systems

Flow charts are an excellent means of modeling the activities and decisions within a system. The previous examples looked at passenger flows through an airport, this is systems modeling at a high level, which some would consider to be a physical system. A physical system as it represents a system that is based on real entities (people, airports etc). When designing large information systems we may wish to analyse and model the physical system that our system will be part of, but at this stage we will concentrate on modeling systems at a lower level. Remember that flowcharts are just one of a range of tools available to the software engineer, they may be used in a number of different ways and may be used as part of other methodologies such as Unified Modeling Language (where they are known as activity diagrams).

3.2 Modeling Software Systems

In this module we will use flow charts as a means of modeling logic that will later be incorporated into code. The boundaries of our systems will be the boundaries of our software, so we do not model processes or decisions that take place outside of the software. Each process within the chart should represent one logical operation in software. For instance if you are asked to input a name and an address you would show this as follows:



```
Print "Enter your name"
name = readString_from_keyboard

Print "Enter your address"
address = readString_from_keyboard
```

ote that the activities can both be easily expressed as pseudo code. From now on we will aim to ensure that any activities in our charts can be expressed in a couple of lines of pseudo code. We will now use the flow chart to help design the *flow of control* within our programs. Let us now take a more complex example:

> A user will be asked for an email address and password, the password will then be encrypted and compared to the encrypted password held in the database. If the passwords match the user's status is set to 'logged in', if the password has not been entered correctly after 3 attempts then the user's status will be set to 'suspended'.

The above is a typical extract from a software specification, before attempting to code it we can begin our design with a flow chart. We can see that there are a number of activities within the system:

> A user will be **asked for an email address and password**, the **password will then be encrypted** and compared to the encrypted password held in the database. If the passwords match the user's **status is set to 'logged in'**, if the password has not been entered correctly after 3 attempts then the user's **status will be set to 'suspended'**.

The highlighted sections of the statement represent statements (in most cases these statements will contain a verb). We can extract these statements from the specifications and draw up a list of operations :

- Input email address
- Input password
- Encrypt password
- Compare to password in database
- Set status to logged in

- Set status to suspended

Note that we separate some of the highlighted sections into two operations, also we reword the statements so they make sense. Each of these operations can be expressed in a couple of lines of pseudo code.

We also note that there is a decision to be made:

> A user will be asked for an email address and password, the password will then be encrypted and **compared to the encrypted password held in the database**. If the passwords match the user's status is set to 'logged in', if the password has not been entered correctly after 3 attempts then the user's status will be set to 'suspended'.

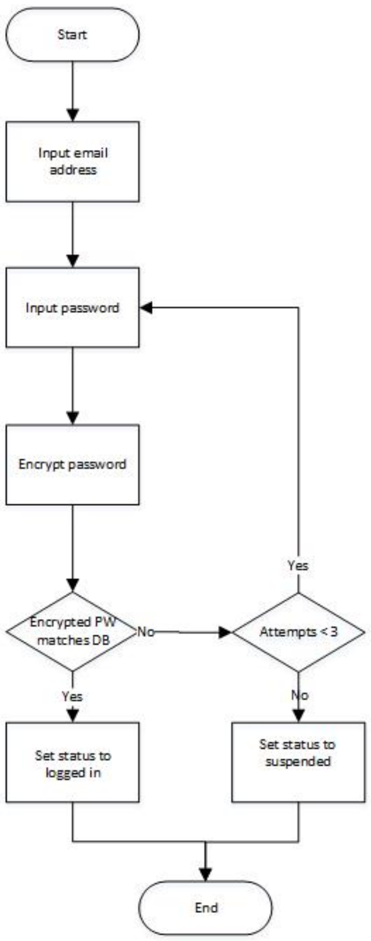
- password same as database

and that there is a loop as well:

> A user will be asked for an email address and password, the password will then be encrypted and compared to the encrypted password held in the database. If the passwords match the user's status is set to 'logged in', if the password has not been entered correctly after **3 attempts** then the user's status will be set to 'suspended'.

- attempts < 3

We can represent the system using the following flowchart:



```
Start

Input email address

Do
          Input password

          Encrypt password

          If (!password same as database)
                    status = logged_In
                    End
          End if
While (attempts < 3)

status = suspended

End
```

Note how some of the activities, such as encrypt password, will eventually encompass several programming operations when we implement the system in C, at this stage we are breaking our system down into logical activities. Later we will break our activities down further so as to match the activities directly to small groups of program statements.

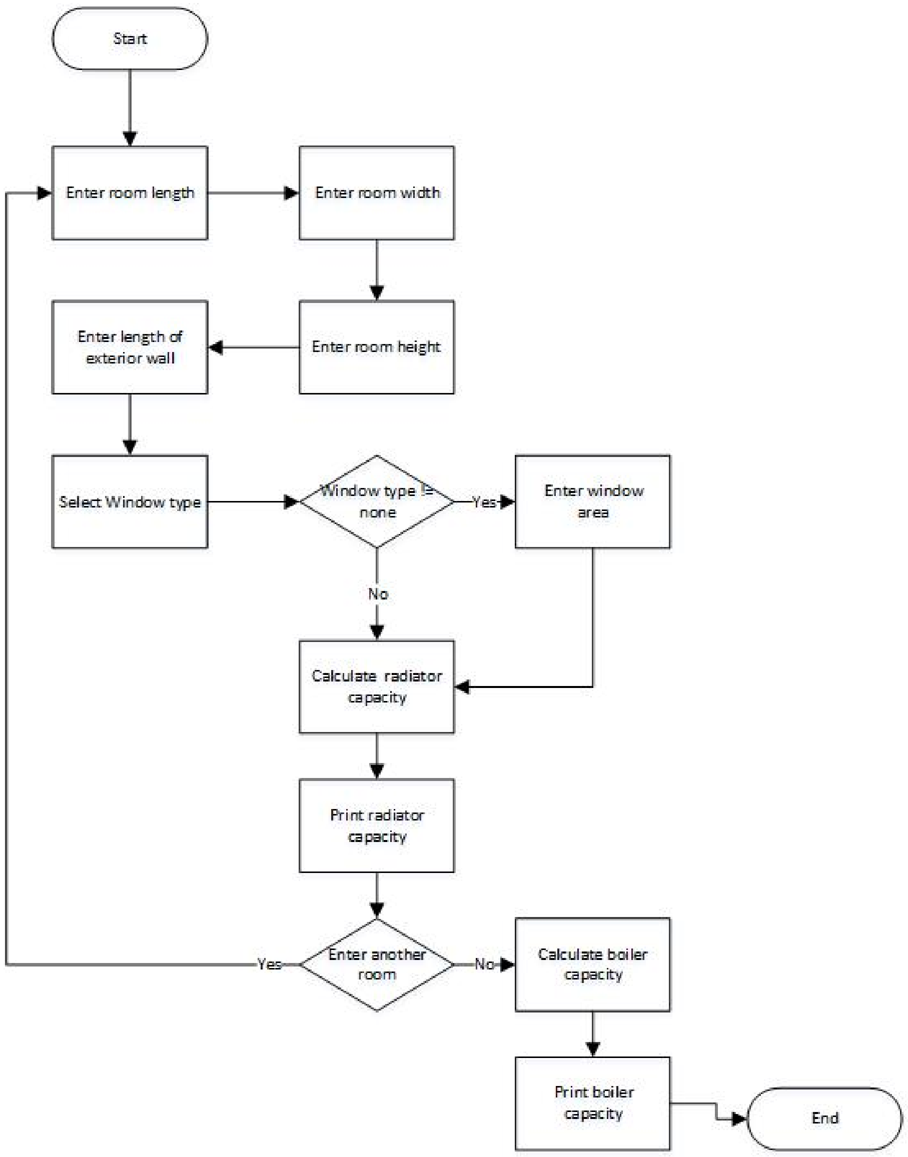
3.3 Exercises

1. Modify the password example so that when a user has entered a wrong password they have the option to have a new password emailed to them. If this option is selected a new password will be generated, the password will then be sent out in an email to the user and then encrypted and stored in the database.

2. The system designed as part of the previous question is not secure enough, when a new password is emailed to a user their status should be changed to 'newpw' and they should then be logged out. When a user logs in successfully, if their status is 'newpw' they should then be invited to change their password. A new password must be entered twice, the passwords will then be validated as follows; both passwords match, password length >8 character and password contains at least one number. If the validation fails then an appropriate error message should be displayed and the user invited to enter the passwords once more.

3. Consider the following specification and flowchart:

   A central heating design system will allow the engineer to input details of each room in a house. For each room the engineer will enter the length, width and height. If the room has external walls they will enter the length of external wall. They must select the window type of each room (from single glazed, double glazed or none), if single or double glazed is selected then the window area should be entered. For each room the system will calculate and print the radiator capacity. After the details of each room have been entered the engineer will be asked if they wish to enter another room. When all of the rooms have been entered then the system will calculate and print the boiler capacity.



   Produce a pseudo code implementation of the system.

4. Consider the following specification for part of an on line auction system.

   Once the user has accessed the web page with details of the item that they wish to bid on the wil be presented with a bid button. Pressing

> the bid button causes a dialog to be displayed requesting the amount to bid. If the user enters an amount ¡0 or non numeric an error message should be displayed and the dialog displayed again. Once a numeric amount has been entered the amount should be compared to the current price, if the new amount is higher then highest bidder should be set to the current user. If the amount entered is <= the current price then a message should be displayed inviting them to bid again by clicking on bid. If the bid button is pressed and there is no user logged in, then the user will be redirected to the login page.

Produce a flowchart describing the actions of the system when the 'Bid' button is clicked.

5. An on line banking application allows users to perform a number of operations on their bank account. The following description covers the actions of the system once the user has logged in and selected an account.

   > A menu will be displayed with the options Show Balance, Show Transactions, Transfer Money and home. The options will work as follows:
   >
   > - Show Balance will retrieve the balance from the database and display it on the page.
   > - Show transactions will retrieve a transaction from the database and display it on the page, this will continue until there are no more transactions to display.
   > - Transfer money, will prompt the user for the sort code and account number of the account they are transferring to. It will then ask for the amount. If the amount is <= 0 then the user will prompted again. Once an amount has been entered then the details of the transfer will be displayed (Account details and amount) and the user will be asked to confirm that they want the transfer to go ahead. If they confirm that the transfer is to go ahead then the transfer details will be sent to the bank's main server.
   > - Home will redirect the user to the bank's home page.
   >
   > Produce a flowchart that describes the displaying of the menu and the actions of the 4 options.

Unit 4

Software design using flowcharts

4.1 Design

As we discussed in the last chapter we will be using flow charts to help us design the flow of control within our software. Designing the flow of control is the element of software engineering that often seems to cause students the most difficulty. It is hoped that by using flow charts this can become a visual process.

Within this section of the module we will adopt a simple software design methodology:

1. Draw flow chart
2. Produce pseudo code
3. Translate pseudo code into program code

The above is only a simple outline of the steps involved, we'll take a look these steps in more detail.

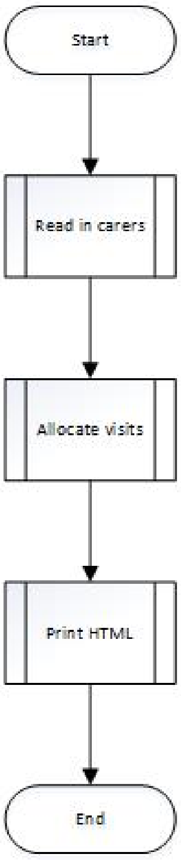
4.2 Functional Decomposition

e know how to draw flow charts and we can use them to design our flow of control. But this can still present us with a complex problem to solve. One means of simplifying our problem is use the technique of *functional decomposition* (sometimes called *algorithmic decomposition*). Decomposition consists of gradually breaking a problem down into component parts, in our case we can start out with a simple high level solution and gradually add more complex sub-solutions into it. For instance consider the following specification:

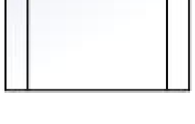
> A system is required to build work schedules for care workers. The system should first load the details of the available carers from a text file. Each carer has a name, payroll number, a base location (the coordinates

> of where that worker is based) and a number of skills, each skill is a single word representing a skill/qualification held by that person. The system will then load a diary file, each entry comprises the location of a visit (given as coordinates), the address, the name of the client and details of any skills required. Each visit is allocated to a carer. Each visit is allocated to the nearest carer (based on the coordinates of the previous visit of base location if it is the first visit) who has matching skills. Once all of the visits have been allocated an HTML file is located that shows the work schedule for each carer.

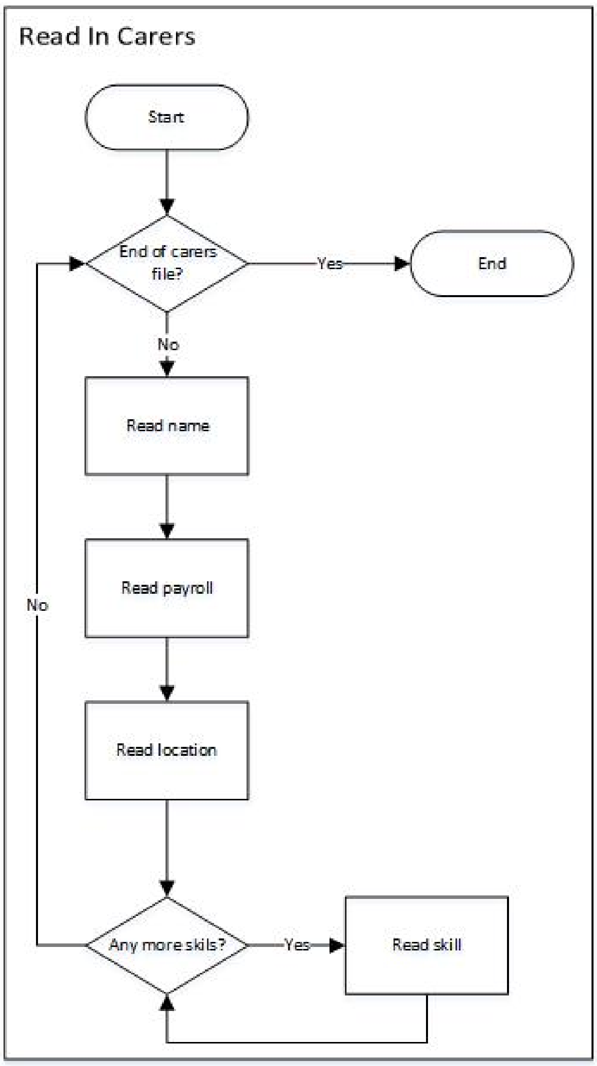
The above specification suggests quite a complex piece of software, certainly more than we should place into one method. We can produce a simple top level design as follows:



We have divided our problem into three processes which we can now consider separately. Each of these processes will have a subprocess. Note that our use of the symbol :



to denote that this process has a sub process. We now have to create three sub processes based upon the above diagram. The first is the process Read In Carers as shown below:



In the above diagram we can see the flow of control for the Read In Carers function. In terms of pseudo code (and our final implementation) we would implement Read In Carers as a subroutine thus:

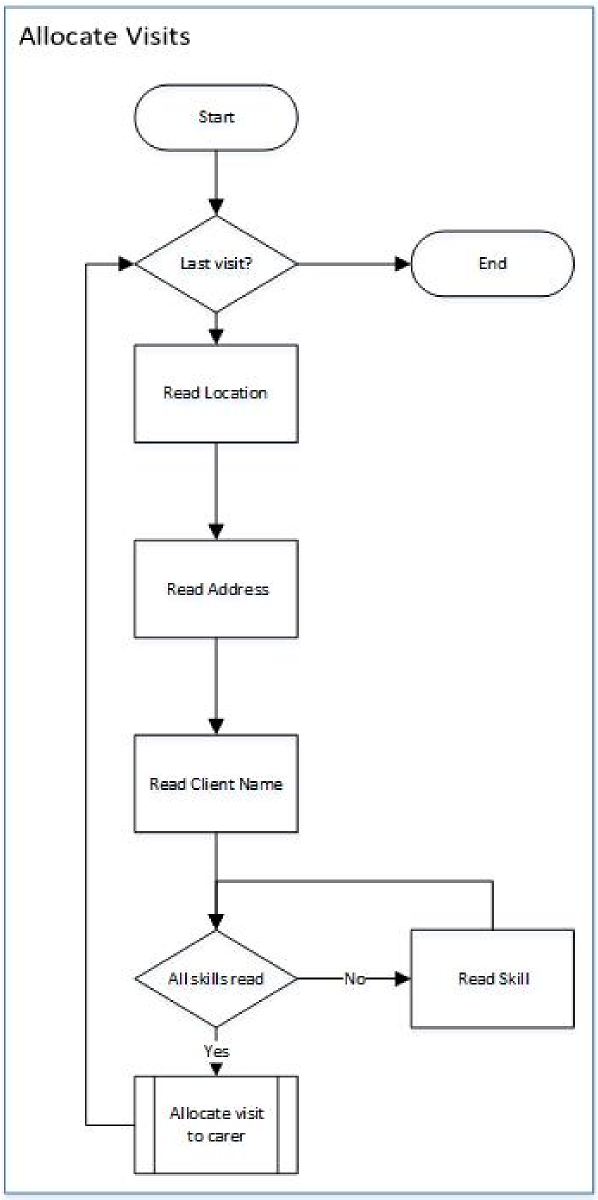
```
Begin
  Read_In_Carers()
  Allocate_visits()
  PrintHTML()
End

Begin Read_In_Carers
  While (! end of carers file) do
    Read Name
    Read Payroll
    Read Location
    While(more skills) do
      Read Skill
    End While
  End while
End
```

Through the use of decomposition we can reduce our program into a series of easily understood methods, there a couple of guidelines that you might wish to keep in mind:

- The method name should reflect its purpose
- The method should only undertake one task (reflected in it's name)
- Further sub methods can be used to break down the method if it's too long

Returning to our example the other sub processes would include:



4.2.1 Useful Pseudo-code

Although our pseudo code does not need to conform to a formal syntax it is useful to have a few standards to work with

4.3 Exercises

1. Produce a flow chart for the sub method Allocate_Visit_to_Carer. This method must consider each carer in turn. If the carer has all of the skills associated with the visit take note of the distance value between the carer and the visit. Allocate the visit to teh carer withe the smallest distance value.

2. Produce A flow chart for PrintHTML() method - Don't worry about HTML formatting at this stage.

3. Produce Pseudo code for the entire system.

Unit 5

Case Study

5.1 introduction

In this tutorial we will bring together some of the techniques that you have learned in previous weeks. In particular we will use some of the C coding skills that you have gained in the lectures and practical sessions combined with our flow charting skills.

5.2 Tasks

5.2.1 specification

Consider the following specification:

> Your program will attempt to calculate the distance traveled when making a journey. It will first allow the user to enter details of the journey, it will then calculate the journey length and then print out a report. Initially the program should accept a journey description (a string) and then a series of latitudes and longitudes (e.g. -0.02137955608, 51.46639893622 note that the user will normally be asked to enter the latitude and longitude separately). When the user has entered all of the latitudes and longitudes they should enter 0 to indicate that they have finished.
>
> Finally your program should print out the journey description and total distance traveled .

5.2.2 Sample Data

To assist you, here are some sample latitudes and longitudes:

```
-0.04247292248,51.48487326661
-0.04374790707,51.48537115578
-0.04641003533,51.48644070181
```

```
-0.04837121115,51.48739960288.
-0.04930547842,51.4877838475
-0.05144180331,51.48869165398
-0.05209244877,51.48897224352
```

5.2.3 Calculating the distance between two points

In order to complete this task you need to be able to calculate the distance between two latitude/longitude points. As with many tasks of this nature, there already exists a means of calculating the distance, this case we use the Haversine formula (seehttp://rosettacode.org/wiki/Haversine_formula#C). Often when faced with the requirement to implement an existing mathematical function we will find that someone has already coded it and placed it on line, saving us having to implement it ourselves.

In this case you can use the dist() function from the following code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define R 6371
#define TO_RAD (3.1415926536 / 180)
double dist(double th1, double ph1, double th2, double ph2)
{
  double dx, dy, dz;
  ph1 -= ph2;
  ph1 *= TO_RAD, th1 *= TO_RAD, th2 *= TO_RAD;

  dz = sin(th1) - sin(th2);
  dx = cos(ph1) * cos(th1) - cos(th2);
  dy = sin(ph1) * cos(th1);
  return asin(sqrt(dx * dx + dy * dy + dz * dz) / 2) * 2 * R;
}

int main()
{
  double d = dist(36.12, -86.67, 33.94, -118.4);
  /* Americans don't know kilometers */
  printf("dist: %.1f km (%.1f mi.)\n", d, d / 1.609344);

  return 0;
}
```

When using someone else's code always take time to test the code first of all. NEVER assume that code that you have legitimately obtained from the Internet is correct!

5.2.4 Development Tasks

Using the skills gained earlier approach this development task as follows:

1. Produce a high level flow chart for your program

2. Using functional decomposition, split any large processes into subprocesses with their own chart
3. Produce a pseudo code listing
4. Translate your pseudo-code into C and compile
5. Test your program

5.3 Reflection

In groups of 4-5 discuss any difficulties that you had with the development task. Compare your designs, code and results.

- Do your designs match? If they are different, how great are the differences?
- Does your code match, can you understand each others code?
- Do your results match? If not, why not?

Unit 6

Coding Standards

6.1 Software Maintenance

When we write software we aim to get out design correct , ensure that our code compiles with no errors and finally we test and debug our program. Is that enough?

In short, no. Most software goes through a life cycle whereby the design, implement, test and release stages are just the start of the life cycle. After an initial version of the software has been released it will require on going maintenance. As you will be aware, software systems don't wear out in the manner that mechanical systems do, e.g. your car will need regular maintenance (such as oil changes) in order to remain running, but software will run indefinitely without degrading [1].

Software maintenance has two fundamental types of operation:

- Fixing bugs, these may be bugs that have been missed in initial testing or they may be bugs that have come to light due to the software being used in a different manner.

- Adding new features, end users will inevitably demand new and exciting features to be added to the software. In most cases such features will be small enough that a new piece of software cannot be justified, so they will be added to the existing system.

Both of the above activities will involve making changes to the source code of an existing system. It is important to remember that the person making these changes may not be the person who initially produced the software. It is important that we write our code in a way which makes it understandable to others (and ourselves!).

[1]This assumes that the underlying hardware continues to function. Mechanical components such as hard disc drives will have a finite lifetime and some electrical components will degrade over time leading to failure

6.2 Tutorial Task: Part 1

Make a note of the time that you are starting this section at *before* you look at the code below.

Have a look at the following function, it is used to print/format invoices for a point of sale system:

```
public int print(String c, String[] d, int[] q, int[] uc){
  int[] pr = new int [uc.length];

  for(int x=0; x < uc.length;x++)
    pr[x] = uc[x] * q[x];

  System.out.println("I N V O I C E \n");
  System.out.println("Item\tQty\tCost(p)\n");

  if(!c.equals("")){
    System.out.println("Customer: "+c+"\n");
  }

  String lines = "";
  for (int z=0; z  < d.length; z++){
    lines = lines + d[z] + "\tx " + q[z]+ "\t = "+pr[z]+"\n";
  }

  System.out.print(lines+"\n");
  int t=0;
  for(int x=0; x < pr.length;x++)
    t = t + pr[x];

  System.out.println("Total \t\t = "+t);
  t=t+(int)(t*0.2);
  System.out.println("Total+VAT \t = "+t);
  return t;
}
```

When run the function produces formatted output like this:

```
I N V O I C E

Item  Qty Cost(p)

Customer: Bob T. Builder

Screw x 100  = 300
Nail  x 150  = 300
Hammer  x 2  = 1100
Saw x 1  = 799
Drill x 1  = 3499

Total      = 5998
Total+VAT    = 7197
```

Now attempt the two tasks given below. Mark your changes on the code above.

Tasks:

1. Indicate on the above listing how you would modify the VAT rate to 15
2. Indicate on the above listing how you would modify the software to print the unit price as well as the total price e.g.: Screw (3) x 100 = 300

Make a note of roughly how long it has taken you to get to this stage.

6.3 Tutorial Task: Part 2

As in part 1, make a note of your starting time before going any further.

The following code performs in exactly the same manner as the previous code:

```java

/**
 * Invoice printer
 * Written by Neil Urquhart 27/2/15
 */

private static  final double VAT =0.2;
//VAT Rate is set to 20%

public static int printInvoice(String custName, String[] orderDescription, int[] quantities, int[] unitCost){
/*
 * Print Invoice to the console
 * Parameters:
 *  custName: the name to appear on the invoice
 *  orderDescription[]: an array of strings describing the items for the invoice
 *  quantities[]: an array of ints denoting the quantities of each item being
 *   ordered
 *  unitCost[]: the unit cost (in pence) of each item being ordered
 *
 *   Returns the total cost (including VAT) in pence
 */

//Print the invoice header
  printHeader(custName);

  //Print the items line by line + return total cost
  int totalPrice = printItems(orderDescription, quantities, unitCost);

  //Print the invoice footer
  totalPrice = printFooter(totalPrice);
  return totalPrice;
}

private static int printFooter(int totalPrice) {
  //Print the invoice footer, with total
  //Returns totalPrice with VAT

  System.out.println("\nTotal \t\t = "+totalPrice);
  totalPrice= addVAT(totalPrice);
  System.out.println("Total+VAT \t = "+totalPrice);
  return totalPrice;
}

private static int addVAT(int cost){
    //Add VAT   tost
    return cost+(int)(cost*VAT);
}

private static int printItems(String[] orderDescription, int[] quantities,int[] unitCost) {
  //Prameters as for PrintIvoice()
  //Returns total price (ex VAT)

  int totalPrice =0;//Total cost of items bought
  for (int z=0; z  < orderDescription.length; z++){//Loop through
     ↪ each item
           int price = unitCost[z] * quantities[z];//Calculate
              ↪ price for each
             //item
     totalPrice = totalPrice + price;
     //Print line on invoice
     System.out.println(orderDescription[z] + "\tx " + quantities[z
        ↪ ]+ "\t = "+price);
  }

  return totalPrice;
}

private static void printHeader(String custName) {
  //Print invoice header
  System.out.println("I N V O I C E \n");
  System.out.println("Item\tQty\tCost(p)\n");

  if(!custName.equals("")){
    System.out.println("Customer: "+custName+"\n");
  }
}
```

Have a look at the above code, it performs exactly the same function (the output is identical).

Now undertake tasks 1 and 2 on the above example. Note the time it has taken you to complete task 2.

6.4 Concluding tasks

In groups of 4-5 discuss the following points, when you have finished you should report back to the rest of the class.

1. Highlight the main aspects of the coding style that have changed between the two examples
2. Suggest a set of coding guidelines that would help improve the readability of your code.

As a class produce a set of guidelines that will improve the readability of your code.

Unit 7

Code reviewing

Introduction

In chapter 6 you came up with coding standards, in this tutorial we will put these standards into practice. You will need the printed listings of the program that wrote for activity 5 (the distance calculator).

7.1 Code Extracts

The following code extracts are from some code that I have been writing, they are not particularly well written. Look at each extract and suggest stylistic improvements, write your suggestions on this worksheet. (I know that you're not familiar with the context and I know that some of the constructs will be new to you, but you need to get used to looking at and understanding code that you didn't write):

7.1.1 Example A

```java
private static void init(){
  try{
  FileInputStream fis = new FileInputStream("C:\\git\\vrptw\\jVrptw
     ↪ \\postcodes\\postCodeCache.ser");
  ObjectInputStream ois = new ObjectInputStream(fis);
  cache = (HashMap<String,Loc>) ois.readObject();
  ois.close();
  }catch(Exception e){
  cache = new HashMap<String,Loc>();
  }
}
```

7.1.2 Example B

```java
private static Loc get(String postCode){
  postCode = postCode.trim();
  postCode = postCode.toUpperCase();
  String area = "";//postCode.substring(0,2);
  for (int x=0; x < postCode.length();x++){
    if (Character.isLetter(postCode.charAt(x)))
     area = area + postCode.charAt(x);
  else
    x= postCode.length();
}

//Fix for London codes
if (area.equals("SE")||area.equals("SW")||area.equals("W")||area.
   ↪ equals("WC")||area.equals("EC")||area.equals("E")||area.
   ↪ equals("N")||area.equals("NW")) {area = "London";}
try{
  BufferedReader reader = new BufferedReader(new FileReader("C:\\
     ↪ git\\vrptw\\jVrptw\\postcodes\\"+area+".csv"));
  String line = null;
  while ((line = reader.readLine()) != null) {
    if (line.contains(postCode)){
      //Loc res  = new Loc();
      String[] items = line.split(",");

      Loc res = new Loc(postCode,Double.parseDouble(items[1]),
         ↪ Double.parseDouble(items[2]));

      reader.close();
      return res;
      }
  }
  reader.close();
  return null;
}catch(Exception e){
  return null;
  }
}
```

7.1.3 Example C

```java
public void calcDistCost(){
  co2Cost=-1;
  distCost=0;
  int prev = Instance.depot;
  for (int i=0; i < pheno.size();i++){
    Route r = pheno.get(i);
    distCost = distCost + this.calcRouteCost(r);//,i);
    }
  }
```

7.2 Your Task

Look at the three code examples given above and suggest ways in which they could be improved. You're trying to alter what the code does, but simply to make it more readable and understandable. A good starting point is to apply the coding standards that you came up with in chapter 6.

Once you have completed this individually, compare and reflect on your findings in groups of 4 or 5. Each group should report back to the rest of the class on the exercise. Areas that you might want to cover include:

- How appropriate the coding standards devised previously were.
- Any updates to the coding standards as a result of this exercise.
- How easily understood were the code examples?
- Would this exercise have been easier if you could have spoken to the individual who wrote the code?

ow take your distance calculators (as coded in section 5) and swap them with another member of the class. Review the code for the example that you have just been given and then discuss your recomendations with the author of the code.

Unit 8

Revision Quiz

The revision quiz that accompanies these materials may be found in module Moodle site.