Section 13.1 - User-defined Data Types

Layer 6: High-Order Language

Syllabus Content Section 13: Data Representation

S13.1.1 Show understanding of why user-defined types are necessary

---

`User-defined data type`: where the programmer includes the definition in the program

- You are not restricted to built in types
- You can make sure that a user only types in a range of values that you allow for. So this avoids user errors.

S13.1.2 Define and use non-composite types

- Including enumerated, pointer

`Non-composite data type`: a data type defined without reference to another data type

`Enumerated data type`: a non-composite user-defined data type for which the definition identifies all possible values

`Pointer variable`: one for which the value is the address in memory of a different variable

- Enumerated:
  This is when you want to have a group of constants (never changes). Like having a list of items
  The list is ordinal, they have an order. The first thing is 0, then the next thing is 1, then 2....
  It’s a list of possible values

- Pointers:
  Refers or points to a memory location. Its used when you want your data type to change often

S13.1.3 Define and use composite data types

- Including set, record and class / object

`Set`: a collection of data items that lacks any structure; contains no duplicates and has a number of defined operations that can be performed on it

`record`: This allows the programmer to create record data types with components that precisely match the data requirements of the particular program.

`class/object`: A class is a data type which is used for an object in object-oriented programming. For a given object-oriented programming language there are likely to be a number of built-in classes

S13.1.4 Choose and design an appropriate user-defined data type for a given problem

---

> EXAMPLE - declaration of composite type

```
TYPE Student
      DECLARE LastName : STRING
      DECLARE FirstName : STRING
      DECLARE DateOfBirth : DATE
      DECLARE YearGroup : INTEGER
      DECLARE FormGroup : CHAR
ENDTYPE
```

> Example – using user-defined data types

```
DECLARE Pupil1 : Student
DECLARE Pupil2 : Student
DECLARE Form : ARRAY[1:30] OF Student
DECLARE ThisSeason : Season
DECLARE NextSeason : Season
DECLARE MyPointer : TIntPointer

Pupil1.LastName ← "Johnson"
Pupil1.Firstname ← "Leroy"
Pupil1.DateOfBirth ← 02/01/2005
Pupil1.YearGroup ← 6
Pupil1.FormGroup ← 'A'

Pupil2 ← Pupil1

FOR Index ← 1 TO 30
        Form[Index].YearGroup ← Form[Index].YearGroup + 1
NEXT Index

ThisSeason ← Spring
MyPointer ← ^ThisSeason
NextSeason ← MyPointer^ + 1
// access the value stored at the memory address
```