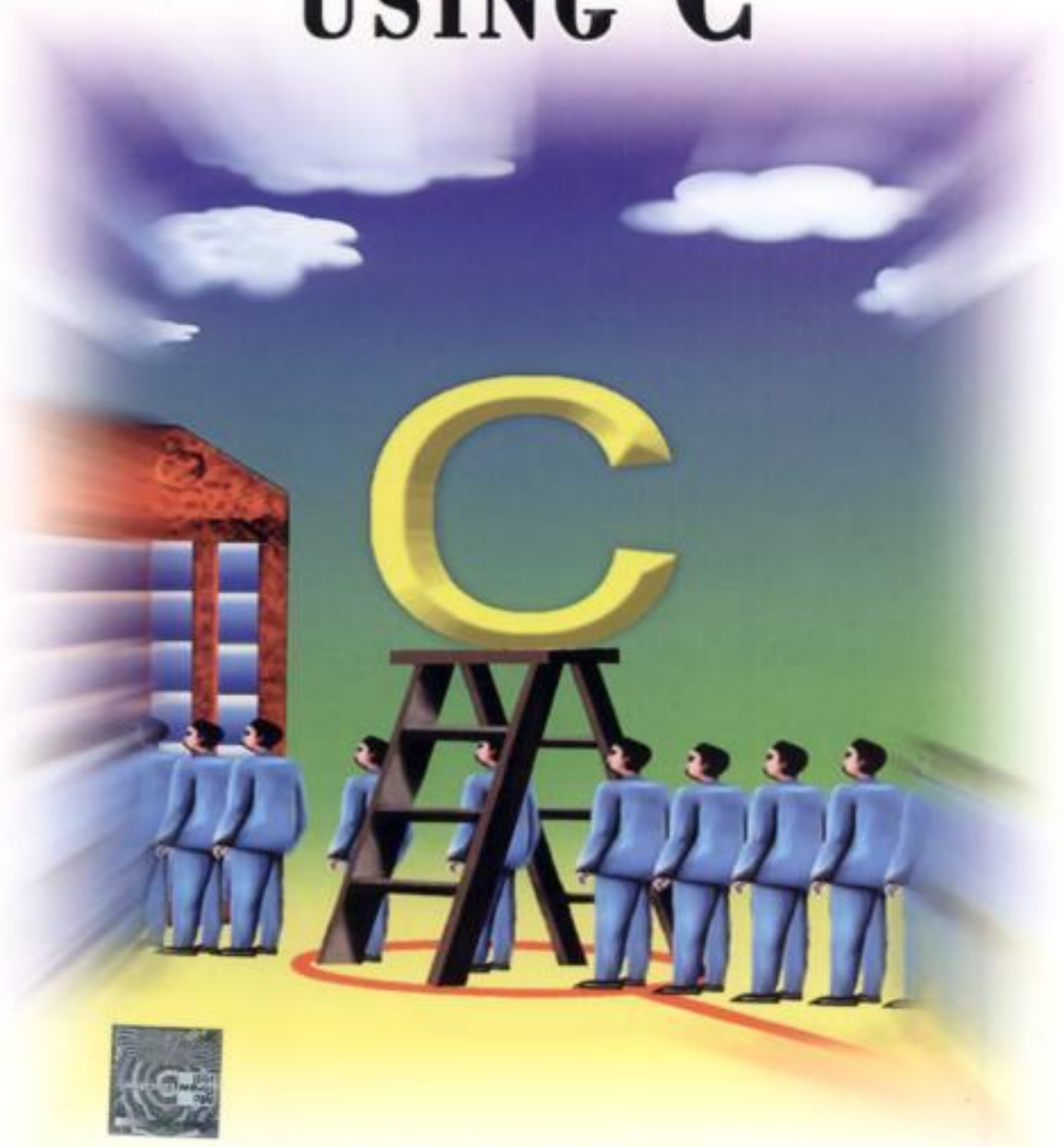
The McGraw-Hill Companies

ace series

DATA STRUCTURES USING C



ISRD Group

Copyrighted material

Copyrighted material

Data Structures Using C

This one
XLZ8-KWT-5ER8
Copyrighted material

Copyrighted material

Data Structures Using C

Instructional Software Research and Development (ISRD) Group

Lucknow

Tata McGraw-Hill Publishing Company Limited

NEW DELHI

McGraw-Hill Offices

New Delhi New York St Louis San Francisco Auckland Bogotá Caracas
Kuala Lumpur Lisbon London Madrid Mexico City Milan Montreal
San Juan Santiago Singapore Sydney Tokyo Toronto

Copyrighted material

Information contained in this work has been obtained by Tata McGraw-Hill, from sources believed to be reliable. However, neither Tata McGraw-Hill nor its authors guarantee the accuracy or completeness of any information published herein, and neither Tata McGraw-Hill nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that Tata McGraw-Hill and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

Tata McGraw-Hill

Copyright © 2006, by Tata McGraw-Hill Publishing Company Limited and UPTEC Computer Consultancy Limited.

No part of this publication may be reproduced or distributed in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise or stored in a database or retrieval system without the prior written permission of the publishers. The program listings (if any) may be entered, stored and executed in a computer system, but they may not be reproduced for publication.

First reprint 2007
RADYCRXYRZYLC

This edition can be exported from India only by the publishers,
Tata McGraw-Hill Publishing Company Limited.

ISBN 0-07-059102-4

Published by the Tata McGraw-Hill Publishing Company Limited,
7 West Patel Nagar, New Delhi 110 008, typeset in Times New Roman at
Tej Composers, WZ-391, Madipur Village, New Delhi 110 063 and
printed at Pashupati Printers (P) Ltd. 429/16 Gali No. 1, Friends Colony,
Industrial Area, G T Road, Shahdara Delhi 110 095

Cover Printer: SDR

The McGraw-Hill Companies

Copyrighted material

The ISRD Group

The Instructional Software Research and Development (ISRD) Group has been established by UPTEC Computer Consultancy Ltd. for content development. The Group is committed to develop and produce high quality learning and instructional material to facilitate all round professional development of students. The Group continuously strives to enrich their content repository by constantly researching and developing new learning and instructional material. The content thus developed is tailored to take various forms including text books, teaching aids, lecture notes, lab assignments and exercises, both in print as well as in electronic media (CDs etc.).

Many professionals have helped in creating UPTEC series of 'A' Level text books, and each of their contributions have been valuable. This particular book has taken its final shape with the noteworthy contributions of

Ms. Geetanjali Khanduri She is a Microsoft Certified Systems Engineer with specialization in Messaging. She also has to her credit the Microsoft Certified Trainer certification. Presently she is working as a faculty member with UPTEC Computer Consultancy Ltd.

Mr. Ashish Srivastava. Holds a Masters degree in Computer Science and is a Microsoft Certified Solutions Developer. He is presently working as Systems Analyst in a reputed multi-national company in Bangalore.

The work would not have been possible without the enriching guidance and support of the following key professionals:

Er. Alok Singhal An Electronics and Telecommunication Engineer from Roorkee University (now IIT Roorkee), with a Master's degree in Systems Engineering. Over 30 years of experience in the areas of entertainment electronics, microcomputers and microprocessor-based products. In his last assignment he was General Manager and Head of Communication Division at UPTRON India Ltd and was handling manufacturing and marketing of EPABX, RAX, radio communication, office communication, data communication, and computer products. He is presently holding the position of Vice President at UPTEC.

Prof.K.K. Bhutani Holds his Bachelor's, Master's and Doctoral degree from the University of Allahabad, he has more than 35 years of experience in academics and industry. He has been a professor of Electronics and Head of Computer Centre, University of Allahabad. He has also worked as General Manager (Computers), Computronics India. He is former Chairman, Computer Engineering Board, Institution of Engineers (India). At present, he is associated with UPTEC as a Director.

Er. K.N. Shukla An Electronics Engineer from IIT, Kanpur. He has more than 30 years of experience in electronics and computer industry. Started his career with J. K. Electronics, Kanpur. He has held several managerial positions at UPTRON, HILTRON and other similar companies. He was Director (Technical) at UPTRON India Limited before joining UPTEC as wholetime Director.

Dr. R.K. Jaiswal Holds a Doctoral degree from the University of Lucknow and has more than 16 years of experience in academics. He started his career with Lucknow University as a Lecturer, and has worked under College Science Improvement Programme (COSIP) a Government of India scheme. He has to his credit more than a dozen research papers in national and international journals. He is presently holding the position of Assistant General Manager at UPTEC.

Er. R.K. Lele Holds a bachelor's degree in Electrical Engineering from IIT Kanpur. He has Design and Development experience in embedded software/firmware, system software, computer & telecommunication/

Copyrighted material

data communication hardware and exposure to real-time software in the area of switching & communication protocols. At managerial position, he has handled various design projects with engineers during all the phases of the product design cycle. He is presently holding the position of General Manager (Systems) at UPTEC.

Prof. S.K. Singh With a bachelor's degree from IIT Kharagpur and master's degree from University of Missouri (USA) he has more than 35 years of professional experience with Bell Systems (USA), IBM, Rashtriya Chemicals & Fertilizers Ltd, National Institute for Training in Industrial Engineering (NITIE), 10 years as Professor of Computers and MIS at Indian Institute of Management (IIM), Lucknow, before joining UPTEC. Prof. Singh has been the Founder Chairman-Post Graduate Programme at IIM, Lucknow. He has also been a consultant to many national/international organizations. UPTEC series of 'A' Level text books have been developed under his active guidance.

Dr. Upendra Kumar An Industrial Engineer, obtained his Ph.D from IIT, Delhi. Has more than 25 years of experience in industry, academics and research. Dr. Kumar has worked with BHEL Corporate Systems Group, Eicher, UPTRON, National Institute for Training in Industrial Engineering (NITIE) and Indian Institute of Management (IIM), Lucknow. Dr. Kumar had conceived, created and established UPTRON-ACL and Computer Consultancy and Services Division at UPTRON earlier, as General Manager and Divisional Incharge. Has been consultant to many national/international organizations and is presently working as the Managing Director of UPTEC Computer Consultancy Ltd.

Ms. Baljeet Kaur She has more than 10 years of experience in academics and has been developing instructional material for UPTEC. At present, she is Manager, Instructional Software Research and Development (ISRD) Group at UPTEC.

We are also indebted to **Mr. Surya Prakash Sharma, Mr. Vikash Bajpayee** and **Mr. Rajan Awasthi** for their help with the DTP work — drawing figures, creating tables, and preparing table of contents and indexes.

Copyrighted material

Contents at a Glance

Copyrighted material

Copyrighted material

Contents

Copyrighted material

Copyrighted material

Copyrighted material

Copyrighted material

Copyrighted material

Copyrighted material

Preface

Data Structure is a core module in the curriculum of almost every computer science programme. This subject makes the students learn the art of analyzing algorithms as well as distinguishing between the specification of a data structure and its realization within an available programming language. It involves identifying the problem, analyzing different algorithms to solve the problem and choosing most appropriate data structure to represent the data.

The value of an implementation ultimately relies on its resource utilization : time and space, and this requires capability of analyzing different factors. The coverage of this book is in line with the syllabus of Data Structure course being taught in the BE/B.Tech. (Computer Science), BCA and MCA programmes of various universities.

The goal of this book is to help you understand different concepts of data structures. It specifically covers the entire syllabus of "Data Structures Through 'C' Language" course as prescribed by DOEACC for its 'A' and 'B' Level Programmes. It contains additional topics that be useful for the students of BE/ B.Tech. (Computer Science), BCA and MCA programmes of various universities.

Chapter 1 gives an introduction to the core topics of data structures. It unleashes the concepts of abstract data types (ADTs), data types, data structures and other useful tools that can be used to solve problems.

Chapter 2 provides an introduction to software engineering and also explores the principles of good program design, the approaches to algorithm design and the analysis of the algorithms.

Chapter 3 discusses one of the linear data structure — Arrays. The storage representation of arrays and various operations possible on arrays have also been explained in this chapter.

Chapter 4 gives an introduction to Linked List, which is again a linear data structure like array, but uses dynamic memory allocation rather than static memory allocation as in the case of arrays.

Chapter 5 presents the two most important applications of arrays and linked lists—Polynomials and Sparse Matrix.

Chapter 6 discusses one of the most important data structures—Stack. The representation of stacks through arrays and linked lists have also been explored in this chapter.

Chapter 7 deals with Queues. Various kind of queues—Circular queues, Dequeues, Priority queues—have been discussed. The chapter also presents the application of queues.

Chapter 8 explores one of the most important non-linear data structures i.e., Trees. The ways to represent binary trees through arrays and linked lists and the various operations and its traversal has also been discussed in this chapter.

Chapter 9 provides a detailed description of AVL trees. The other kinds of trees—Forests, Orchards and Expression trees—are also discussed in this chapter.

Copyrighted material

Chapter 10 explores the Multiway trees. The searching, insertion and deletion operations on a B-tree have also been discussed in this chapter.

Chapter 11 focuses on various searching and sorting methods. The implementation of various sorting methods—Bubble sort, Selection sort, Quick sort, Insertion sort, Merge sort, Radix sort, Heap sort—also form part of this chapter.

Chapter 12 discusses Graphs in detail. Various basic terms and applications of graphs have been explored in this chapter.

Chapter 13 is all about Hashing—a technique which provides a conceptually different mechanism to search a table for a given key value.

Chapter 14 gives an introduction to Strings. The representation of strings through arrays and linked lists and various string operations have been covered in this chapter.

Chapter 15 includes some of the techniques and algorithms that could be used to provide various levels of storage management and control.

Appendix A explains various mathematical concepts—Matrices, Polynomials, Logarithms, Factorials, etc.

Appendix B at the end of this book is a collection of question papers from July 2002 to July 2004 which will be useful for the students in preparation of the DOEACC exams.

We look forward to receiving feedback and suggestions from the users of this book. It will help us improve the future editions of this book.

ISRD Group

Copyrighted material

Syllabus

OBJECTIVE OF THE COURSE

The objective of the course is to introduce the fundamentals of Data Structures, Abstract concepts and how these concepts are useful in problem solving. After completion of this course student will be able to:

- Understand and use the process of abstraction using a programming language such as 'C'.
- Analyze step by step and develop algorithms to solve real problems.
- Implement various data structures viz. Stacks, Queues, Linked Lists, Trees and Graphs.
- Understand various searching and sorting techniques.

Given below is the outline of the course that an instructor can use for effective delivery of the course. The suggested time distribution for various topics are specifically in line with the syllabus prescribed for the subject by DOEACC for its 'A' and 'B' Level programmes.

Outline of Course

S.No	*Topic*	*Minimum Hours*
1.	Basic Concepts of data representation	03
2.	Introduction to Algorithm Design and Data Structure	06
3.	Arrays	05
4.	Stacks and Queues	08
5.	Linked lists	08
6.	Trees	10
7.	Searching, sorting and complexity	10
8.	Graphs	10
	Lectures =	**60**
	Practicals/Tutorials =	60
	Total =	**120**

Detailed Syllabus

1. Basic concepts of data representation **03 Hrs.**

Abstract data types: Fundamental and derived data types. Representation, primitive data structures.

Copyrighted material

2. Introduction to Algorithm Design and Data Structures **06 Hrs.**

Design and analysis of algorithm: Algorithm definition, comparison of algorithms. Top-down and bottom up approaches to Algorithm design. Analysis of Algorithm; Frequency count, Complexity measures in terms of time and space. Structured approach to programming.

3. Arrays **05 Hrs.**

Representation of arrays: single and multidimensional arrays. Address calculation using column and row major ordering. Various operations on Arrays. Vectors, Application of arrays: Matrix multiplication, Sparse polynomial representation and addition.

4. Stacks and Queues **08 Hrs.**

Representation of stacks and queues using arrays and linked-list. Circular queues, priority Queues and D-Queue. Applications of stacks: Conversion from infix to postfix and prefix expressions, Evaluation of postfix expression using stacks.

5. Linked Lists **08 Hrs.**

Singly linked list; operations on list. Linked stacks and queues. Polynominal representation and manipulation using linked lists. Circular linked lists, Doubly linked lists. Generalized list structure. Sparse Matrix representation using generalized list structure.

6. Trees **10 Hrs.**

Binary tree traversal methods: Preorder, In-order, Post-ordered traversal. Recursive and non-recursive Algorithm for above mentioned Traversal methods. Representation of trees and its applciations: Binary tree representation of a tree. Conversion of forest into tree. Threaded binary trees; Lexical binary trees. Decision and game trees. Binary search tree.: Height balanced (AVL) tree, B-trees.

7. Searching, Sorting and complexity **10 Hrs.**

Searching: Sequential and binary searches, indexed search, Hashing Schemes. Sorting: Insertion, selection, bubble, Quick, merge, radix, Shell, Heap sort. comparison of time complexity.

8. Graphs **10 Hrs.**

Graph representation: Adjacency matrix, Adjacency lists, Adjacency Multicasts. Traversal schemes: Depth first search, Breadth first search. Spanning tree: Definition, Minimal spanning tree algorithm. Shortest path algorithms (Prime's and Kruskal's).

The syallbus given above (specifically for DOEACC 'A' and 'B' Level programmes) has been covered in the first twelve chapters of the book. The last three chapters i.e., Chapter 13, 14 and 15 on Hashing, String Processing and Storage Management have been added to cover additional topics prescribed by other universities. The instructors can accordingly schedule these topics and cover the additional material given in the last three chapters according to the syllabus requirements of the concerned university/ institute.

Copyrighted material

CHAPTER

1 Introduction to Data Structures

Key Features

- Introduction to the Theory of Data Structures
- Data Representation
- Abstract Data Types
- Data Types
- Primitive Data Types
- Data Structure and Structured Type
- Atomic Type
- Difference between Abstract Data Types, Data Types and Data Structures
- Refinement Stages

The advancement in the study of data structures for analyzing algorithms has continued. The study of data structures has two objectives. The first is to identify and create useful mathematical entities and operations to determine what classes of problems can be solved by using these entities and operations. The second is to determine the representation of these abstract entities and to implement the abstract operations on these concrete representations.

This chapter introduces the core concepts of data structures. It explores the concepts of abstract data types (ADTs), data types, data structures and other useful tools that can be used to analyze and solve problems.

INTRODUCTION TO THE THEORY OF DATA STRUCTURES

The study of computer science encompasses the study of organization and flow of data in a computer. Data structure is the branch of computer science that unleashes the knowledge of how the data should be organized, how the flow of data should be controlled and how a data structure should be designed and implemented to reduce the complexity and increase the efficiency of the algorithm.

A course in data structures offers an excellent opportunity to introduce the concepts associated with object oriented programming. For example, the concept of classes in object-oriented programming language (like C++) is based on the key concept of Abstract Data Type (ADT). An ADT exhibits many of the concepts enshrined in the theory of data structures.

The theory of structures not only introduces you to the data structures, but also helps you to understand and use the concept of abstraction, analyse problems step by step and develop algorithms to solve real world problems. It enables you to design and implement various data structures, for example, the

stacks, queues, linked lists, trees and graphs. Effective use of principles of data structures increases efficiency of algorithms to solve problems like searching, sorting, populating and handling voluminous data.

Need of a Data Structure

A data structure helps you to understand the relationship of one data element with the other and organize it within the memory. Sometimes the organization might be simple and can be very clearly visioned, for example, list of names of months in a year. We can see that names of months have a linear relationship between them and thus can be stored in a sequential location or in a type of data structure in which each month points to the next month of the year and it is itself pointed by its preceding month. This principle is overruled in case of first and last month's names. Similarly, think of a set of data that represents location of historical places in a country (Fig. 1.1). For each historical place the location is given by country name followed by state name followed by city name, and then followed by the historical place name. We can see that such data form a hierarchical relationship between them and must be represented in the memory using a hierarchical type of data structure.

The above two examples clearly identify the usefulness of a data structure. A data structure helps you to analyze the data, store it and organize it in a logical or mathematical manner.

Fig. 1.1

DATA REPRESENTATION

Various methods are used to represent data in computers. Hierarchical layers of data structure are used to make the use of data structure easy and efficient. The basic unit of data representation is a **bit.** The value of a bit asserts one of the two mutually exclusive possibilities—0 or 1. Various combinations of two values of a bit are used to represent data in a different manner in different systems. Eight bits together form one **byte** which represents a **character** and one or more than one characters are used to form a **string**. A string can thus be seen as a data structure that emerges through several layers of data structures as shown in Fig. 1.2.

The representation of a string can be made easier (i.e. working with the strings without bothering about the NULL (\0) character at the end of the string) by wrapping it into another data structure which takes care of such intricacies and supports a set of operations that allows us to perform various string related operations like storing and fetching a string, joining two strings, finding the length of strings, etc.

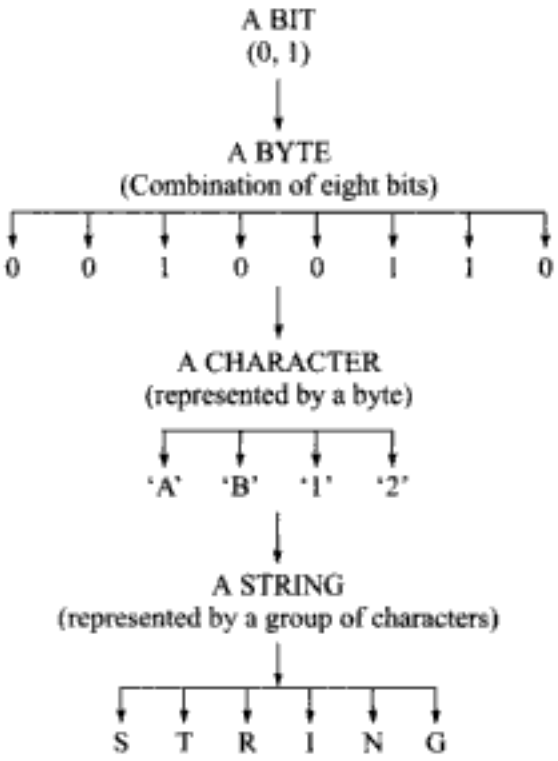


Fig. 1.2 *Data Representation*

Copyrighted material

The use of the concrete data structure during design creates lot of difficulties and requires much more efforts, such a problem can be avoided by using **abstract data type** in the design process. But before moving to the discussion of concepts of Abstract Data Types (ADTs), let us discuss how the primitive or basic data types of any language (i.e. integer, character, etc.) are internally represented in the memory.

Integer Representation

An integer is the basic data type which is commonly used for storing negative as well as non-negative integer numbers. The non-negative data is represented using **binary number system**. In this, each bit position represents the power of 2. The rightmost bit position represents 2^0 which is 1, the next represents 2^1 which is 2, then 2^2 which is 4 and so on. For example, 00100110 represents the integer as $2^1 + 2^2 + 2^5 = 2 + 4 + 32 = 38$.

For negative binary numbers the methods of representation used are **one's complement** and **two's complement**.

In **one's complement** method the number is represented by complementing each bit, i.e. changing each bit in its value to the opposite bit setting. For example, 0 0 1 0 0 1 10 represents 38, after complementing, it becomes 1 1 0 1 1 0 0 1 which is used to represent –38.

In **two's complement** method, 1 is added to one's complement representation of the negative number. For example, –38 is represented by 1 1 0 1 1 0 0 1 which on adding 1 to it will become

```
  1 1 0 1 1 0 0 1
              + 1
-----------------
  1 1 0 1 1 0 1 0,  which represents –38 in two's complement notation.
```

Real Number Representation

The method used to represent real numbers in computers is **floating-point notation**. In this notation, the real number is represented by a number called a **mantissa,** times a base raised to an integer power, called an **exponent**. For example, if the base is fixed as 10, the number 209.52 could be represented as 20952×10^{-2}. The mantissa is 20952 and exponent is – 2. Both the mantissa and exponents are two's complement binary integers. For example, 20952 can be represented as 1 0 1 0 0 0 1 1 1 0 1 1 in binary form.

Therefore, the 24 bit representation of the number will be 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 1 0 1 1 and 8 bit two's complement binary representation of –2 is 1 1 1 1 1 1 1 0; thus, the number is represented as 0 0 0 0 0 0 0 0 1 0 1 00 0 1 1 1 0 1 1 . 1 1 1 1 1 1 1 0.

Character Representation

The information in computers is not always interpreted numerically. For example, to store the salary of an employee we use the integer representation of the data but with salary we also need to store the name of the employee which requires a different data representation. Such information is usually represented in character string form. There are different codes available to store data in character form such as BCD, EBCDIC and ASCII.

For example, if 8 bits are used to represent a character, then up to 2^8=256 different characters can be represented as bit patterns. 1 1 0 0 0 0 0 0 is used to represent the character 'A' and 1 1 0 0 0 0 0 1 is used to represent character 'B'. Then, finally AB can be represented as 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1.

Copyrighted material

ABSTRACT DATA TYPES

An **Abstract Data Type** (ADT) is defined as a mathematical model of the data objects that make up a data type as well as the functions that operate on these objects. An abstract data type is the specification of logical and mathematical properties of a data type or structure. ADT acts as a useful guideline to implement a data type correctly. The specification of an ADT does not imply any implementation consideration. The implementation of an ADT involves the translation of the ADT's specification into syntax of a particular programming language. The important step is the definition of ADT that involves mainly two parts:

1. Description of the way in which components are related to each other
2. Statements of operations that can be performed on that data type.

For example, the **int** data type, available in the 'C' programming language provides an implementation of the mathematical concept of an integer number. The int data type in 'C' can be considered as an implementation of Abstract Data Type, INTEGER-ADT. INTEGER-ADT defines the set of numbers given by the union of the set {-1, -2, -3....∞} and the set of whole numbers {0, 1... + ∞}. INTEGER-ADT also specifies the operations that can be performed on an integer number, for example, addition, subtraction, multiplication, etc. In the specification of the INTEGER-ADT, following issues must be considered:

- Range of numbers that will be represented.
- Format for storing the integer numbers.

This determines the representation of a signed integer value.

Example of an ADT Specification for a Stack

A stack is a set of finite number of elements, where insertion or removal of an element is allowed from one end only. Any new element that joins the set is kept at the top and only the element at the topmost position can be taken out. In other words, a stack can be defined as a list where the only element that is accessible is the most recently inserted one.

There are a number of methods for specifying an ADT. The example below uses an informal notation to specify STACK abstract data type.

Specification of Abstract data type STACK is as follows:

```
abstract typedef <1, 2, 3 .....n, top> STACK;
condition top == NULL
abstract STACK PUSH (data)
Precondition  :      top ! = n
operation     :      insert (data)
                     top → data
                     /* top points to the most recently inserted elements */
abstract             Pop ( )
Preconditions top ! NULL
operation     :      Remove (top)
                     top → current data
```

Copyrighted material

sample content of Data Structures Using C

- *click Art of Cycling: A Guide to Bicycling in 21st-Century America*
- download England To Me: A Memoir
- download online Asian Bites: A feast of flavours from Turkey through India to Japan
- Knit Kimono Too book
- click In My Father's Country: An Afghan Woman Defies Her Fate
- A Hidden Witch (A Modern Witch Series, Book 2) for free

- http://chelseaprintandpublishing.com/?freebooks/The-Rise-of-the-Cult-of-Rembrandt--Reinventing-an-Old-Master-in-19th-century-France.pdf
- http://nexson.arzamaszev.com/library/England-To-Me--A-Memoir.pdf
- http://www.experienceolvera.co.uk/library/Globalization--Effects-on-Fisheries-Resources.pdf
- http://junkrobots.com/ebooks/Knit-Kimono-Too.pdf
- http://econtact.webschaefer.com/?books/It-s-Not-Too-Late.pdf
- http://wind-in-herleshausen.de/?freebooks/Remembering-Whitney--My-Story-of-Love--Loss--and-the-Night-the-Music-Stopped.pdf