AQA Computer Science A-Level

3.1.2 Procedural-oriented programming

Advanced 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, by other procedures or recursively.

Recursion

The process in which a block of code that is defined in terms of itself makes a call to itself.

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. If all of a program's code is contained within the same module, finding the specific line of code that needs fixing can be incredibly difficult - especially in large projects.

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. Furthermore, development can be split over a team of developers each of which is assigned a different module to work on.

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

These procedures form part of a program that allows a user to play a game.

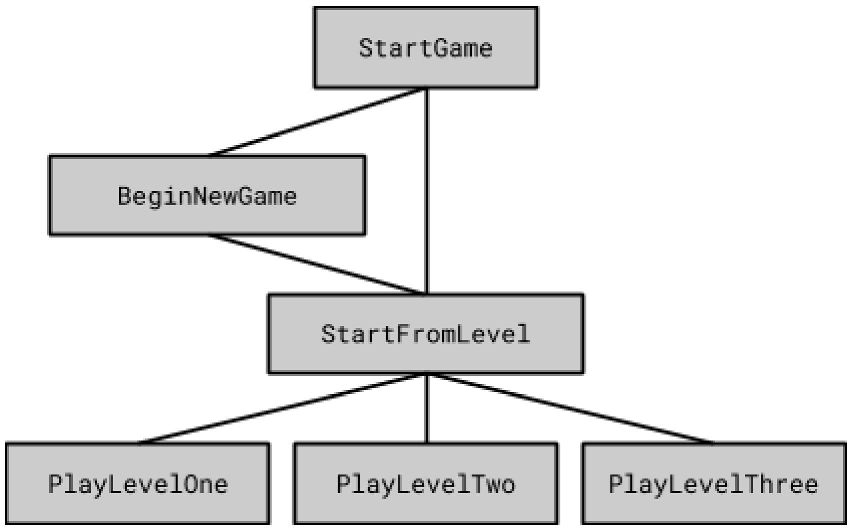
```
SUBROUTINE StartGame()
  level ← INPUT
  IF level = 1 THEN
    BeginNewGame()
  ELSE
    StartFromLevel(level)
  END IF
END SUBROUTINE
```

```
SUBROUTINE BeginNewGame()
  username ← INPUT
  OUTPUT "Hello" + username
  StartFromLevel(1)
END SUBROUTINE
```

```
SUBROUTINE StartFromLevel(number)
  IF number = 1 THEN
    PlayLevelOne()
  ELSEIF number = 2 THEN
    PlayLevelTwo()
  ELSEIF number = 3 THEN
    PlayLevelThree()
  ELSE
    OUTPUT "No such level"
  END IF
END SUBROUTINE
```

The three procedures above form a program which could be represented by the hierarchy chart below.

Each rectangle in the hierarchy chart represents a procedure in the program. The lines between the rectangles show the relationships that exist between the different procedures.


StartGame
BeginNewGame
StartFromLevel
PlayLevelOne
PlayLevelTwo
PlayLevelThree