



Cambridge (CIE) A Level Computer Science



Your notes

User-defined Data Types

Contents

* User-Defined Types



What are user-defined data types?

- A user-defined data type is a **custom data structure** created by the programmer to match the specific needs of a program
 - It is built using **primitive data types** such as **INTEGER**, **STRING**, **BOOLEAN**, etc
 - It helps represent **real-world entities** more accurately than built-in types alone

Type	Description
Non-composite	Made up of a single data item (e.g. enumerated types like Day = (Mon, Tue, ...))
Composite	Made up of multiple data items , often of different types (e.g. records/structures)

Non-composite

What is an enumerated data type?

- An enumerated data type is a type of **non-composite, user-defined data type** that consists of a **list of named, fixed values**
- It allows the programmer to define a variable that can only take one of a **limited set of values**
- Making code **clearer, more readable**, and **easier to validate**
- An enumerated data type should be used:
 - When a variable should only hold **predefined values**
 - To improve **code readability** and **reduce errors**
 - To **restrict input** to a known, fixed list

Pseudocode
TYPE <identifier> = (value1, value2, value3, ...)

Example code

```
TYPE DayType = (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday)

DECLARE Today : DayType

Today ← Monday
```

```

IF Today = Saturday OR Today = Sunday THEN
    OUTPUT "It's the weekend!"
ELSE
    OUTPUT "It's a weekday."
ENDIF

```



Your notes

- **DayType** is the enumerated type
- **Today** can only be one of the seven values listed in **DayType**
- This makes it easy to check against valid options and avoids invalid assignments

What is a pointer data type?

- A pointer is a special type of variable that **stores the memory address** of another variable or data item, rather than the data itself
- Pointers are used to demonstrate **dynamic data structures**, such as **linked lists**

Pseudocode
TYPE <identifier> = ^<data type>

- The ^ indicates the data type is a pointer
- <data type> indicates the type of data stored in the memory location

Example code

```

TYPE Node
    DECLARE Data : INTEGER
    DECLARE Next : ^Node
ENDTYPE

```

```

DECLARE StartPtr : ^Node
DECLARE NewNode : ^Node

```

```

// Create a new node
NEW(NewNode)
NewNode^.Data ← 10
NewNode^.Next ← NULL

```

```

// Initialise the list
StartPtr ← NewNode

```

Line	What It Does
TYPE Node	Defines a new record type called Node
DECLARE Next : ^Node	Declares a pointer to another Node , used to link to the next item



Your notes

NEW(NewNode)	Allocates memory for a new node and assigns the pointer to NewNode
NewNode^.	Accesses the contents of the memory the pointer is pointing to
NULL	A special value used to indicate the end of a linked list (no further nodes)

Use of pointers in dynamic structures

- **Linked lists:** Each node uses a pointer to link to the next
- **Stacks and queues:** Implemented using pointers to track top/front positions
- **Memory efficiency:** Allows storage to be reused and allocated at runtime

Composite

What are records?

- A record is a **composite user-defined data type** that allows you to group together multiple **related fields**, each of which can have **different data types**
- Records are used to represent **real-world entities**, such as a student, product, or customer, in a single structure

Feature	Explanation
Composite type	Stores multiple values of different types in one structure
Custom fields	Each field has a name and a specific data type
Access via dot notation	Use the format RecordName.FieldName to access data
Reusable	Once defined, the record type can be used to declare multiple variables of that type

Pseudocode



Your notes

```
TYPE <RecordTypeName>

DECLARE <FieldName1> : <DataType1>

DECLARE <FieldName2> : <DataType2>

...

ENDTYPE
```

Example code

```
TYPE Student
  DECLARE Name : STRING
  DECLARE Age : INTEGER
  DECLARE Enrolled : BOOLEAN
ENDTYPE
```

```
DECLARE S1 : Student
```

```
S1.Name ← "Ali"
S1.Age ← 17
S1.Enrolled ← TRUE
```

```
OUTPUT S1.Name, " is ", S1.Age, " years old."
```

- Groups **related data** together for easy access
- Makes programs **easier to manage and read**
- Ideal for **arrays of records** (e.g. a list of students)

What are sets?

- A set is an **unordered collection** of **unique elements**
- Sets are useful for situations where **duplication is not allowed**, and where **membership testing** is common
- Features of a set include:
 - **No duplicate values**
 - **Order is not important**
 - **Can perform operations like union, intersection, and difference**

Pseudocode

```
TYPE <identifier> = SET OF <data type>
```

Example code

```
TYPE ColourSet = SET OF STRING
DEFINE Colours ("Red", "Green", "Blue") : ColourSet
```

```
IF "Green" ISIN Colours THEN  
  OUTPUT "Green is in the set"  
ENDIF
```



Your notes

- This creates a set named **Colours** containing **"Red"**, **"Green"**, and **"Blue"** using the user-defined set type **ColourSet**