# Strings and built-in functions

Ref As Pseudocode

## Joining strings with &

To **concatenate** means to join strings end to end. In 9618 the operator is `&`.

```
DECLARE FirstName : STRING
DECLARE LastName : STRING
DECLARE FullName : STRING
FirstName ← "Summer"
LastName ← "Pudding"
FullName ← FirstName & " " & LastName
OUTPUT FullName
OUTPUT LENGTH(FullName)
```

- `&` makes **one new string**. You can store it, pass it to a function, or output it.
- Nothing is added between the parts. To get a space, join `" "` yourself.
- `&` joins strings and characters. To join a number, first turn it into a string with `NUM_TO_STR` (subtopic 7.4).

### Building a string in a loop

Start with the empty string and add to it on each pass:

```
DECLARE Stars : STRING
DECLARE i : INTEGER
Stars ← ""
FOR i ← 1 TO 5
   Stars ← Stars & "*"
NEXT i
OUTPUT Stars
```

### & or a comma?

- In `OUTPUT`, a comma prints values one after another: `OUTPUT "Age: ", Age`. The values can be any type.
- `&` makes a new **string value**, for example to store in a variable or write to a file.

### Common mistakes

- `"Total: " & 25`. 25 is a number; write `"Total: " & NUM_TO_STR(25)`.
- `+` to join strings, as in Python. Pseudocode uses `&`.



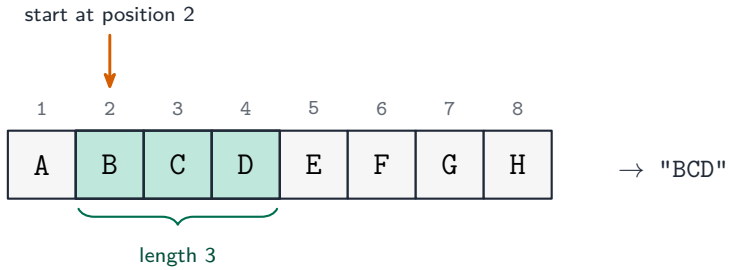
exercises.lol exercises.com.cn

## LEFT, RIGHT, MID and LENGTH

In the exam, the string functions you may use are **always printed in an insert** with the paper. You still need to know how they behave, because you must use them quickly and correctly. Positions in a string start at **1**.

| Function | Returns | Example | Result |
|---|---|---|---|
| `LENGTH(S)` | the number of characters in S | `LENGTH("Happy Days")` | `10` |
| `LEFT(S, x)` | the first x characters | `LEFT("ABCDEFGH", 3)` | `"ABC"` |
| `RIGHT(S, x)` | the last x characters | `RIGHT("ABCDEFGH", 3)` | `"FGH"` |
| `MID(S, x, y)` | y characters, starting at position x | `MID("ABCDEFGH", 2, 3)` | `"BCD"` |

- `LENGTH` counts spaces and punctuation too.
- In `MID`, the third value is a **length**, not an end position. `MID("ABCDEFGH", 2, 3)` starts at B and takes 3 characters.
- A **substring** is part of a string. `LEFT`, `RIGHT` and `MID` all return one.




*MID("ABCDEFGH", 2, 3): start at position 2 and take 3 characters*

### Visit each character

`MID(S, i, 1)` is the character at position `i`. A loop from 1 to `LENGTH(S)` visits them all. This pattern appears in almost every string question.

```
DECLARE Word : STRING
DECLARE i : INTEGER
DECLARE Vowels : INTEGER
DECLARE Letter : CHAR
Word ← "EDUCATION"
Vowels ← 0
FOR i ← 1 TO LENGTH(Word)
   Letter ← MID(Word, i, 1)
   IF Letter = 'A' OR Letter = 'E' OR Letter = 'I' OR Letter = 'O' OR
   ↪  Letter = 'U' THEN
      Vowels ← Vowels + 1
   ENDIF
NEXT i
OUTPUT Word, " has ", Vowels, " vowels"
```

### Reverse a string

```
DECLARE Word : STRING
DECLARE Reversed : STRING
DECLARE i : INTEGER
Word ← "STRESSED"
Reversed ← ""
FOR i ← LENGTH(Word) TO 1 STEP -1
   Reversed ← Reversed & MID(Word, i, 1)
NEXT i
OUTPUT Reversed
```

### Common mistakes

- Starting at position 0. The first character is at position **1**.
- `MID(S, 2, 5)` meaning "from 2 to 5". It means "5 characters from position 2".
- `SUBSTRING(S, 2, 3)`. That is the IGCSE name. In 9618 the function is `MID`.
- Asking for more characters than there are: `LEFT("Hi", 5)` is an error.

## Case and character codes



exercises.lol exercises.com.cn

### Upper and lower case

| Function | Returns | Example | Result |
|---|---|---|---|
| `TO_UPPER(x)` | x with every letter in upper case; x is a `STRING` or a `CHAR` | `TO_UPPER("Hello")` | `"HELLO"` |
| `TO_LOWER(x)` | x with every letter in lower case | `TO_LOWER("Hello")` | `"hello"` |
| `UCASE(c)` | one `CHAR` in upper case | `UCASE('h')` | `'H'` |
| `LCASE(c)` | one `CHAR` in lower case | `LCASE('W')` | `'w'` |

Characters that are not letters, such as digits and spaces, do not change. `TO_UPPER` and `TO_LOWER` are the ones printed in the Paper 2 insert, and they work on whole strings.

A common use: compare two strings **without caring about case**.

```
DECLARE Answer : STRING
Answer ← "yEs"
IF TO_UPPER(Answer) = "YES" THEN
   OUTPUT "You said yes"
ENDIF
```

### Character codes: ASC and CHR

Each character is stored as a number, its **character code** . In ASCII, `'A'` is 65, `'B'` is 66, … `'Z'` is 90, and `'a'` is 97.

| Function | Returns | Example | Result |
|---|---|---|---|
| `ASC(c)` | the code of the character c | `ASC('A')` | 65 |
| `CHR(x)` | the character with code x | `CHR(66)` | `'B'` |

```
DECLARE Letter : CHAR
DECLARE Code : INTEGER
Letter ← 'C'
Code ← ASC(Letter)
OUTPUT Letter, " has code ", Code
OUTPUT "The next letter is ", CHR(Code + 1)
```

Two things the codes make easy:

- **Is it a capital letter?** `Letter >= 'A' AND Letter <= 'Z'` works because the codes are in order.
- **Shift a letter** (as in a Caesar cipher): `CHR(ASC(Letter) + 3)`.

### Common mistakes

- `UCASE("hello")` on a whole string in a 9618 answer. `UCASE` takes one character; use `TO_UPPER` for a string.
- Forgetting that `'a'` and `'A'` have different codes, so `'a' = 'A'` is `FALSE`.

## Numbers and strings

### Converting between them

| Function | Returns | Example | Result |
|---|---|---|---|
| `NUM_TO_STR(x)` | the number x as a string | `NUM_TO_STR(87.5)` | `"87.5"` |
| `STR_TO_NUM(x)` | the string x as a number | `STR_TO_NUM("23")` | 23 |
| `IS_NUM(S)` | `TRUE` if S is a valid number | `IS_NUM("12a")` | `FALSE` |

Data read from a file or typed by a user often arrives as a string. Check it with `IS_NUM` before you convert it, or the program can fail.

```
DECLARE Entry : STRING
DECLARE Value : INTEGER
Entry ← "42"
IF IS_NUM(Entry) THEN
   Value ← STR_TO_NUM(Entry)
   OUTPUT "Double: ", Value * 2
ELSE
   OUTPUT "Not a number"
ENDIF
OUTPUT "Code " & NUM_TO_STR(7) & "B"
```

### INT and RAND

| Function | Returns | Example | Result |
|---|---|---|---|
| `INT(x)` | the whole-number part of x; it **cuts off** the decimals | `INT(27.9)` | 27 |
| `RAND(x)` | a random `REAL` from 0 up to x, **not including** x | `RAND(87)` | e.g. `35.43` |

`INT` does **not** round: `INT(27.9)` is 27, not 28. To round to the nearest whole number, add 0.5 first: `INT(X + 0.5)`.

To get a random whole number from 1 to 6, like a dice:

```
DECLARE Dice : INTEGER
Dice ← INT(RAND(6)) + 1
OUTPUT "You rolled ", Dice
```



exercises.lol exercises.com.cn

`RAND(6)` gives a value from 0 up to (not including) 6. `INT` makes that 0 to 5, and `+ 1` gives 1 to 6.

### Dates

`DATE` values have their own functions in the insert, such as `DAY(ThisDate)`, `MONTH(ThisDate)`, `YEAR(ThisDate)` and `SETDATE(Day, Month, Year)`. Read the insert in your paper for the exact list: it describes each one.

### Practice

**1.** A product code such as `"AB1234"` has two letters and then four digits. Write pseudocode that outputs `"Valid"` if the last four characters are a number, and `"Invalid"` otherwise.

**2.** Output the initials of `"Grace Brewster Hopper"`: the first letter of each word.

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

```
DECLARE Code : STRING
Code ← "AB1234"
IF LENGTH(Code) = 6 AND IS_NUM(RIGHT(Code, 4)) THEN
   OUTPUT "Valid"
ELSE
   OUTPUT "Invalid"
ENDIF
```

```
DECLARE FullName : STRING
DECLARE Initials : STRING
DECLARE i : INTEGER
FullName ← "Grace Brewster Hopper"
Initials ← LEFT(FullName, 1)
FOR i ← 2 TO LENGTH(FullName)
   IF MID(FullName, i - 1, 1) = " " THEN
      Initials ← Initials & MID(FullName, i, 1)
   ENDIF
NEXT i
OUTPUT Initials
```

It prints `GBH`. A letter starts a new word when the character **before** it is a space.

</details>

**Common mistakes**

- Doing arithmetic on a string: `"5" + 1`. Convert first with `STR_TO_NUM`.
- Thinking `INT` rounds. It always cuts off the fractional part.
- `RAND(6)` used as a dice. It gives decimals like 3.81 and can give 0; use `INT(RAND(6)) + 1`.