**LO1:  Understand programming languages.**

1.1 Explain the principles and concepts of programming languages.

  Principles and Concepts of Programming Languages

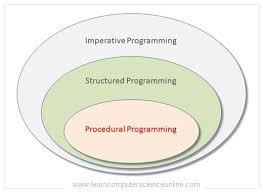
Split into 2 blog pages!!!

# Introduction

https://www.tiobe.com/tiobe-index/

Guidance:   
 one page for each maybe two for oop

Not including diagrams

  
   
As a minimum, your report should include:

## Procedural Programming

Procedure Programming is a programming paradigm that revolves around the concept that programs are a sequence of instructions to be executed. There is also a focus on splitting a program into procedures.

Some of the earliest procedure programming languages were Fortran (1956) and Algol (1958), followed by BASIC (1964), Pascal (1970) and C (1973).

Too many words! Add some images

Procedure languages use procedures which are logical blocks of code which can be invoked from other places in the code. These procedures can accept arguments and return values to the caller. Variables declared within each procedure are scoped only to the code within the procedure, but it can also access and modify variables which as within the global scope of the program.

Procedure languages follow structured programming practises where control flow is mandated by the use of reserved words such as **IF**, **WHILE** and **FOR** to define blocks of code. For comparison, non-structured languages (such as assembly) use **GOTO** or branch tables for this purpose.

Procedures callable units of code with a defined behaviour, a defined interface and can be called from other places in the code multiple times. This allows for code re-use, modularity and ease of comprehension. Some programming languages make a distinction between procedures that return a value (functions) and those that do not (subroutine, method or procedure). Others only use the term function whether they return a value or not.

Reusable code can further be grouped into Libraries, typically group together for a purpose, that can be used (or ‘imported’) by any compatible program. For example, a library could provide networking functionality or provide graphical functions. Typically, libraries are used without the need to access the source code behind the library provided by pre-compiled binary code.

Procedures allow for both passing parameters in and retrieving values from it. The exact methods use will depend on how the language is defined, but typically parameters are passed into a procedure by enclosing them as a comma separated list in brackets after the procedures name. If the procedure returns a single value, this is often directly assigned to a variable by use of an equals sign. For example:

**volume = CalculateVolumeOfCube(100, 200,300)**

Here a procedure called CalculateVolumeOfCube is called with three parameters (100, 200, 300) and the returned valued is stored in a variable called volume.

## Debugging Code

Debugging Code is the process of removing errors (called bugs) from your code. Errors can be divided into two main classifications – Syntactic Errors and Logical Errors.

**Syntactic Errors** are coding errors created by not correctly following the syntax of the programming language in use. They will prevent the code from running or compiling and can be caused by something as simple as mismatched quotations or brackets or even a missing semi-colon.

**Logical Errors** are coding errors where code is syntactically correct cause the program to do the wrong thing or generate Exceptions. An exception is how a program reports unexpected or illegal behaviour. For example, an infinite loop could generate a StackOverflowException, or the wrong variable could be passed into a procedure.

There are many tools that developers can use to help find and remove bugs. Often provided as part of an Integrated Development Environment (IDE) or simply as separate tools to be used as needed.

**Logging** allows programs to out messages to some external system (such as a database or a text file). These messages called logs can provide essential information to find out what went happened within a program, even while the program is deployed on the client’s system and far away from a programmer. When an error occurs, the logs can be sent to the programmer to help them identify and correct the error. They can also be used to audit interactions with a system in (for example) a security log.

**IntelliSense** can be provided by a code editing program to highlight the correct syntax for operations such as function calls. By providing the information directly as the code is created IntelliSense greatly decreases the chance of errors. IntelliSense can also incorporate syntax checking which validates the code as it is written. Earlier tools such as ‘lint’ provided syntax checking, leading to this functionality sometimes being called a ‘linter’ or ‘linting’. Some IDE’s provide IntelliSense that can be quite verbose and allow addin software to expand the information shown, while others will simply provide the correct syntax.

Include screenshot

A **Debugger** will also allow a developer to step through executing code line by line. A Debugger is an additional piece of software that examines a running program in real time. Use of breakpoints will cause the debugger to pause execution of the code and allow inspection of variables, a list of executed methods (an execution stack). The programmer may then be able to modify these variables to test different scenarios to reproduce or help correct an error.

Debuggers also allow for Watches where the value in a variable is highlighted and may be set to trigger a breakpoint when it changes or reaches a specific value.

Debuggers can also be used remotely to examine code running on other computers either locally or remote over the internet.

Add a screenshot of a debugger with watch window etc

## Object Orientated Programming

* Object Oriented languages: (Visual C#, C++, .NET, Java, Python etc.)
* Classes, Objects
* Abstraction
* Encapsulation
* Inheritance
* Polymorphism
* Design patterns
* IDEs

## Event Driven Programming

* Event Driven languages (Visual Basic, Visual C++ and Java etc.)
* Event types
* Triggers
* Controls and tools
* Use of loops
* Event listeners and handlers
* Links to Visual Languages – objects/properties/menus

Programming languages: e.g. Visual Basic, Visual C++ and Java

Task 1

Write a sentence or two about the key points you have learned about

* + Procedural programming
    - Sequence of command executed in order
    - Uses procedures for code re-use and ease of comprehension
    - Focuses on the over all task, normally a single task
    - Allows for global variables (which is kinda bad) and not for data protection/encapsulation
    - Program starts, does its thing and then ends
  + Event driven programming
    - Uses a loop to respond to events raised by systems or users i.e. flow of control determined by events not a linear progression
    - Can stall the loop with long running events
    - Order of event processing is not controlled by programmer
    - Triggers -> event listener (if active) -> event handler (if defined)
    - Uses procedures for code re-use and ease of comprehension
    - Programs do not ‘end’ but continue process the event loop until exited