UNIT-II

DATA TYPES

① **Introduction** -

A data type defines a collection of data values and a set of predefined operations on those values. Computer programs produce results by manipulating data.

A descriptor is the collection of the attributes of a variable

② **Primitive Data Types** -

Data types that are not defined in terms of other types are called primitive data types. There are mainly three types of primitive data types -

Numeric Types - Integer, Floating-Point, Complex, Decimal

Boolean Types - (True, False), simplest of all types

Character Types - ASCII character set, string of length 1

③ **Character String Types** -

It is one in which the values consist of sequence of characters

→ **Design Issues** -

→ should strings be simply a special kind of character array or a primitive type?

→ should strings have static or dynamic length?

→ **String and their operations** -

Assignment, Catenation, substring reference, comparison & pattern matching

→ **String length options** -

① static length string - fixed number of characters

② limited dynamic length string - zero to maximum number of characters

③ Dynamic length string - Varying number of characters

→ **Implementation** -

Static string
length
Address

Compile-time descriptor

limited dynamic string
Maximum length
Current length
Address

Run-time descriptor

Dynamic string
current length
Address

Run time descriptor

Dynamic allocation - linked list, arrays of pointers, storage cells.

④ User-defined Types -(Ordinal)

An ordinal type is one in which the range of possible values can be easily associated with the set of positive integers. There are two - Enumeration Type and Subrange Type.

→ Enumeration Types -

It is one in which all of the possible values, which are named constants, are provided, or enumerated, in the definition.

Design Issues -

→ Is an enumeration constant allowed to appear in more than one type definition, and if so, how is the type of an occurrence of that constant in the program checked?

→ Are enumeration values coerced to integer?

→ Are any other types coerced to an enumeration type?

→ Subrange Types -

It is a contiguous subsequence of an ordinal type.

→ No design issues that are specific to subrange types.

→ Implementation -

Enumeration types are usually implemented as integers.

Subrange types are implemented in exactly the same way as their parent types, except that range checks must be implicitly included by the compiler in every assignment of a variable or expression to a subrange variable.

⑤ Array Types -

An array is a homogeneous aggregate of data elements in which an individual element is identified by its position in the aggregate, relative to the first element.

→ Design Issues - subscript ⇉ Index

→ What types are legal for subscripts?

→ Are subscripting expressions in element references range checked?

→ When are subscript ranges bound?

→ When does array allocation take place?
→ Are ragged or rectangular multidimensional arrays allowed, or both?
→ Can arrays be initialized when they have their storage allocated?
→ What kinds of slices are allowed, if any?

→ Array Categories -

Based on the binding to subscript ranges, the binding to storage and from where the storage is allocated. Five categories are -

① Static array - subscript ranges are statically bound and storage allocation is static (before run-time) Eg: static in C/C++

② Fixed stack-dynamic array - subscript ranges are statically bound ~~and~~ but storage allocation is done at declaration elaboration time during execution. Eg: (arrays within C/C++ functions)

③ Stack-dynamic array - Both the subscript ranges and the storage allocation are dynamically bound at elaboration time (but fixed during lifetime). Eg: Ada arrays

④ Fixed Heap-dynamic array - similar to fixed stack-dynamic but bindings are done when the user program requests them during execution Eg: malloc/free and new/delete.

⑤ Heap-dynamic array - binding of subscript ranges and storage allocation is dynamic and can change any number of lines during the array's lifetime. Eg: ~~to~~ Perl and JavaScript arrays

→ Array Operations -

C → none

Ada → assignment, catenation, relational operators

Python → array catenation (+), element membership (in), equality (is, ==) etc

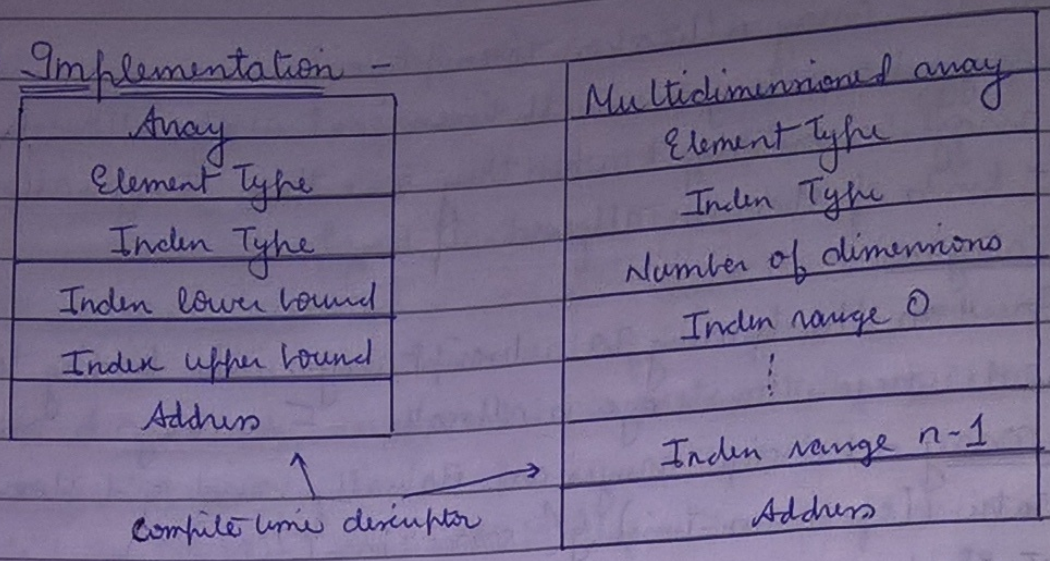
Fortran → elemental operations like pair wise addition.

→ Rectangular array - multi-dimensioned array in which all of the rows have the same number of elements and all columns have the same number of elements.

→ Jagged array - Rows with varying number of elements Eg - character arrays in C.

→ Slices - substructure of an array.

→ Implementation -



⑥ Associative Arrays -

It is an unordered collection of data elements that are indexed by an equal number of values called Keys.

→ Design Issue - form of references to their elements.
- In Perl, they are called hashes.
- In Python, they are called dictionaries.

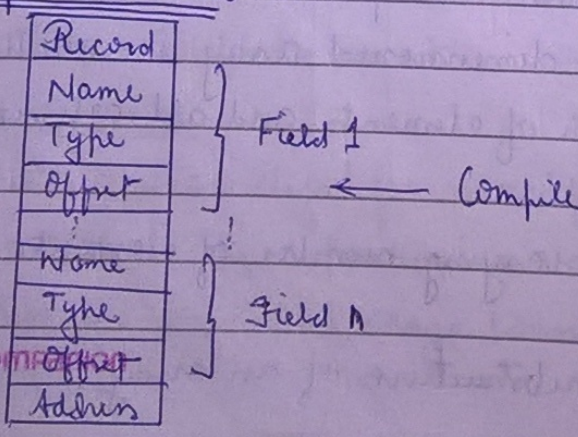
→ Implementation - In Perl and PHP uses hash function.

⑦ Record Types -

It is an aggregate of data elements in which the individual elements are identified by names and accessed through offsets from the beginning of the structure.

→ Design Issues -
→ What is the syntactic form of references to fields?
→ Are elliptical references allowed.

→ Implementation -



⑧ Union Types -

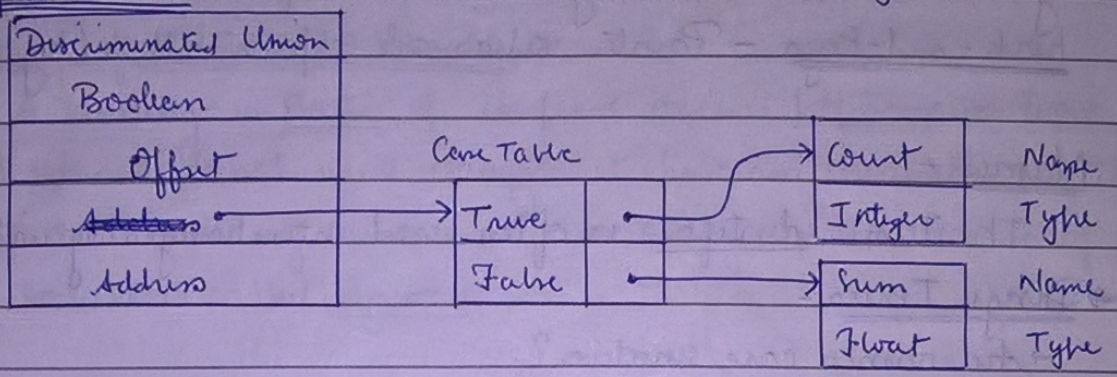
It is a type whose variables may store different type values at different times during program execution.

→ Design Issues -

→ Should type checking be required? Note that any such type checking must be dynamic.

→ Should unions be embedded in records?

→ Implementation - Discriminated ≡ Type indicator



⑨ Pointer and Reference Types -

A pointer type is one in which the variable have a range of values that consists of memory addresses and a special value, nil.

→ Design Issues -

→ What are the scope and lifetime of a pointer variable?

→ What is the lifetime of a heap-dynamic variable (the value a pointer references)?

→ Are pointers restricted as to the type of value to which they can point?

→ Are pointers used for dynamic storage management, indirect addressing, or both?

→ Should the language support pointer types, reference types, or both?

→ Pointer Problems -

Dangling pointers - Contains the address of a heap-dynamic variable that has been deallocated.

lost heap-dynamic variable - That is no longer accessible to the user program.

→ Reference Types - Similar to pointer but a reference refers to an object or a value in memory, while pointer refers to an address

→ Implementation -

→ Used in heap management

Representation - | SEGMENT | OFFSET |

Solution to the Dangling pointer problem -

Tombstone - Extra heap cell that is a pointer to the heap-dynamic variable. Costly in time and space.

Lock-and-Keys - Pointer values are represented as (key, address) pairs

(10) Names -

The term identifier is often used interchangeably with name.

→ Design Issues -

→ Are names case sensitive?

→ Are the special words of the language reserved words or keywords?

→ Name Forms -

In most programming languages, Name have the same form: a letter followed by a string consisting of letters, digits & underscore characters

→ Special words -

Keyword - It is special only in certain contents. Eg - Fortran

Reserved word - It cannot be used as a user-defined name.

(11) Variables -

It is an abstraction of a computer memory cells or collection of cells. Variable can be categorized as a sextuple of attributes :-

(Name, address, Value, Type, lifetime, and Scope)

Aliases - If two variable names can be used to access the same memory location, they are called aliases.

→ l-value of a variable is its address.

→ r-value of a variable is its value.

(12) Concept of Binding -

A binding is an association between an attribute and an entity, such as between a variable and its type or value, or between an operation and a symbol.

The time at which a binding takes place is called binding time.

→ Possible Binding Times -

Language design time, language implementation time, compile time, load time, run time and link time.

Binding of Attributes to Variables -

A binding is static if it first occurs before run time begins and remains unchanged throughout program execution. If the binding first occurs during runtime or can change in the course of program execution, it is called dynamic.

(13) Type Checking -

It is the activity of ensuring that the operands of an operator are of compatible types.

Dynamic type binding requires types checking at run-time, which is called dynamic type checking.

(14) Strong Typing -

A programming language is strongly typed if type errors are always detected.

(15) Type compatibility -

Pre-defined scalar types: simple.

Structure types and user-defined types: use type equivalence rules (no coercion)

Name type equivalence - judge by name

Structure type equivalence - judge by structure

(16) Named constant -

It is a variable that is bound to a value only once

Advantages - Readability and Reliability

Used to parameterize programs

(17) Variable Initialization -

The binding of a variable to a value at the time it is bound to storage is called initialization. Eg:- int sum = 0;

(18) Sequence Control -

The control of the order of execution of the operations both primitive and user defined.

Implicit -- determined by the order of the statements in the source program or by the built-in execution model.

Explicit -- the programmer uses statements to change the order of execution

→ level of sequence control -

Expressions - computing expressions using precedence rules and parentheses.

Statements - sequential execution, conditional and iteration statements

Declarative programming - an execution model that does not depend on the order of the statements in the source program.

Subprograms - transfer control from one program to another

→ sequencing with expressions -

We need to know operator's precedence and associativity.

Arithmetic expressions - prefix, postfix and infix notation.

→ Sequencing with conditional statements - if - else

→ Sequencing with loops - for, while, do - while

Handling special case in loops → break, continue, return, goto

→ Sequencing with exception handling - try - catch, throw, throws.