A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

13.1 User-defined data types

Name: _____ Class: _____ Date: _____ **Total: 31 marks**

KEY WORDS class 类 • object 对象 • record 记录 • enumerated type 枚举类型 • user-defined data types 用户定义类型

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

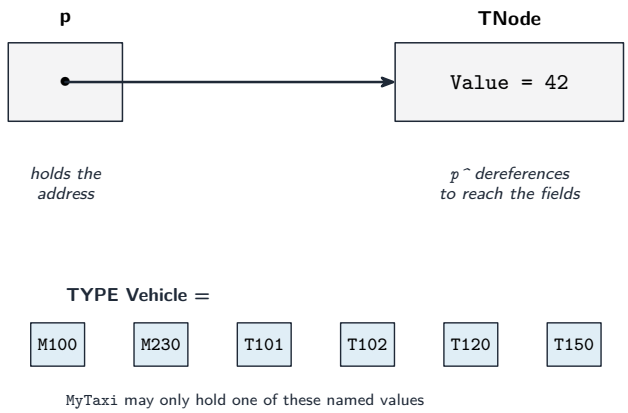
User-defined types make code **clearer** and let the compiler **reject illegal values**. An **enumerated** type is a fixed list of named constants:

```
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]

3.5 A booking system stores one record for each activity choice. A user-defined type `Choice` holds a member's name, the activity chosen and the date.

(a) **Write** the pseudocode type declaration for `Choice`. [4]

(b) **Write** the pseudocode statement that sets up a **single variable** `ThisChoice` of type `Choice`. [1]

(c) **Write** the pseudocode statements that assign the name `"Ahmed"`, the activity `"Tennis"` and the date `12/03/2027` to that variable. [3]

(d) A separate type `Activity` must hold the **names of up to 20 members** who have chosen one activity. **Write** its type declaration. [3]

(e) `Choice` must now also record whether the booking has been **paid for**. **Write** the new statement required in the declaration, and **state** the data type you chose and why. [2]

(f) **Explain** one advantage of a user-defined record type over holding the same values in three separate variables. [2]

Solutions

Each answer shows the steps, then the **result**.

2.1

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

2.2

- `TYPE … = ( … )` with the seven named values
- `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 … = (…)` with the values, then declare and assign

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

3.2

(i) `TYPE PNode = ^TNode`

(ii) `NEW` then dereference and assign

```
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

3.5

(a)

```
TYPE Choice
    DECLARE MemberName : STRING
    DECLARE ActivityName : STRING
    DECLARE ChoiceDate : DATE
ENDTYPE
```

- `TYPE`/`ENDTYPE`; all three fields; correct types; correct syntax

(b) `DECLARE ThisChoice : Choice`

(c) `ThisChoice.MemberName ← "Ahmed"`

- `ThisChoice.ActivityName ← "Tennis"`
- `ThisChoice.ChoiceDate ← 12/03/2027`

(d)

```
TYPE Activity
    DECLARE ActivityName : STRING
    DECLARE Members : ARRAY[1:20] OF STRING
ENDTYPE
```

- `TYPE`/`ENDTYPE`; the array of `STRING`; bounds `1:20`

(e) `DECLARE Paid : BOOLEAN` inside the type

- `BOOLEAN`, because the value can only be paid or not paid — two states, and no other value is valid

(f) the three values belong to **one** booking, so keeping them in one record means they can be passed to a procedure, stored in an array or written to a file as a single item

- they cannot drift out of step with each other the way three separate variables can