©PCooke

2 Unit Preliminary SDD

Introduction to Structured Algorithms

What is Structured Programming?

Structured programming is based on the Von Numan architecture, where variable are setup and used to hold data and the 3 standard programming constructs are used to create the algorithm.

Software Development Cycle

When using the development cycle to produce software to solve a problem, a number of steps are undertaken (as seen in unit 2)

1. **Defining** and understanding the **problem**. This where the problem to be solved is defined and an understanding of the problem is developed and modelled
2. **Planning** and designing the solution. This is where a solution is designed and tools are used to document is solution
3. **Implementation (Building)**, this is where the algorithm is coded (implemented) into a computer language
4. **Testing & Evaluation (Checking)** , which is the testing of the software which occurs throughout the process
5. **Maintaining (Modifying)**, which is the operation and maintenance of the program when being used.

*(In this unit of work we will focus on gaining an understanding of **the first 3 stages** of the cycle, Defining, Planning and Implementation stages, including data types and control structures and how to create simple algorithms and php code.)*

Methodology for Developing Algorithms

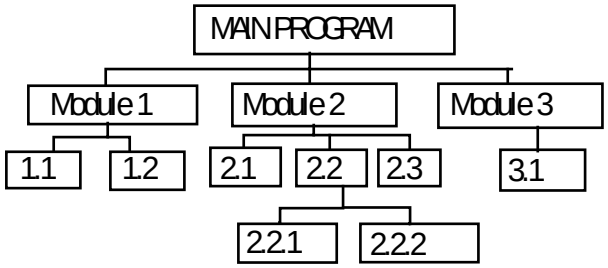
Top Down Design

This method is used to create a solution for a large problem, by dividing it up into smaller problems, which are **sub-programs** or sub-modules. This process of creating a solution by creating sub-programs is called "top down design". It is a systematic approach to the development of the solution

The advantages of using top down design include:

- Each module can be solved individually reducing the complexity of the problem.
- The algorithm is functional the whole time and as more modules are completed the algorithm becomes more operational.

The continual process of gradually completing all sub-modules from the main program down to the bottom module is called **step wise refinement**.



An example of a problem broken into modules.

Activity: Discuss an example of a problem which has been broken down into modules, using top down design.

Defining the problem

Problem definition

This is where the specification and requirements for the problem to solved is set. It involves working out what the problem is and having a full understanding of the problem.

The purpose is to produce a detailed description of the external behaviour of the system. The resulting document is called a requirements specification.

The inputs and outputs are some of the required specifications that define the external behaviour of the system.

Planning

The planning stage consists of analysing and designing a solution to enable the building of the problem. This is a very important stage because many problems with program originate from a lack of analysis and planning of the problem.

There are many different types of planning, analysing and design documents and tools that we will look at over the next 2 years.

At this stage we be using:-

- ❑ IPO charts to analyse the problem
- ❑ Data Dictionary to identify the variable and data types used to hold the data
- ❑ Algorithms to describe a solution to the problem
- ❑ Desk checks and walk throughs to test the algorithm.

IPO Chart

IPO charts are used to analyse the input, output and processes involved in the problem.

Input Data $\Rightarrow$ Processing Data $\Rightarrow$ Output Data

E.g. An IPO Chart for calculating the area of a rectangle

Input	Processes	Output
Base Height	Area = Base * Height	Area

Output specifications

It is easier to determine the output of a problem before the input because the input may rely on what the output determined.
You need to ask questions like:

- What output is required?
- What format is needed?
- How will it be presented?
- You should design what the output screen will look like

Input specification

Ask the questions like:

- What type of input is needed to produce the output?
- Where does the data come from?
- What is required to achieve the output?
- What is the nature and quantity of the input data?
- You should design what the input screen will look like

Processing rules

These are the precise processing operations needed to produce the output from the input data. It usually involves:-

- Formulas needed (eg for calculations)
- The rules or decisions (eg the logic of the algorithm)
- The variables using this program should be defined

Variables

A variable is a name given to a data storage location. Data used by a program needs to be held in the computer's memory. Variables are used to hold the data temporarily while the program is being executed. There are different types of data variables can be set up to hold, including numbers and text.

Data Dictionary

List the variable names and their data types which will be used by the problem. We will look at this in detail later in data types.

Test Data

The test data that is created at this stage helps to define the problem and will be used later to test the algorithm and the final code.

Representing the plan (design) using an Algorithm

This stage involves creating a solution to the problem. This solution can be represented using algorithms, in the form of pseudocode or flowcharts. Top down design and stepwise refinement techniques are used on larger problems.

Understanding and determining a solution to a problem can come from looking for pattens, working back from the output to the input, studying similar problems, consulting others, etc.

A few rules for producing a good algorithm are

- Organisation (produce a good problem definition and analysis)
- Use building blocks (sections of algorithms from previous problems) to build up a solution.

Algorithms

Definitions:

Here are 2 definitions for an algorithm

An algorithm is a set of precise (step-by-step) instructions which solve a particular problem within a finite number of unambiguous steps.

"An algorithm is a formula, a recipe, a step-by-step procedure to be followed in order to obtain the solution to a problem.
To be useful as a basis for writing a program, the algorithm must:
(1) arrive at a correct solution within a finite time;
(2) be clear, precise and unambiguous;
(3) be in a format which lends itself to an elegant implementation in a programming language." Peter Juliff, *Program Design*, 1990

In most situations there **are more** than one possible algorithm that can solve a particular problem. The best solution (algorithm) is often a personal choice.

Exercise: A cooking recipe is an example of an algorithm you would across in your day-to-day life, list some others.

_____

_____

_____

Question: Are all the algorithms listed above easy to solve?
If not, why?

_____

_____

_____

Algorithms and computers are closely related because algorithms are used to provide computer solutions, eg. algorithms are used to describe the steps that software must follow to run successfully.

There are a number of methods available for describing and documenting an algorithms, these include:- Nassi-Schneiderman diagrams,

_____ ,
_____ , and
_____

The two methods which are used in this course that you must be able to both read and write are

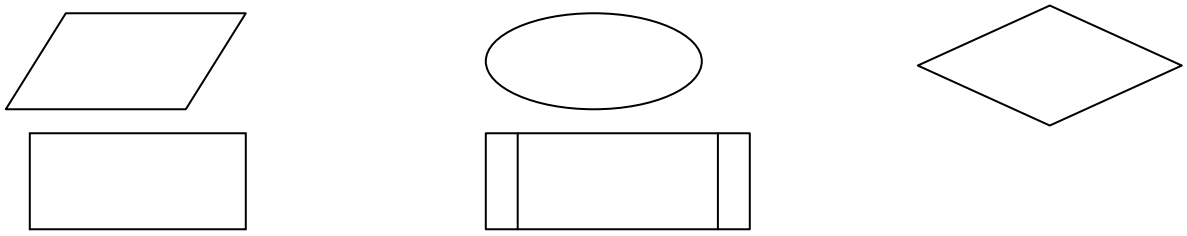
_____ , and
_____ .

Each method of describing an algorithm must be
Understandable,
precise,
unambiguous and
clear.

To make this possible there are **structures** and **rules** for each method that should be followed to allow anyone reading the algorithm to understand it.

Flowcharts
Flowcharts are diagrams, which have instructions inside symbols (shown later) which are connected by arrowed lines. Flowcharts flow from top to bottom, and should fit onto one page. If a flowchart does not fit onto one page it should be broken into smaller modules (sub algorithms). The line of a flowchart should never cross. Each symbol has only one entry point and one exit point (except decision symbols which have two exit points).

Label the five symbols used in flowcharts,



Pseudocode
Pseudocode uses structured English to describe the algorithms. Statements are written in free-flowing English text, although some key words (normally written in UPPERCASE, eg BEGIN) are defined to represent control structures. Sub programs/modules are represented by underlining the word.

Algorithm Constructs (Control Structures)
There are 3 main standard constructs that are used in algorithms. Every algorithm is built up from the 3 constructs listed below.

1. **Sequence**
   Sequence structure is when a series of processes are carried out one by one with no decisions or loops. This is the most common form.

2. **Selection**
   - Binary selection (IF-THEN-ELSE)
   - Multiway selection (CASEWHERE)

   Selection structure asks a question and provides 2 or more alternatives, depending upon the condition asked.

3. **Iteration**
   - Pre-test (WHILE LOOP)
   - Post-test (REPEAT UNTIL LOOP)
   - FOR NEXT LOOP

   Iteration (or Repetition) structure has a loop, where a set of instructions are repeated until/while a condition is satisfied. This condition can be tested **before** (Pre-test, While loop, guarded loop) or **after** (Post-test, repeat-until loop, unguarded loop) the instructions are undertaken once.

Control Structures
For the following examples constructs, write down the **construct type** and create a corresponding flowchart for the corresponding pseudocode.

1. _____

Pseudocode ***Flowcharts***

BEGIN
Get length and height
Area = length * height
Print Area
END

2._____

Pseudocode ***Flowcharts***

IF *age <18* **THEN**
Print ‘too young’
ELSE
Print ‘Old enough’
ENDIF

OR

Pseudocode ***Flowcharts***

IF *Sex = male* **THEN**
Enter male toilet
ENDIF

3, _____

Pseudocode ***Flowcharts***

CASEWHERE *colour*
= green: Proceed
= amber: stop car
= red: stop car
OTHERWISE
Proceed with caution
ENDCASE

4._____

Pseudocode ***Flowcharts***

WHILE *radius > 0*
Get radius
Area = radius * 3.14
Print area
ENDWHILE

5. _____

Pseudocode ***Flowcharts***

REPEAT
Enter age
Totalage = totalage + age
UNTIL *totalage > 100*

PSDD RECTANGLE PROBLEM

The following is the documentation for a whole solution.

Problem

Calculate out the area and perimeter of a rectangle, given the length and width as input data.

IPO CHART

INPUT	PROCESSING	OUTPUT
Length (of the rectangle) Width (of the rectangle)	If length <= 0 Error If width < 0 Error Calculate Area = length * width Perimeter = 2 * (length + width)	Error Error Area of rectangle Perimeter of rectangle

DATA DICTIONARY

Variable	Data	Entry, Calculation, Constant	Description of Variable
Length	Real	Entry	Length entered from keyboard.
Width	Real	Entry	Width entered from keyboard.
Area	Real	Calculation	Area of the Rectangle, calculated from input
Perimeter	Real	Calculation	Rectangle of the Rectangle, calculated from input

TEST DATA

Length	Width	Area	Perimeter	Output
0	0	---	---	Error
-1	7	---	---	Error
2	-4	---	---	Error
5	3	15	16	
0	7	---	---	Error
10	15	150	50	

ALGORITHM

PSEUDO CODE

```
BEGIN
        Ask for length of rectangle
        Enter length
        Enter width
        IF length <= 0 OR length <=0 THEN
                Print Error incorrect entry, program will finish
        ELSE
                Area = length * width
                Print Area
                Perimeter = 2 * (length + width)
                Print Perimeter
        ENDIF
END q
```

ACTIVITY: Write a flowchart for the above algorithm

PHP CODE

```
<h2> Enter Rectange shape details</h2>
<form action="" method="post" name="form">
        <h5>Width: </h5> <input name='Width' type='text' >
        <h5>Height: </h5> <input name='Height' type='text' > <br>
        <input type="submit" name="Submit" value="Submit">
</form>

<?php
        $len1 = $_POST['Width'] ;
        $len2 = $_POST['Height'] ;
        IF (($len1 <= 0) OR ($len2 <=0)){
                echo "<br> Incorrect rectangle entry";
        }ELSE{
                echo "<h2> Rectangle Results </h2>";
                echo "<h5> Width = $len1 </h5>";
                echo "<h5> Height = $len2 </h5>";
                $area = $len1 * $len2;
                echo "<h2> The area is: $area<br></h2>";
                $perimeter = 2 * ($len1 + $len2);
                echo "<h2> The Perimeter is: $perimeter<br></h2>";
        }
?>
```

ACTIVITY

1. Run the program.
   (a) Use the test data supplied to check that the program works as expected
   (b) When you run the program for the first time what message appears on the screen. Outline why this may happen

2. Outline what each of the following sections from the code does/means
   - <h2>
   - $_POST[ ]
   -

©PCooke

Example ALGORITHM

Create the flowchart

PROBLEM STATEMENT

```
BEGIN
    ItemValue = $2.00
    TotalCoinsV = 0
    Finished = No

    REPEAT
        Get value of coin entered ; coinV
        TotalCoinsV = TotalCoinsV + coinV
        IF (TotalCoinsV  <  ItemValue )   THEN
              LessV  =  ItemValue  -  TotalCoinsV
              PRINT "You still need to pay ;  LessV "
        ENDIF
        IF (TotalCoinsV  =  ItemValue )   THEN
              Dispense ITEM
              Finished = Yes
        ENDIF
        IF (TotalCoinsV  >   ItemValue )   THEN
              Dispense ITEM
              ReturnCoins  =  TotalCoinsV  - ItemValue
              Dispense    ReturnCoins
              Finished = Yes
        ENDIF
    UNTIL   ( Finished = Yes )
END
```

WHAT CONTROL STRUCTURES ARE USED?

IPO CHART

INPUT	PROCESS	OUTPUT

DATA DICTIONARY

DATA	TYPE	SIZE	EXAMPLE