# Data types, variables and constants

Ref As Pseudocode

## The six data types

Every value has a **data type** . The type says what kind of value it is and what you can do with it. 9618 pseudocode has six basic types.

| Type | What it holds | Example values |
|---|---|---|
| `INTEGER` | a whole number whole number | `5`, `-3`, `0` |
| `REAL` | a number that can have a fractional part fractional part | `4.7`, `0.3`, `-4.0` |
| `CHAR` | exactly one character | `'x'`, `'C'`, `'@'` |
| `STRING` | zero or more characters | `"Hello"`, `""` |
| `BOOLEAN` | `TRUE` or `FALSE` | `TRUE` |
| `DATE` | a calendar date | `02/01/2005` |

### How to write each value

A value written straight into the code is called a **literal** . Each type has its own way of writing one:

- **INTEGER**: normal digits, `25` or `-3`.
- **REAL**: always at least one digit **on each side** of the decimal point. Write `0.5`, not `.5`, and `4.0`, not `4.`.
- **CHAR**: one character in **single** quotes: `'A'`.
- **STRING**: characters in **double** quotes: `"A"`. The empty string empty string `""` has no characters at all.
- **BOOLEAN**: the words `TRUE` and `FALSE`, with no quotes.
- **DATE**: normally `dd/mm/yyyy`. Dates are written differently around the world, so it is good practice to say which format you use.

```
DECLARE Price : REAL
DECLARE Initial : CHAR
DECLARE Name : STRING
DECLARE Paid : BOOLEAN
Price ← 12.50
Initial ← 'M'
Name ← "Mei Lin"
Paid ← TRUE
OUTPUT Name, " (", Initial, ") paid ", Price, ": ", Paid
```

### Which type should I choose?

- Something you **count** (people, goals, items): `INTEGER`.
- Something you **measure** or that can have decimals (money, height, average): `REAL`.
- A yes/no answer, or a flag flag that remembers whether something happened: `BOOLEAN`.
- A phone number or an ID such as `"007"`: `STRING`. You never do arithmetic on it, and an `INTEGER` would lose the leading zeros.

**Common mistakes**

- `'Hello'` in single quotes. Single quotes hold **one** character only; a string needs double quotes.
- `"TRUE"` in quotes. That is a string of four letters, not a Boolean.
- Storing an average in an `INTEGER`. An average is usually not a whole number, so use `REAL`.

## DECLARE and assignment

### DECLARE

A **variable** is a named place in memory that holds one value, and the value can change while the program runs. Before you use a variable, you **declare** it: you give its name and its data type.

```
DECLARE <identifier> : <data type>
```

```
DECLARE Counter : INTEGER
DECLARE TotalToPay : REAL
DECLARE GameOver : BOOLEAN
Counter ← 0
TotalToPay ← 0.0
GameOver ← FALSE
OUTPUT Counter, " ", TotalToPay, " ", GameOver
```

Declare variables **at the top** of your answer, one per line. The guide says declaring is good practice, and many mark schemes give a mark for it. If a question says ”declare all variables”, you must do it.

### Assignment

**Assignment** puts a value into a variable. The operator is a **left arrow** ←.

```
<identifier> ← <value>
```

Read `Counter ← 0` as "Counter **becomes** 0" or "Counter is given 0". The right side is worked out first, then the result is stored on the left. That is why this line makes sense:

```
DECLARE Counter : INTEGER
Counter ← 5
Counter ← Counter + 1
OUTPUT Counter
```

The right side `Counter + 1` uses the **old** value, 5. The result, 6, then replaces it.

- The left side must be **one variable** (or one array element, or one field of a record).
- The right side can be any expression that gives a value of the same type.
- Typing on a keyboard? The site also accepts `<-` and turns it into `←`.

**Common mistakes**

- `Counter = 0` to assign. In pseudocode `=` **compares** two values; it never stores one.
- `0 ← Counter`, with the sides swapped. The variable that changes is always on the left.
- Using a variable before it has a value, for example adding to `Total` before `Total ← 0`. Always give a starting value. This is called **initialising** the variable.

## CONSTANT

A **constant** is a named value that **never changes** while the program runs.

```
CONSTANT <identifier> = <value>
```

```
CONSTANT HourlyRate = 6.50
CONSTANT DefaultText = "N/A"
DECLARE Hours : INTEGER
Hours ← 8
OUTPUT "Pay: ", Hours * HourlyRate
OUTPUT "Comment: ", DefaultText
```

Note the **equals sign** `=` here, not the arrow. A constant is defined once; it is not assigned.

### ■ Why use a constant?

- `HourlyRate` tells the reader what `6.50` means. A bare number in the middle of code does not.
- If the rate changes, you change **one line**, not every place the number appears.



exercises.lol exercises.com.cn

## The rules

- Put constants at the **start** of the program.
- The value must be a **literal**, such as `6.50` or `"N/A"`. It cannot be a variable, another constant or a calculation. `CONSTANT Total = Price * 2` is not allowed.
- A program must never assign to a constant. `HourlyRate ← 7.00` is an error.

## Practice

**1.** Write declarations for: a person's age, their height in metres, whether they are a member, and their first initial.

**2.** A shop has a fixed tax rate of 0.08. Write the constant, then output the tax on a price of 50.

<details> <summary>Model answer</summary>

```
DECLARE Age : INTEGER
DECLARE Height : REAL
DECLARE IsMember : BOOLEAN
DECLARE FirstInitial : CHAR
Age ← 17
Height ← 1.72
IsMember ← TRUE
FirstInitial ← 'J'
OUTPUT Age, " ", Height, " ", IsMember, " ", FirstInitial
```

```
CONSTANT TaxRate = 0.08
DECLARE Price : REAL
Price ← 50.0
OUTPUT "Tax: ", Price * TaxRate
```

The output of the second program is `Tax: 4.0`.

</details>

### Common mistakes

- `CONSTANT TaxRate ← 0.08`. That is the **IGCSE** form. In 9618 a constant uses `=`.
- Writing `DECLARE` in front of a constant. A constant uses the keyword `CONSTANT` only.