Chapter 6 – Data Types

* Introduction
  + Data type- defines a collection of data values and a set of predefined operations on those values
  + Descriptor- the collection of the attributes of a variable
* Primitive Data Types
  + Primitive data types- data types that are not defined in terms of other types
  + Numeric Types
    - Integer, stored using two’s compliment
    - Floating-point, models real numbers, but the representations are only approximations for many real values
      * Precision- accuracy of the fractional part of a value, measured as the number of bits
      * Range- combination of the range of fractions and, the range of exponents
    - Complex
    - Decimal
  + Boolean
  + Character Types
* Character String types
  + CST- one in which the values consist of sequences of characters
  + Design Issues
    - Should strings be a special kind of character array or a primitive type?
    - Should strings have static or dynamic length?
  + Strings and their operations
    - Substring reference- reference to a substring of a given string
    - Slices- substring references that are discussed in the more general context of arrays
    - Regular expressions- built-in pattern-matching operations that are somewhat loosely based on mathematical regular expressions
  + String Length Options
    - Static length String- the length can be static and set when the string is created
    - Limited Dynamic length String- Allows strings to have varying length up to a declared and fixed maximum set by the variable’s definition
    - Dynamic length Strings- Allows strings to have varying length with no maximum
  + Evaluation
* Enumeration Types
  + ET- one in which all of the possible values, which are named constants, are provided, or enumerated, in the definition
  + Enumeration constants- defined and grouped collections of named constants
* Array Types
  + Array- a homogeneous aggregate of data elements in which an individual element is identified by its position in the aggregate, relative to the first element
  + Arrays and Indices
    - Subscripts (or indices)- Specific elements of an array that are referenced by means of a two-level syntactic mechanism, where the first part is the aggregate name, and the second part is a possibly dynamic selector consisting of one or more items
  + Subscript Bindings and Array Categories
    - Static array- one in which the subscript ranges are statically bound, and storage allocation is static
    - Fixed stack-dynamic array- one in which the subscript ranges are statically bound, but the allocation is done at declaration elaboration time during execution
    - Fixed heap-dynamic array- subscript ranges and the storage binding are both fixed after storage is allocated
    - Heap dynamic array- binding of subscript ranges and storage allocation is dynamic and can change any number of times during the array’s lifetime
    - Rectangular Array- multidimensional array in which all of the rows have the same number of elements and all of the columns have the same number of elements
    - Jagged array- lengths of the rows need not be the same
    - Slices- of an array, is some structure of that array
  + Implementation of Array Types
    - Row major order- elements of the array that have as their first subscript the lower bound value of that subscript are stored first, followed by the elements of the second value of the first subscript, and so forth
* Associative Arrays
  + AA- unordered collection of data elements that are indexed by an equal number of values called keys
  + Structure and Operations
* Record Types
  + Record- aggregate of data elements in which the individual elements are identified by name and accessed through offsets from the beginning of the structure
  + Definitions of Records
    - Fields- record elements that are not referenced by indices
    - Level numbers- numerals that begin the lines of the record declaration
  + References to Record Fields
    - Dot notation- components of the reference relate to periods
    - Fully qualified reference- one in which all intermediate record names, from the largest enclosing record to the specific field, are named in the reference
    - Elliptical references- an alternative to fully qualified references (in COBOL)
* Tuple Types- data type that is like a record, except that the elements are not named
* List Types
  + List comprehension- an idea derived from set notation
* Union Types
  + Union- type whose variables may store different type values at different times during program execution
  + Free union- unions where programmers are allowed complete freedom from type checking in their use
  + Tag (or discriminant)- type indicator that is included in type checking of unions
  + Discriminated union- union with a discriminant
* Pointer and Reference Types
  + Pointer- the variables have a range of values that consists of memory addresses and a special value, nil
  + Nil- not a valid address and is used to indicate that a pointer cannot currently be used to reference a memory cell
  + Heap- dynamically allocated storage which can be accessed by a pointer
  + Heap-dynamic variables- variables that are dynamically allocated from the heap
  + Anonymous variables- variables without names
  + Reference types and Value types- categories of variables that are used to reference some other variable, rather than being used to store data
  + Dereferencing- the second fundamental pointer operation
  + Dangling pointer- pointer that contains the address of a heap-dynamic variable that has been deallocated
  + Lost heap-dynamic variable- allocated heap-dynamic variable that is no longer accessible to the user program
  + Garbage- variables that are not useful for their original purpose, and they also cannot be reallocated for some new use in the program
  + Reference Type- variable that is similar to a pointer, with one important and fundamental difference: a pointer refers to an address in memory, while a reference refers to an object or a value in memory
  + Tombstones- ever heap-dynamic variable includes a special cell, called a tombstone, that is itself a pointer to the heap-dynamic variable
  + Locks-and-keys approach- pointer values are presented as ordered pairs, where the key is an integer value
  + Reference counters- reclamation is incremental and is done when inaccessible cells are created
  + Mark-sweep- in which reclamation occurs only when the list of available space becomes empty
  + Eager approach and lazy approach
  + Deferred reference counting- avoids reference counters for some pointer
  + Incremental mark-sweep- garbage collection that occurs more frequently, long before memory is exhausted, making the process more effective in terms of the amount of storage that is reclaimed
* Type checking
  + TC- activity of ensuring that the operands of an operator are of compatible types
  + Compatible- type that is either legal for the operator or is allowed under language rules to be implicitly converted by compiler-generated code to a legal type
  + Coercion- automatic conversion
  + Type error- application of an operator to an operand of an inappropriate type
  + Dynamic Type checking- dynamic type binding requires type checking at run time
* Strong Typing
  + ST- highly valuable language characteristic
  + Strongly typed- if type errors are always detected
* Type Equivalence
  + Name type equivalence- two variables have equivalent types if they are defined either in the same declaration or in declarations that use the same type name
  + Structure type equivalence- means that the two variables have equivalent types if their types have identical structures
  + (Ada) Subtype- possibly range-constrained version of an existing type
* Theory and Data Types
  + Type system- set of types and the rules that govern their use in programs