A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

13.1 User-defined data types

Name: _____ Class: _____ Date: _____ **Total: 16 marks**

Objective

Build the skills to answer exam questions on **user-defined data types** 用户定义类型 —enumerated, pointer, set, record and class.

You must be able to:

- explain **why** user-defined types are needed
- define and use **enumerated** and **pointer** (non-composite) types
- define and use **set**, **record** and **class** (composite) types

1 Worked examples

Study these first. Each one shows the method for a question type used later —follow the steps and you can do the Practice and Exam-style questions yourself.

Why and the enumerated type

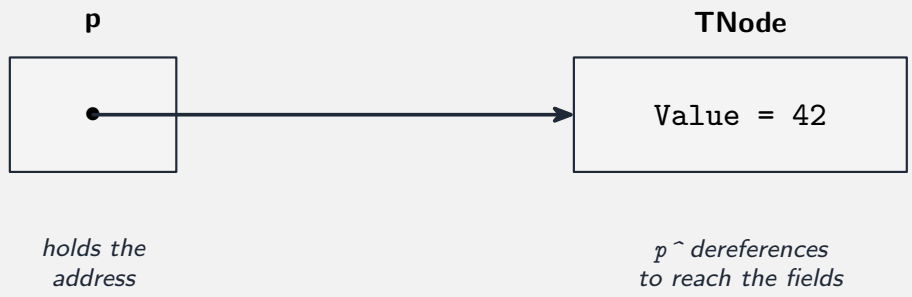
User-defined types make code **clearer** and let the compiler **reject illegal values**. An **enumerated** type is a fixed list of named constants:

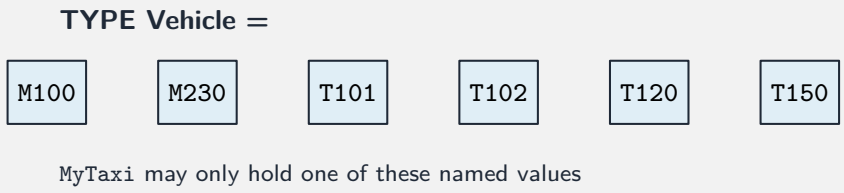
```
TYPE Vehicle = (M100, M230, T101, T102, T120, T150)
DECLARE MyTaxi : Vehicle
MyTaxi ← T102
```

You cannot assign anything outside the list (the names are stored as small integers).

Pointer type

A **pointer** holds the **memory address** of another variable (or `NULL`). `p^` **dereferences** it —reaches the variable it points to:





An enumerated type is another user-defined type

```
TYPE PNode = ^TNode
DECLARE p : PNode
p ← NEW TNode
p^.Value ← 42
```

■ Composite types: set and class

A **set** is an unordered collection of unique values (operations: add, remove, membership, union):

```
DECLARE Available : SET OF Colour
Available ← {Red, Blue}
IF Green IN Available THEN OUTPUT "ok"
```

A **class** combines **attributes** (data) with **methods** (operations); an **object** is an instance of a class.

2 Practice

Now apply the methods above.

2.1 State **one** reason user-defined types are useful. [1]

2.2 Declare an **enumerated** type `Day` holding the seven days of the week. [2]

2.3 State what a **pointer** holds. [1]

2.4 State what it means to **dereference** a pointer. [1]

---

3 Exam-style questions

3.1 Declare an enumerated type `Vehicle` with values `M100, M230, T101, T102`. Then declare a variable `MyTaxi` of that type and assign it `T101`. [3]

3.2 (i) Declare a **pointer** type `PNode` that points to a `TNode`. [1]

(ii) Write pseudocode to create a **new** `TNode` using a pointer `p` and set its `Value` to `5`. [2]

3.3 A program records which of `{Red, Green, Blue}` are available. State why a **SET** is suitable, and write a line that tests whether `Green` is available. [3]

3.4 State the difference between a **class** and an **object**. [2]

---

4 Go further

You are now ready for the real exam questions on this subtopic. Open the **13.1 User-defined data types** past-paper sheet in the Library, or try these in **Practice mode**:

- 9618/31 J25 —Q1 (user-defined types)
- 9618/31 N25 —Q1 (enumerated and pointer types)
- 9618/33 N25 —Q1 (composite types)
- 9618/33 N24 —Q6 (records and types)

Solutions

2.1 Any one: makes code **self-documenting/clearer**; lets the compiler **reject illegal values**; groups related values that belong together.

2.2 `TYPE Day = (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday)`.

2.3 The **memory address** of another variable (or `NULL` for no target).

2.4 To **follow the pointer** and reach (read or change) the **variable it points to**.

3.1

```
TYPE Vehicle = (M100, M230, T101, T102)
DECLARE MyTaxi : Vehicle
MyTaxi ← T101
```

3.2 (i) `TYPE PNode = ^TNode`.

3.2 (ii)

```
p ← NEW TNode
p^.Value ← 5
```

3.3 A **set** stores **unordered, unique** values and supports a **membership test** —exactly what "is this colour available?" needs. Test: `IF Green IN Available THEN` ⋯.

3.4 A **class** is the **definition/template** (attributes + methods); an **object** is an **instance** of that class, holding its own data.