AQA Computer Science AS-Level

3.1.2 Procedural-oriented programming

Intermediate Notes

Specification:

3.1.2.1 Structured programming:

Understand the structured approach to program design and construction

Be able to construct and use hierarchy charts when designing programs

Be able to explain the advantages of the structured approach

The procedural programming paradigm

Programs written in the procedural programming paradigm are formed from sequences of instructions that are executed in the order in which they appear. Procedures like functions and subroutines form parts of the program and can be called from anywhere within the program.

Data is stored in procedural programs by constants and variables. A data structure is said to have a global scope if it can be accessed from all parts of the program and a local scope if it is only accessible from the structure within which it is declared.

Most of the programs that you may have written are likely to have been procedural.

The structured approach

Using the structured approach to program design and construction keeps programs easy to understand and manage. Four basic structures are used: assignment, sequence, selection and iteration.

Synoptic Link

Assignment, sequence, selection and iteration are defined in the notes for **abstraction and automation** under **theory of computation**.

Structured programs are said to be designed from the top down, meaning that the most important elements of a problem are broken down into smaller tasks, each of which can be solved in a block of code such as a procedure or module which goes on to form part of the overall solution.

Advantages of the structured approach

Designing a program from the top down makes maintaining the program easier as navigation of different elements of the overall solution is improved.

When a program is split into modules, testing can be carried out on the individual modules before they are combined to form the overall solution. This means that individual parts of the program can be tested before all of the other parts are ready.

Hierarchy charts

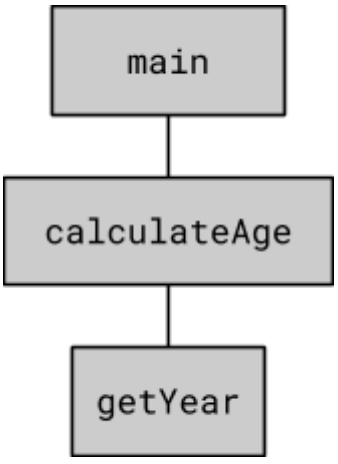
A hierarchy chart graphically represents the structure of a structured program. Each procedure is displayed as a rectangle which is connected to any other procedures that are used within it.

Example

```
SUBROUTINE Main()
  name ← INPUT
  yearBorn ← INPUT
  age ← calculateAge(yearBorn)
  IF age > 17 THEN
    OUTPUT name + "can drive"
  ELSE
    OUTPUT name + "can't drive"
  END IF
END SUBROUTINE
```

```
FUNCTION calculateAge(year)
  yearNow ← getYear()
  age ← yearNow - year
  RETURN age
END FUNCTION
```

```
FUNCTION getYear()
  RETURN system.year
END FUNCTION
```



The three procedures above (`main`, `calculateAge` and `getYear`) form a program which could be represented by the hierarchy chart on the left.

Each rectangle represents a part of the overall program. The lines between the rectangles show the relationships that exist between the different parts of the program.

In more complicated programs, it's possible for each rectangle to be linked to more than one other. This occurs when a procedure calls more than one other procedure.