# Data Structures using C and C++

Yedidyah Langsam
Moshe J. Augenstein
Aaron M. Tenenbaum

Chapter 1

# Introduction to Data Structures



## Information and Meaning

- The basic unit of information is the bit, whose value asserts one of two mutually exclusive possibilities.
- The binary digits 0 and 1 are used to represent the two possible states of a particular bit.

### Binary and Decimal Integers

- The most widely used method for interpreting bit settings as nonnegative integers is the **binary number system**.
- In this system each bit position represents a power of 2.
- The right most position represents $2^0$ and the left most position represents $2^n$.
- Any string of bits of length n represents a unique nonnegative integer between 0 and $2^n - 1$.
- For representing negative integers **ones complement notation** and **twos complement notation** methods are used.
- **Ones complement notation** – change each bit in its absolute value to the opposite bit.
- A bit string starting with 0 represents a positive number and a bit string starting with 1 represents a negative number.

- N bits can represent numbers in range: $-2^{(n-1)} + 1$ to $2^{(n-1)} - 1$.
- A string of bits consisting of all 0's or all 1's both could mean 0 following the above pattern.
- **Twos complement notation** – 1 is added to ones complement representation of a negative number.
- N bits can represent numbers in range: $-2^{(n-1)}$ to $2^{(n-1)} - 1$.
- Note: There is only one representation for 0 in this method. Also, -2(n-1) can be represented only in this method but its absolute value cannot be represented in either notation in n bits.
- **Binary Coded Decimal** – each digit is represented by four bits and then all the four bit sets are arranged in the order of their decimal counter.
- It must be noted that not all four set bits are valid as there are digits only form 0 – 9 while a four set bit can represent 16 possible states.

## Real Numbers

- Computers use **floating point notation** to represent real numbers.
- There are many variations to it but the key concept is that a real number is represented by a number called **mantissa**, times a **base** raised to an integer power, called an **exponent**. The base is usually fixed and mantissa and exponent vary to generate different real numbers.
- Mantissa is chosen such that it is an integer with no trailing 0s.
- An example can be that for a 32-bit string representing real number fist 24-bit is mantissa and last 8-bit exponent.
- The advantage of floating point notation is that it can be used to represent numbers with extremely large or extremely small absolute values.
- The limiting factor on the precision to which numbers can be represented on a particular machine is the number of significant binary digits in mantissa.

## Character Strings

- Nonnumeric objects are represented in character string form.
- For and 8 bit representation of characters 256 different characters can be represented.

- Generally, character string (STR) is represented by the concatenation of the bit strings that represent the individual characters of the string.
- The number of bits necessary to represent a character in a particular computer is called the **byte size** and a group of bits of that number is called a **byte**.
- A method of interpreting a bit pattern is often called a **data type**.

- We can say that information itself has no meaning. Any meaning can be assigned to a particular bit pattern, as long as it is done consistently.

## Hardware and software

- The **memory** (also called **storage** or **core**) of a computer is simply a group of bits.
- The setting of a bit is called its **value** or its **content**.
- The bits in a computer memory are grouped together into larger units such as **bytes**. Some computers group several bytes into units called **words**.
- Each unit is assigned an **address/location**, which is a name identifying a particular unit among all the units in memory.
- Every computer has a set of native data types, i.e., it is constructed with a mechanism for manipulating bit patterns consistent with the objects they represent.

- It is the programmers responsibility to know which data type is contained in which location.
- High level programming languages aid in this task by allowing to create **identifier**s which reference a memory location.
- Identifiers provide convenience in referring to a memory location using the identifier instead of the numerical address.

- Operators like “+” are **generic** and have several meanings depending upon its context.
- The compiler relieves the programmer of specifying the type of addition by examining the content and using the appropriate version.
- Declarations in high level languages play a key role as they specify how the contents of the content memory are to be interpreted by the program.

## Concept of implementation

- When we remove the hardware capabilities of the computer from the concept of data-type, a limitless number of data types can be considered.
- A **data type** is an abstract concept defined by a set of logical properties.
- After defining a data type and specifying the legal operations we can implement it.
- An implementation can be a **hardware implementation**, in which the circuitry necessary to perform the required operation is designed and constructed as part of computer; or it may be a **software implementation**, in which a program consisting of already existing hardware instructions is written to interpret bit strings in the desired fashion to perform the required operations.

- Sometimes we desire to add a quantity to an address to obtain another address. We cannot use + to do so as it is reserved for integer contents. We can use the notation a[x] to refer to the new location.
  Here x is added to the address a.
- Data type is a method of treating the contents of memory and that those contents have no intrinsic meaning.

- All strings in C are terminated by the special character '\0', which doesn't appears in string but is automatically placed by the compiler at the end of every string. When length of string is not known in advance we can use this property to check if we have reached the end of the string.
- C has a disadvantage of not having length of character string available readily, which is an offset to the advantage of having no arbitrary limit placed on the length of the string.

## Abstract Data Types

- The term **abstract data types** refers to the basic mathematical concept that defines the data type. Fundamentally, a data type is a collection of values and a set of operations on those values. That collection and those operations form a mathematical construct that may be implemented using a particular hardware or software data structure.
- An ADT consists of two parts:

    - A value definition: defines the collection of values for ADT and consist of two parts:
        - Definition clause
        - Condition clause
    - An operator definition
- The keywords **abstract typedef** introduce a value definition and the keyword **condition** is used to specify any conditions on the newly defined type.
- The definition clause is required but condition clause may not be necessary for every ADT.
- Operator definition is defined as an abstract function with three parts: a header, the optional preconditions, and the postconditions.

## Data Types in C

- The C language contains four basic data types: int, float, char and double.
- A variable declaration in C specifies two things:
    - The amount of storage that must be set aside for objects declared with that type.
    - Specifies how data represented by strings of bits are to be interpreted.

## Pointers in C

- In C programmer can reference to the location of objects as well as the objects themselves.
- Example, for x defined as an integer, &x refers to the location that has been set aside to contain x. &x is called a pointer.
- It is possible to declare variables whose data type is pointer and whose possible values are memory locations.
- Eg: int *pi;  float *pf;  char *pc;
- The asterisk (*) indicates that the values of the variables being declared are pointers to values of the type specified in the declaration rather than objects of that type.
- The notation *pi would refer to the value at the location referenced by the pointer pi.

## Data Structures and C

- The study of data structures involves two complementary goals:

    - To identify and develop useful mathematical entities and operations and to determine what classes of problems can be solved using these entities and operations.
    - To determine representations for those abstract entities and to implement the abstract operations on these concrete representations.
- Often, no implementation, hardware or software, can model a mathematical concept completely. It is important to recognize the limitations of a particular implementation.
- One important consideration in any implementation is its efficiency. Efficiency is usually measured by two factors: time and space.
- The choice of implementation involves a careful evaluation of the trade-offs among the various possibilities.

# Arrays in C

- The simplest form of array is a one-dimensional array that may be defined abstractly as a finite ordered set of homogenous elements.
- Before a value has been assigned to an element of the array, its value is undefined and a reference to it in an expression is illegal.
- The smallest element of an array's index is called its lower bound and the highest element its upper bound. In C, 0 is the lower index and n-1 the upper index for an array of size n.
- In C the lower and upper bound of an array cannot be changed during a program's execution.

## The Array as an ADT

```
abstract typedef <<eltype,ub>> ARRTYPE(ub,eltype)
condition type(ub) == int;

abstract eltype extract(a,i)    /* written a[i] */
ARRTYPE(ub,eltype) a;
int I;
```

```
precondition 0 <= i < ub;
postcondition extract == a;

abstract store(a,i,elt)  /* written a[i] = elt */
ARRTYPE (ub,eltype) a;
int I;
eltype elt;
precondition 0 <= i < ub;
postcondition a[i] == elt;
```

Here ARRTYPE(ub,eltype) denotes an ADT for a C array type eltype array[ub]. eltype is a type indicatior and ub upperbound.

## Using One-Dimensional Arrays

- A one-d array is used when it is necessary to keep a large number of items in memory and reference all the items in a uniform manner.
- The advantage of an array is that it allows the programmer to declare only a single identifier and yet obtain a large amount of space. Further with the help of a loop this large space can be accessed to reference each element in an uniform manner.
- A particular element of an array can be accessed through its index.

## Implementing One-Dimensional Arrays

- `int b[100];`
  This declaration reserves 100 successive memory locations, each large enough to contain a single integer.
- The address of the first of these locations is called the base address of the array b and is referenced by b[0].
- In general a reference to b[i] is to the element at location **base(b) + i*esize**, where esize is the size of elements and base(b) the base address.
- In C array variable is implemented as a pointer variable. An asterisk doesnot appear in declaration because the [] automatically imply that the variable is a pointer.
- The difference between int *b; and int b[100]; is that the first one declares a pointer variable of type int but the second one also reserves memory for 100 integers.

- Now b[i] can also be understood as *(b+i), which according to pointer arithmetic will add i times the size of data type of b to the base address of b.
- In C strings are also implemented as an array of characters.

## Array as Parameters

- Since an array variable in C is a pointer, array parameters are passed by reference rather than by value.
- The array range needs to be passed separately.
- Instead of copied the contents of array to another variable, the base address of the array is passed.
- This is more efficient both in time and space.

## Character strings in C

- A string in C is defined as an array of characters.
- Each string is terminated by the NULL character, indicating the end of string.
- A string constant is denoted by any set of characters included in double-quote marks.
- The NULL character (\0) is automatically appended to the end of the characters in a string constant when they are stored.

## Two-Dimensional Arrays

- int a[2][3];
- This declaration defines an array a of 2 rows with 3 columns each.
- One can think of 2-d arrays as a table but in memory they are stored most likely in a linear fashion.
- One method is row-major representation, in which first row occupies the first set of memory locations reserved for the array, the second row occupies the next set and so forth.
- So the address of a[i][j] is at base(a) + (i*r+j)*esize for array of size r*c.
- Another possible implementation can be to declare an array ar with upper bounds u1 and u2 consisting of u1+1 1-d arrays. The ith element of ar will be a pointer to a one-d array whose elements are elements of a[i].
- To reference a[i][j] first reference ar[i] to get the pointer to the array of a[i] and then access the jth element in it.

## Multidimensional Arrays

C allows an arbitrary number of dimensions. C allows to define arrays of more than 2 dimensions.

# Structures in C