

AQA Computer Science A-Level
4.12.1 Functional programming
paradigm
Concise Notes



Specification:

4.12.1.1 Function type

Know that a function, f , has a function type $f: A \rightarrow B$ (where the type is $A \rightarrow B$, A is the argument type, and B is the result type).

Know that A is called the domain and B is called the co-domain.

Know that the domain and co-domain are always subsets of objects in some data type.

Loosely speaking, a function is a rule that, for each element in some set A of inputs, assigns an output chosen from set B , but without necessarily using every member of B . For example, $f: \{a,b,c,\dots,z\} \rightarrow \{0,1,2,\dots,25\}$ could use the rule that maps a to 0, b to 1, and so on, using all values which are members of set B .

The domain is a set from which the function's input values are chosen.

The co-domain is a set from which the function's output values are chosen. Not all of the codomain's members need to be outputs.

4.12.1.2 First-class object

Know that a function is a first-class object in functional programming languages and in imperative programming languages that support such objects.

This means that it can be an argument to another function as well as the result of a function call.

First-class objects (or values) are objects which may:

- appear in expressions
- be assigned to a variable
- be assigned as arguments
- be returned in function calls.

For example, integers, floating-point values, characters and strings are first class objects in many programming languages.



4.12.1.3 Function application

Know that function application means a function applied to its arguments.

The process of giving particular inputs to a function is called function application, for example $\text{add}(3,4)$ represents the application of the function add to integer arguments 3 and 4. The type of the function is $f: \text{integer} \times \text{integer} \rightarrow \text{integer}$ where $\text{integer} \times \text{integer}$ is the Cartesian product of the set integer with itself. Although we would say that function f takes two arguments, in fact it takes only one argument, which is a pair, for example $(3,4)$.

4.12.1.4 Partial function application

Know what is meant by partial function application for one, two and three argument functions and be able to use the notations shown opposite. The function add takes two integers as arguments and gives an integer as a result. Viewed as follows in the partial function application scheme: $\text{add}: \text{integer} \rightarrow (\text{integer} \rightarrow \text{integer})$ $\text{add } 4$ returns a function which when applied to another integer adds 4 to that integer. The brackets may be dropped so function add becomes $\text{add}: \text{integer} \rightarrow \text{integer} \rightarrow \text{integer}$ The function add is now viewed as taking one argument after another and returning a result of data type integer .

4.12.1.5 Composition of functions

Know what is meant by composition of functions. The operation functional composition combines two functions to get a new function. Given two functions $f: A \rightarrow B$ $g: B \rightarrow C$ function $g \circ f$, called the composition of g and f , is a function whose domain is A and co-domain is C . If the domain and co-domains of f and g are \mathbb{R} , and $f(x) = (x + 2)$ and $g(y) = y^3$. Then $g \circ f = (x + 2)^3$ f is applied first and then g is applied to the result returned by f .



Functions

- **Rules** that, for each **element** in some **set A** of **inputs**, **assigns** an **output** chosen from **set B**
- Does not necessarily use every member of B
- An **argument** is a piece of data passed to a **function**
- Functions are said to be applied to their **arguments**, creating the **return value**
- An **argument** could be a **number** (0, 1, 3.4, -8 e.t.c), a **character** (“a”, “D”, “!” e.t.c) or any **other data type**
- Functions will **specify** what **data type** is required for their **arguments**

Function Types

- All functions have a **function type**
- If **f** is a function, **A** is the input and **B** is the output, the function type can be defined as the following:

$$f: A \rightarrow B$$

- A is known as the **argument type**, and B is the **result type**
- This means function **f** **maps A to B**
- The **set of inputs** (A) is described as the **domain**
- The **set of outputs** (B) is called the **co-domain**
- The domain and co-domain are always **subsets** of **objects** in **some data type**

First-class objects

- Objects (or values) are objects which may:
 - appear in expressions
 - be assigned to a variable
 - be assigned as arguments
 - be returned in function calls
- Examples of first-class objects in procedural programs include **integers**, **floating-point values**, **characters** and **strings**
- In functional programming, **functions** are **first-class objects**



Function application

- A function's arguments are said to be **applied** to the function
- The process of giving particular input to a function is called **function application**

Partial function application

- **Partial function application** takes advantage of the **inability of a function** to take **more than one input**
- In partial function application, one of the **arguments** is fixed, leading to a more **restricted, specialised** function
- For example, the function *add 2 3* would be broken down into the new function *add 2* and the parameter 3

Composition of functions

- **Function composition** is the process of **combining two functions** to create a **new function**
- The user is able to use the functions both **separately**, and in **conjunction**
- **Any two** functions can be combined as long as the **domain** of one of the functions is the **same** as the **co-domain** of the other
- The symbol \circ indicates that two functions are being combined

