

Abstract Data Types (ADT)

Abstraction : Represent a process or entity by its essential attributes and hide away other details

- Process : Way of allowing a program to specify that some process is to be done, without providing the details of how it is to be done.
- Data (Abstract Data Type): Informally, it is an encapsulation that includes only the data representation of a particular data type and the subprograms that provide the operations for objects of that type.

Formal Definition : An ADT satisfies the following two conditions :

1. The representation of the type and the operations on objects of the type are contained in a single syntactic unit. Also, other program units can create objects of the defined type.
2. The representation of objects of the type are hidden from the program units that use the type, so the only direct operations possible on those objects are those provided in the types' definition.

Examples of ADT's

Built-in : All built-in types are abstract data types;
Provides portability.

- *Float* : Means of creating a floating-point data type; comes with a set of arithmetic operations to manipulate an object of that type.
- Actual format of a data value is hidden; user cannot manipulate the actual representation.
- Only the system provided operations are valid.

User-defined : Should provide a type of definition and a set of operations for manipulating objects of that type

- Hiding representation allows for change without the user knowing about it.
- Increases reliability.
- Abstract data type for a *stack*. Don't care if actual representation is a linked-list or an array.

Programs can be written that depend only on the abstract properties and not on the representation.

How Languages provide Data Abstraction

Simula 67 : First language to provide facilities for data abstraction.

- **class** class_name;
 begin
 - class variable declarations –
 - class subprogram definitions –
 - class code section –**end** class_name;
- Code section executed during creation (constructor).
- Manipulation done by accessing variables (not hidden) and subprograms.

Modula-2 : Primary mechanism for data abstraction is *module*.

- Contains type defs., objects and subprograms.
- Definition module (partial specs);
 implementation module (complete definition).
- Program units that have access to a module are called client units.
- A type in the definition module can be *transparent* or *opaque*.
- Only pointer types can be opaque.

How languages provide Data Abstraction

Ada : Primary mechanism for data abstraction is *package*.

- Similar to Modula-2, except can export more than pointers.
- Specification package (invisible/visible); package body (not all packages have body).
- *private* and *limited private* types
- Can construct *generic* or parameterized abstract data types.
- Recompile of clients when exported type changes.