OCR Computer Science A Level

1.2.4 Types of Programming Language

Concise Notes

Specification:

1.2.4 a)

- **Programming paradigms**
  - Need for these paradigms
  - Characteristics of these paradigms

1.2.4 b)

- **Procedural languages**

1.2.4 c)

- **Assembly language**
  - Following LMC programs
  - Writing LMC programs

1.2.4 d)

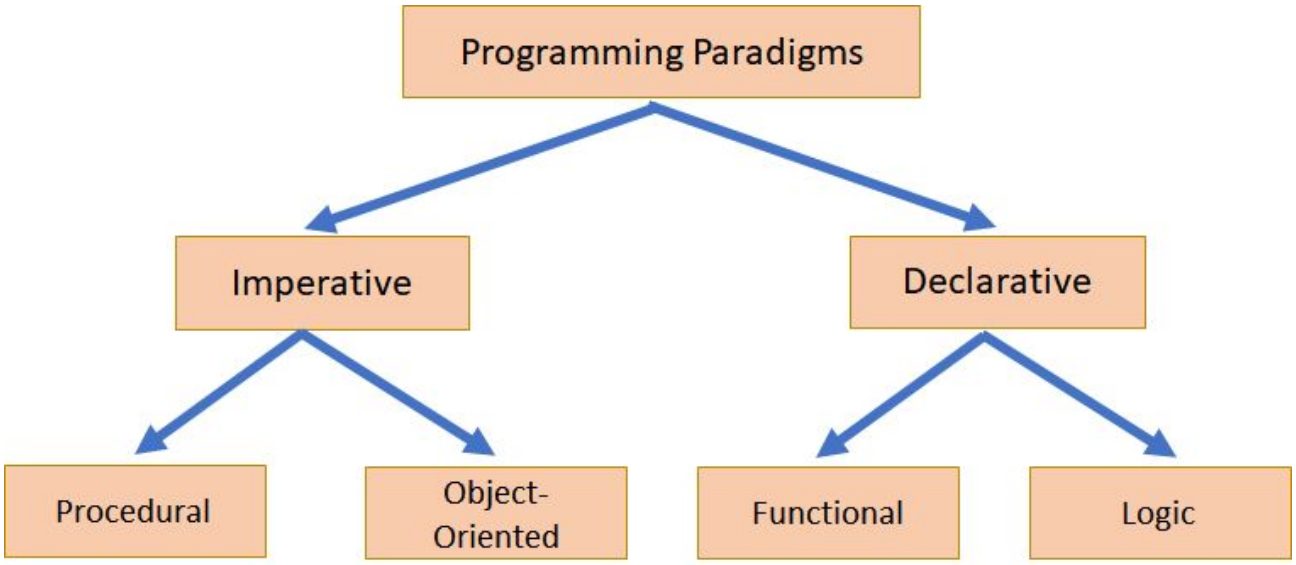
- **Modes of addressing memory**
  - Intermediate, Direct, Indirect, Indexed

1.2.4. e)

- **Object-oriented languages**
  - Classes
  - Objects
  - Methods
  - Attributes
  - Inheritance
  - Encapsulation
  - Polymorphism

Programming Paradigms

- Different approaches to using a programming language to solve a problem
- Split into two broad categories - imperative and declarative - which can be broken down into more specific paradigms



Imperative

- Use code that clearly specifies the actions to be performed

Procedural

- Widely-used paradigms as it can be applied to a wide range of problems
- Easy to write and interpret
- Written as a sequence of instructions
- Instructions are carried out in a step-by-step manner

Object-Oriented

- Suited to problems which can be broken into reusable components with similar characteristics
- Based on objects formed from classes which have attributes and methods
- Focuses on making programs that are reusable and easy to update and maintain

Declarative

- States the desired result and the programming language determines how best to obtain the result
- Details about how result is obtained are abstracted from the user

Functional

- Functions form the core of the program
- Function calls are often combined within each other
- Closely linked to mathematics

Logic

- A set of facts and rules based on the problem is defined
- Queries are used to find answers to problems

Procedural Language

- Simple to implement and applicable to most problems
- Not possible to solve all kinds of problems or may be inefficient to do so
- Provide traditional data types and data structures
- Structured programming is a popular subsection of procedural programming in which the control flow is given by four main programming structures:
    - Sequence
    - Selection
    - Iteration
    - Recursion

Assembly Language

- Low level language that is the next level up from machine code
- Uses mnemonics, which are abbreviations for machine code instructions
- Commands used are processor-specific
- Each line in assembly language is equivalent to one line of machine code

Below is a list of the mnemonics you need to be aware of and be able to use:

Mnemonic	Instruction	Function
ADD	Add	Add the value at the given memory address to the value in the Accumulator
SUB	Subtract	Subtract the value at the given memory address from the value in the Accumulator
STA	Store	Store the value in the Accumulator at the given memory address
LDA	Load	Load the value at the given memory address into the Accumulator

INP	Input	Allows the user to input a value which will be held in the Accumulator
OUT	Output	Prints the value currently held in the Accumulator
HLT	Halt	Stops the program at that line, preventing the rest of the code from executing.
DAT	Data	Creates a flag with a label at which data is stored.
BRZ	Branch if zero	Branches to a given address if the value in the Accumulator is zero. This is a conditional branch.
BRP	Branch if positive	Branches to a given address if the value in the Accumulator is positive. This is a conditional branch.
BRA	Branch always	Branches to a given address no matter the value in the Accumulator. This is an unconditional branch.

Modes of Addressing Memory

- Machine code instructions are made up of an opcode and operand
- Opcode specifies the instruction to be performed and the addressing mode
- Addressing mode specifies how the operand should be interpreted
- Operand holds a value related to the data on which the instruction is to be performed
- There are four addressing modes you need to know:
    - Immediate Addressing
        The operand is the actual value upon which the instruction is to be performed
    - Direct Addressing
        The operand gives the address which holds the value upon which the instruction is to be performed
    - Indirect Addressing
        The operand gives the address of a register which holds another address, where the data is located
    - Indexed Addressing
        An index register is used, which stores a certain value. The address of the operand is determined by adding the operand to the index register

Object Oriented Languages

Classes, Objects, Methods and Attributes

- A class is a template for an object and defines the state and behaviour of an object
- State is given by attributes which give an object's properties
- Behaviour is defined by the methods, which describe the actions it can perform
- Classes can be used to create objects by a process called instantiation
- An object is a particular instance of a class, and a class can be used to create multiple objects
- A setter is a method that sets the value of a particular attribute
- A getter is another special method used in OOP which retrieves the value of a given attribute
- Getters and setters ensure attributes cannot be directly accessed and edited but can only be altered by public methods. This is called encapsulation.
- Every class must also have a constructor method which allows a new object to be created

Inheritance

- Process in which subclass inherits all of the methods and attributes of the superclass
- Subclass can also have its own additional properties

Polymorphism

- Enables objects to behave differently depending on their class

Overloading

- Passing in different parameters into a method

Overriding

- Redefining a method so that it functions differently and produces a different output

Advantages of OOP

- High level of reusability
- Code made more reliable through encapsulation
- Makes code easy to maintain and update
- Classes can be reused as a black box which saves time and effort

Disadvantages of OOP

- Requires an alternative style of thinking
- Not suited to all types of problems
- Generally unsuitable for smaller problems