

Data Structures using C and C++

Yedidyah Langsam

Moshe J. Augenstein

Aaron M. Tenenbaum

Chapter 1

Introduction to Data Structures

1.1 Information and Meaning

Binary and Decimal Integers, Real Numbers, Character Strings, Hardware and Software, Concept of Implementation, Abstract Data Types, Sequences as Value Definitions, ADT for Varying-length Character Strings, Data types in C, Pointers in C, Data Structures in C

1.2 Arrays in C

The Array as an ADT, Using one-dimensional Arrays, Implementing One-Dimensional Arrays, Arrays as Parameters, Character Strings in C, Character String Operations, Two-Dimensional Arrays, Multidimensional Arrays

1.3 Structures in C

Implementing Structures, Unions, Implementation of Unions, Structure Parameters, Representing other data structures, rational numbers, Allocation of storage and scope of variables

1.4 Classes in C++

The Class Rational, Using the class Rational, Implementing the methods, overloading, inheritance, constructors

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 from 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 first 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 an 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 **identifiers** 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 appear 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.
- Pg 14