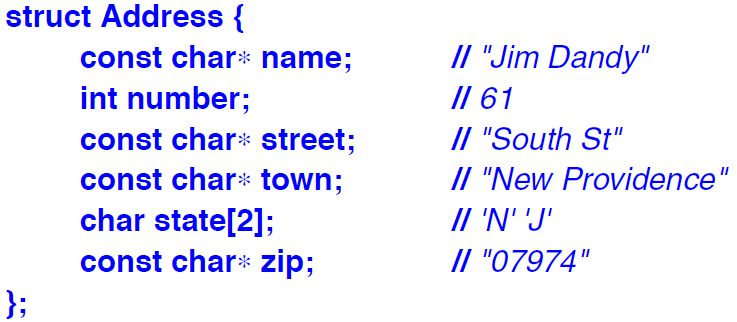
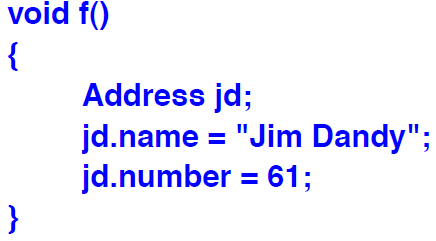
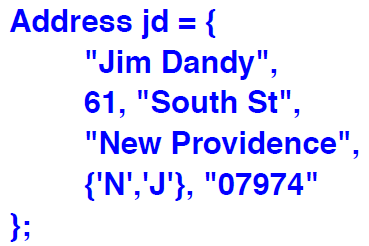
* Most primitive variants of the notion of a user-defined class are –
* **struct:** A sequence of elements/members of arbitrary type. It is a simple form of a class.
* **union:** A struct that holds the value of just one of its members at any one time.
* **enum:** A type with a set of named constants called enumerators.
* **enum class:** An enum where the enumerators are within the scope of the enumeration and no implicit conversions to other types are provided.
* **Structures –**



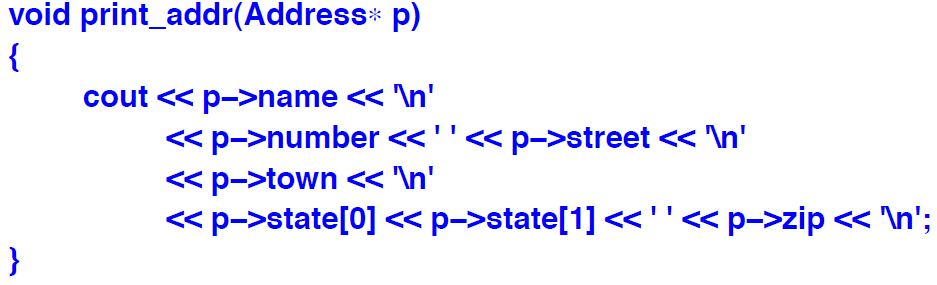
* The members of a struct can be accessed/initialised in various ways –
* By the use of a . (dot) operator.



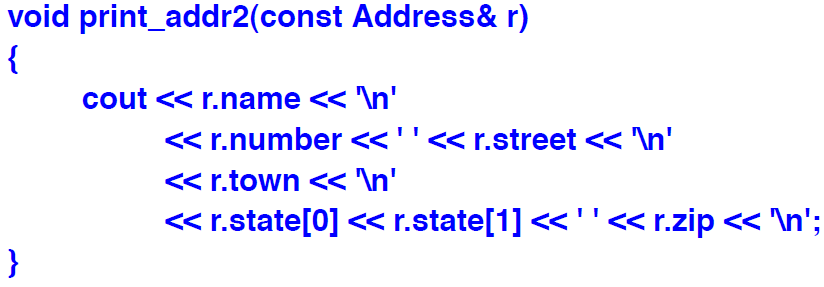
* By the use of {} to initialise only.



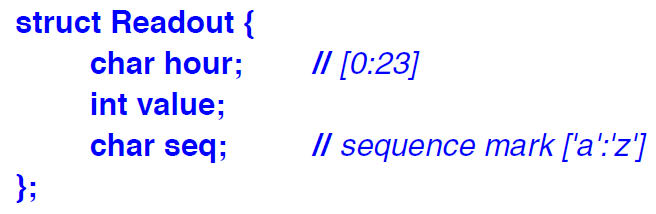
* By the use of -> (pointer).



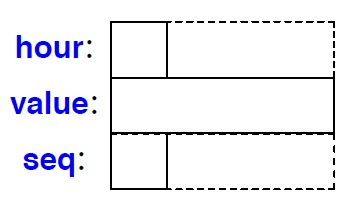
* By the use of a reference and . (dot) operator.



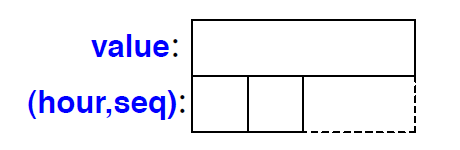
* Structures can be passed as function arguments and returned as a result of a function.
* Other plausible operators, such as comparison, are not available by default, but can be defined by the user as and when required.
* **struct layout:** An object of a struct holds its members in the order they are declared.



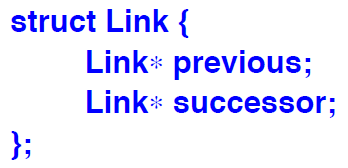
* Members are allocated in memory in declaration order.
* The size of an object of a struct is not necessarily the sum of sizes of its members.
* This is because many machine require object of certain types to be allocated on architecture dependent boundaries.
* This leads to “holes” in the structures.



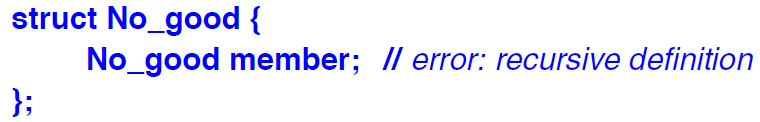
* The wasted space can be minimised by simply ordering members by decreasing order of their size.



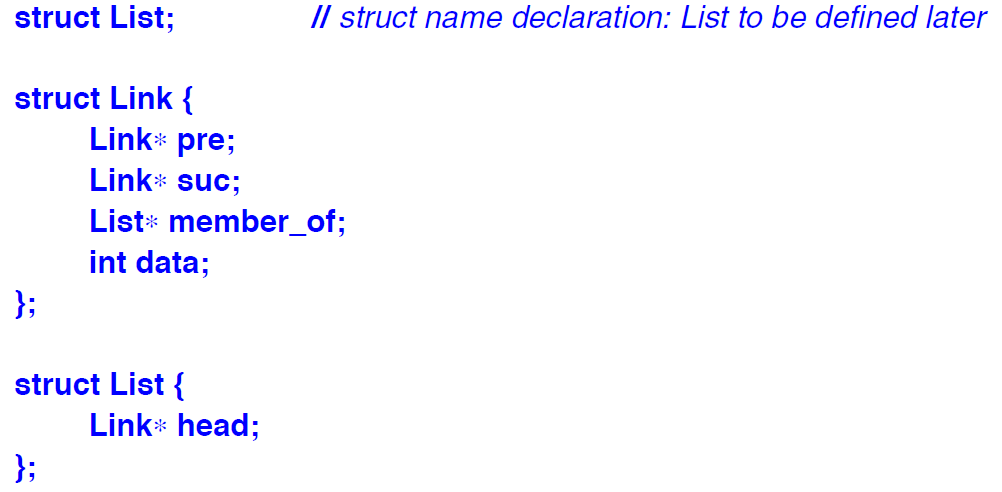
* It is best to order members by readability.
* Members should only be sorted by size if there is a demonstrated need to optimise memory.
* The **name of a struct** becomes available immediately after it has been encountered.



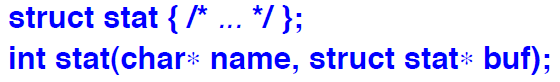
* However, it is not possible to declare objects of a struct until it is completely declared.



* This gives an error because the compiler is not able to determine the size of the object declared.
* To allow two or more structs to refer to each other, one struct can be just declared and defined later.



* A struct and a non-struct can be declared with the same name in the same scope. The plain name will be referring to the non-struct and the one with struct prefix will be referring to the struct.



* However, it is best to avoid similar names and hence, confusion.
* **Structures and classes:** A struct is simply a class where the members are public by default. So, a struct can have member functions and constructors.
* **Structures and arrays:** We can have arrays of structs and structs containing arrays.