

# 10.1 Data Types and Records

Name: \_\_\_\_\_ Class: \_\_\_\_\_ Date: \_\_\_\_\_ **Total: 37 marks**

**KEY WORDS** data type 数据类型 • record 记录 • record structure 记录结构 • array 数组  
• index 索引

## Objective

Build the skills to answer exam questions on **data types** 数据类型 and **records** 记录 — choosing the right type, and defining and using a record structure.

You must be able to:

- choose the most appropriate **data type** for a value
- explain the purpose of a **record** structure
- write pseudocode to **define** a record and to **read/write** its fields

## 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.

### ■ Choosing a data type

Pick the **smallest precise type** that fits the value:

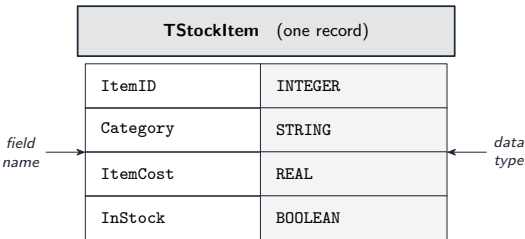
| To store                 | Best type | Why                   |
|--------------------------|-----------|-----------------------|
| a count, an index, an ID | INTEGER   | whole number          |
| a price, a measurement   | REAL      | has a decimal part    |
| a name, a sentence       | STRING    | text                  |
| a single grade letter    | CHAR      | exactly one character |
| ”is it paid?”            | BOOLEAN   | a TRUE/FALSE flag     |

**Exam method** — “give the data type of this expression”. Look at the **operation**, not the inputs:

- one character — LEFT(Name,1) or 'A' — gives a **CHAR**
- text in "...", or joined with & — gives a **STRING**
- + - \* on whole numbers gives an **INTEGER**; any / or a decimal value gives a **REAL**
- a comparison (>=, =) or AND/OR/NOT gives a **BOOLEAN**

### ■ Defining a record

A **record** groups several fields of **different types** under one type name:



```
TYPE TStockItem
  DECLARE ItemID : INTEGER
  DECLARE Category : STRING
  DECLARE ItemCost : REAL
  DECLARE InStock : BOOLEAN
ENDTYPE
```

### ■ Using a record

Declare a variable of that type (or a whole **array of records**), then use **dot notation** to reach each field:

```
DECLARE Item1 : TStockItem
DECLARE Items : ARRAY[1:100] OF TStockItem

Item1.Category ← "Fruit"
Item1.ItemCost ← 2.50
OUTPUT Item1.Category, " costs ", Item1.ItemCost
```

## 2 Practice

Now apply the methods above.

**2.1** Give the most appropriate data type for each value: (a) a student’s age; (b) a price in dollars; (c) whether a light is on. [3]

2.2 State the purpose of a **record** structure. [1]

2.3 A record type TBook must store a **Title** (text), **Pages** (whole number) and **InLibrary** (a flag). Write pseudocode to **define** it. [3]

2.4 Book1 is a variable of type TBook. Write one pseudocode line to set its **Title** to "Maths". [1]

3 Exam-style questions

3.1 The table shows four pseudocode assignment statements. Give an appropriate **data type** to declare each variable. [4]

| Statement            | Data type |
|----------------------|-----------|
| A ← LEFT(Name, 1)    |           |
| B ← Price * Quantity |           |
| C ← Count + 1        |           |
| D ← (Score >= 50)    |           |

3.2 A factory stores data about each component in a record type **Component**:

| Field    | Example | Meaning        |
|----------|---------|----------------|
| Item_Num | 123478  | a numeric ID   |
| Reject   | FALSE   | rejected?      |
| Stage    | 'B'     | a stage letter |
| Limit_1  | 13.5    | a value 0–100  |

(i) Write pseudocode to **declare the record structure** Component. [4]

(ii) A 1D array **Item** of **2000** elements will store all components. Write pseudocode to **declare the Item** array. [2]

3.3 State **two** benefits of using an **array of records** instead of several separate arrays.[2]

3.4 A bicycle-hire company stores one record for each hire. Each record holds a hire reference (up to 8 characters), the bicycle number, the date of hire, the number of hours hired, the cost per hour and whether the bicycle has been returned.

(a) **Complete** the table by giving an appropriate data type for each field. [3]

| field       | example value | data type |
|-------------|---------------|-----------|
| HireRef     | "HR004821"    |           |
| HoursHired  | 3             |           |
| CostPerHour | 4.50          |           |
| Returned    | TRUE          |           |

(b) **Write** pseudocode to declare the record structure `HireRecord`, including a field for the date. [5]

(c) A 1D array `Hire` of 500 elements will hold all the records. **Write** the pseudocode declaration for this array. [2]

(d) **Write** pseudocode to assign the value `4.50` to the cost per hour of the **first** record in the array. [2]

(e) **Outline** three benefits of storing the data as an array of records rather than as six separate parallel arrays. [3]

(f) **Evaluate** the expression `Hire[1].HoursHired * Hire[1].CostPerHour`, and **state** the data type of the result. [2]

## Solutions

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

### 2.1

- (a) INTEGER
- (b) REAL
- (c) BOOLEAN

### 2.2

- it groups **several fields of different data types** under **one identifier**, so values that belong together are stored as one unit

### 2.3

- TYPE ... ENDTYPE with three fields of the right types

```
TYPE TBook
  DECLARE Title      : STRING
  DECLARE Pages      : INTEGER
  DECLARE InLibrary  : BOOLEAN
ENDTYPE
```

### 2.4

- Book1.Title ← "Maths"

### 3.1

- A → **CHAR** (or STRING — one character from LEFT)
- B → **REAL** (a price/decimal)
- C → **INTEGER** (whole-number add)
- D → **BOOLEAN** (a comparison)

### 3.2

- (i) TYPE ... ENDTYPE with the four field types

```
TYPE Component
  DECLARE Item_Num : INTEGER
  DECLARE Reject   : BOOLEAN
  DECLARE Stage    : CHAR
  DECLARE Limit_1  : REAL
ENDTYPE
```

- (ii) DECLARE Item : ARRAY[1:2000] OF Component

### 3.3

- any two: one identifier for all the data; easy to process with a **loop** and an index; all fields for one item stay together; easy to pass to a procedure

### 3.4

- (a) HireRef — STRING

- HoursHired — INTEGER; CostPerHour — REAL
- Returned — BOOLEAN

- (b)

```
TYPE HireRecord
  DECLARE HireRef : STRING
  DECLARE BicycleNumber : INTEGER
  DECLARE HireDate : DATE
  DECLARE HoursHired : INTEGER
  DECLARE CostPerHour : REAL
  DECLARE Returned : BOOLEAN
ENDTYPE
```

- TYPE/ENDTYPE; all six fields; correct types; DATE for the date; correct pseudocode syntax

- (c) DECLARE Hire : ARRAY[1:500] OF HireRecord

- (d) Hire[1].CostPerHour ← 4.50

- (e) any three: the fields belonging to one hire are kept **together**, so a record can be passed to a procedure as a single item

- there is no risk of the arrays drifting **out of step** with one another
- adding a new field means changing one type definition instead of creating another array

- (f)  $3 \times 4.50 = \mathbf{13.50}$

- the result is **REAL**, because multiplying an **INTEGER** by a **REAL** gives a **REAL**