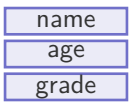
User-Defined Data Types

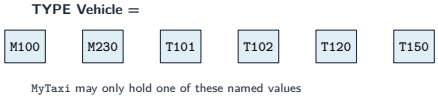
A-Level Computer Science ▪ Topic 13

A-Level Computer Science



What we will cover

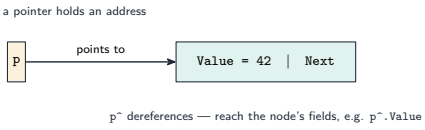
1. User-defined types
2. File organisation & access
3. Hashing
4. Floating-point numbers
5. Recap



Building your own types

- **enumerated** 枚举: a fixed named list, e.g. `(Mon, Tue, Wed)`
- **record** 记录: fields of different types bundled together
- **pointer** 指针: holds the address of another item
- **set** 集合: an unordered collection, no duplicates

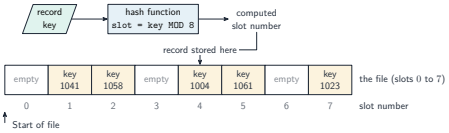
A composite type is built from others; a non-composite one (enumerated, pointer) is not.



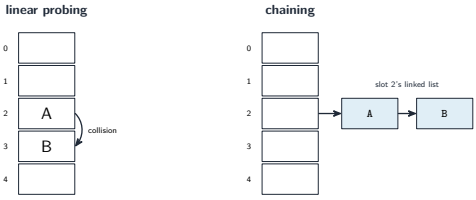
File organisation and access

- serial: records in the order added – read from the start
- sequential: sorted on a key – still read in order
- random: a hash gives direct access to any record

Serial and sequential need sequential access; a random-organised file allows direct access – fast for one record, no sorting kept.



Hashing



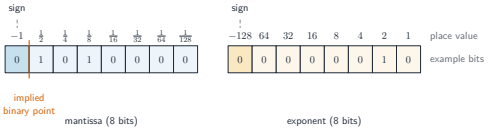
A **hash function** 哈希函数 turns a key into a storage address:

$$\text{address} = \text{hash}(\text{key})$$

- near-instant lookup, no search
- a **collision** 冲突 (two keys, one address) needs overflow or open addressing

> A good hash spreads keys evenly to keep collisions rare.

Floating-point numbers



A real is stored as a **mantissa** 尾数 and an **exponent** 阶码:

$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

- more mantissa bits ⇒ more precision
- more exponent bits ⇒ more range

Normalisation 规格化 maximises precision: the first mantissa bit after the point is significant (0.1… for positives).

Worked example – normalising

Normalise the positive fraction 0.0101 (mantissa) with its exponent.

Shift left until it reads $0.1\ldots$:

$0.0101 \rightarrow 0.101 \times 2^{-1}$

mantissa 0.101, exponent -1

Each left shift subtracts one from the exponent. A normalised positive always starts 0.1; a negative starts 1.0.

Topic 13 – key takeaways

1 Types

enumerated, record, pointer, set

2 Files

serial, sequential, random (direct)

3 Hashing

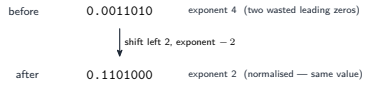
key $\rightarrow$ address; watch collisions

4 Float

mantissa $\times 2^{\text{exp}}$; normalise for precision

Check yourself

1. Which file type gives direct access?
2. What is a hash collision?
3. More mantissa bits: range or precision?
4. How does a positive normalised value start?



Answers **1.** random **2.** two keys, one address **3.** precision **4.** $0.1\ldots$