DATA STRUCTURES

- Data Structure: any method used to store data in an organised and accessible format
- Abstract data type: a conceptual model of how data can be stored and the operations can be carried out on the data.
- Array: set of related data items stored under a single identifier

File: a collection of related data. The type can be specific to certain applications or portable which can be used in a wide range of programs. Two common portable formats are:

- Text file: a file that contains human-readable characters
    - Record: one line of a text file
- Binary file: stores data as sequences of 0s and 1s

Static and dynamic data structures

- Static data structure: method of storing data where the amount of data stored is fixed.

  A static data structure stores a set amount of data which is usually defined by the programmer allocated at the creation. Access is very quick as the memory location is fixed, however the data structure will take up memory even if it doesn't need it.

- Dynamic data structure: method of storing data where the amount of data stored will vary as the program is being run.

  A dynamic data structure can use more or less memory as needed through the use of a heap (collection of unused block of memory, usable by the program). The structure can take and put back memory to the heap as needed, so it can use resources efficiently.

Stack

Last in first out structure (LIFO), last item of data added is the first to leave.

Actually when data are popped out of the stack they are not actually removed, but there is a pointer that keeps track of where the top of the stack is

Call stack: special type of stack used to store information about active subroutines and functions within the program (stack frame: a collection of data about a subroutine call). Stacks are also used to keep track of nested loops and recursion.

Queue

First in first out (FIFO), data leaves in the order it arrives.

Can be implemented in different ways:

- *Linear queue:* data stored statically or dynamically in a line of data
- *Circular queue:* data is stored in a static array that acts as a fixed size ring where the back is connected to the front. Two pointers store the front and the rear of the queue.
- *Priority queue:* variation of the FIFO structure where some data may leave out of the sequence if it has higher priority than other data items

Buffer is a common use of queues, it stores items that are temporarily waiting to, for example, be executed by CPU or printed by a printer.

Graphs and Trees

- Graph: mathematical structure that models the relationship (edge or arc) between pairs of objects (vertex or node)
  - *Weighted graph:* a graph that has data value labelled on each edge
  - *Directed graph:* a graph where the relationship between vertices is one-way
  - *Adjacency list:* a data structure that stores a list of nodes with their adjacent nodes
  - *Adjacency matrix:* a data structure set up as a two-dimensional array or grid that shows whether there is an edge between each two pair of nodes
- Tree: connected undirected and acyclic graph
  - *Root:* starting node in a rooted tree from which all other nodes branch off
  - *Leaf:* node that has no child
  - *Binary tree:* a tree where each node can only have up to two child nodes attached to it

Hash Table

A data structure that stores key/value pairs based on an index calculated form an algorithm. The hashing algorithm creates a unique index from given items of key data, the index will point to the specific location in the array of data where the item will be stored.

Uses: databases (quick storage and retrieval), memory addressing (cache), encryption

Hashing algorithm:

- generates numeric value
- avoid/reduce collisions (same index for multiple keys)
- avoid clustering (indices are not randomly distributed)
- low load factor (ratio of how many indices are available to how many there are in total)

Collisions:

When a collision occurs there must be some way to of handling it. The two main methods are:

- *Chaining*: adds the key/value to a list stored at the same index
- *Rehashing*: rerun the same algorithm or a different again until a unique key is created. Normally uses probing where the algorithm searches for an empty slot (often the next available slot)

Vectors

Can be defined using magnitude and direction or components.

Convex combination: a method of multiplying vectors that produces a resulting vector within the convex hull (spatial representation of the vector space between two vectors).

$V_c = \alpha V_1 + \beta V_2$ where α and β are non-negative real numbers and $\alpha + \beta = 1$

VB implementation

Stack

```
Public Class Stack
    Private Data(10) As String
    Private Top As Integer

    Public Sub New() ' Constructor
        Top = 0
    End Sub

    Public Function IsEmpty() As Boolean
        Return IIf(Top = 0, True, False)
    End Function

    Public Function IsFull() As Boolean
        Return IIf(Top = 10, True, False)
    End Function

    Public Function Push(Value As String) As Boolean
        If IsFull() Then Return False
        Top += 1
        Data(Top) = Value
        Return True
    End Function

    Public Function Pop() As String
        If IsEmpty() Then Return Nothing
        Dim Temp As String = Data(Top)
        Top -= 1
        Return Temp
    End Function
End Class
```

Queue

```
Public Class Queue
    Private Data(10) As String
    Private Front, Back, Count As Integer

    Public Sub New()
        Back = 10
        Front = 0
        Count = 0
    End Sub

    Private Sub IncWithWrap(ByRef Ptr As Integer)
        Ptr += 1
        If Ptr > 10 Then Ptr = 0
    End Sub

    Public Function EnQueue(V As String) As Boolean
        If IsFull() Then Return False
        IncWithWrap(Back)
        Data(Back) = V
        Count += 1
```

```
        Return True
    End Function

    Public Function DeQueue() As String
        If IsEmpty() Then Return Nothing
        Dim Temp As String = Data(Front)
        IncWithWrap(Front)
        Count -= 1
        Return Temp
    End Function

    Public Function IsEmpty() As Boolean
        Return IIf(Count = 0, True, False)
    End Function

    Public Function IsFull() As Boolean
        If Count = 11 Then Return True
        Return False
    End Function

End Class
```