Modules and crates

Crates and modules are used to organize code into separate namespaces, allowing for easier management of code and better code reusability. A module is a named container foritems such as structs, enums, functions, constants, and traits. A crate is a single compilation unit. It has a root directory, and a root module defined at the filelib.cairo under this directory.

Defining a module

A module can be defined in two ways: inlined in the current file or in a separate file. When a module is defined in a separate file, the file is located according to the module hierarchy. For example, the modulecrate_name::a::b::c is defined in the file/a/b/c.cairo . a.cairo:

// This is a module defined in the current file. mod foo { // ... items ... }

// This is a module defined ina/bar.cairo. mod bar;

Module hierarchy

The module hierarchy is defined by the module definitions in the source code. When a module is defined inside another module, it is called a submodule, and it can be referred to using its name. The module containing the current module is called the parent module, and can be referred to using thesuper keyword. The root module of a crate has no parent.

Module scope

Items defined inside a module are directly available within the module. In particular, items defined at outer modules are not directly available in inner modules, even when inlined. To refer to an item defined in another module, the item must be either fully qualified or imported with theuse keyword. For more information, seePaths . struct A {} struct B {}

mod foo { use super::B; // Import the struct B from the parent module. fn bar() { A {}; // This is not allowed, because A is not in scope. super::A {}; // This is allowed, because super refers to the parent module. B {}; // This is allowed, because B is imported. } }

2.5 Punctuation 4. Items