Mc Graw Hill Education 大学计算机教育国外著名教材系列 影印版

Data Structures Using C

数据结构

（C语言版）

R Krishnamoorthy
G Indirani Kumaravel 著

Mc Graw Hill
清华大学出版社

大学计算机教育国外著名教材系列（影印版）

Data Structures Using C

数据结构

（C 语言版）

R Krishnamoorthy
G Indirani Kumaravel 著

清华大学出版社
北京

R Krishnamoorthy, G Indirani Kumaravel
Data Structures Using C
EISBN: 0-07-066919-8

图书在版编目(CIP)数据

数据结构 = Data Strctures Using C：C 语言版：英文 /（印）克里斯哈拉莫斯（Krishnamoorthy, R.），（印）库玛纳维尔（Kumaravel, G. I.）著. —影印本. —北京：清华大学出版社，2009.9
（大学计算机教育国外著名教材系列）

ISBN 978-7-302-20731-3

Ⅰ. 数…　Ⅱ. ①克…　②库…　Ⅲ. ①数据结构－高等学校－教材－英文　②C 语言－程序设计－高等学校－教材－英文　Ⅳ. TP311. 12　TP312

中国版本图书馆 CIP 数据核字（2009）第 144169 号

责任印制：杨　艳

出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编：**100084
社　总　机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印　刷　者：北京市清华园胶印厂
装　订　者：三河市溧源装订厂
发　行　者：全国新华书店
开　　本：148×210　　**印张：**19.375
版　　次：2009 年 9 月第 1 版　　**印　　次：**2009 年 9 月第 1 次印刷
印　　数：1~3000
定　　价：39.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：010-62770177 转 3103　　产品编号：034111-01

出版说明

进入 21 世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从 1996 年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书（影印版）”等一系列引进图书，受到国内读者的欢迎和支持。跨入 21 世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列（影印版）”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列（影印版）”做得更好，更适合高校师生的需要。

清华大学出版社

ABOUT THE AUTHORS

R Krishnamoorthy, PhD is Professor of Information Technology, Bharathidasan Institute of Technology, Bharathidasan University, Trichy. Dr R Krishnamoorthy received his M. Tech Degree in Computer Science and Engineering from Indian Institute of Technology, Kanpur and PhD degree in Computer Science and Engineering from Indian Institute of Technology, Kharagpur, with specialization in Computer Vision and Image Processing. He has 24 years of teaching experience. He is the author of three books, and forty-four technical papers published in National and International Conferences and International Journals. He has produced five PhDs. He is member of CSI, ISTE, IEEE and ACM. His areas of interest include network security, image processing and software testing.

G Indirani Kumaravel is Senior Lecturer in Computer Science and Engineering, Annamalai University, Chidambaram. She received her M E degree in Computer Science and Engineering from Annamalai University. Indirani has 12 years of teaching experience. She is a member of CSI. Her areas of interest include Speech and Image Processing.

PREFACE

C programming language offers several facilities to group data together in convenient packages, or *data structures*. With the emergence of C as the most popular language of implementation, it has been used in this book to extensively examine data structures.

This Book is Meant for...

Keeping in mind the level of beginners, the book is written without any prerequisites. It is an ideal textbook for students of various courses in Computer Science at the diploma, polytechnic, undergraduate and postgraduate levels, and also for new programmers who wish to know about the usage of different data structures in their project.

Student Friendly Approach...

Students will gain a good appreciation of the subject as this book has a clear display of syntax and abundant programming examples. To simplify concepts, the data structures are implemented using C language, in a step-by-step manner.

Organisation of the Chapters...

Having understood the difficulties faced by beginners, an introductory material with *fundamentals of data structure* and *an introduction to C language* is presented in Chapter 1. Chapter 2 deals with *strings, their representation and operation*. Chapter 3 is devoted entirely to *stack data structure* as the same has many applications in different fields of Computer Science and Engineering. Various stack operations, implementation issues, and applications of stack data structure are clearly explained in this chapter. The *queue data structure* and *its types* such as circular queue, deque and priority queue are described in Chapter 4 along with their operations, implementations and applications. Chapter 5 offers a clear understanding of *linked list data structure*. Chapter 6 details the concepts of *tree data structure*. It starts with basic terminology and describes tree representation, operations, types and applications with illustrative programs. *Graph data structure* with its use, representation, implementation and applications are introduced in Chapter 7. Chapter 8 is completely devoted to *sorting techniques* as it has many applications in various areas of Computer Science and Engineering. *Different searching techniques* and *search trees* are emphasised in Chapters 9 and 10 respectively. Recent advances in search trees, Binary Search Trees, AVL, B, B+ and Trie Structures are also included in Chapter 10. *File structure along with various access strategies* are presented in Chapter 11.

The Key Pedagogical Features are...

In essence, this book is totally self-contained and provides good number of illustrations and tested programs that demonstrate the concepts.

- Every chapter begins with *an introduction* that elucidates key topics and provides basic background.
- *Solved examples, tables, figures and flow diagrams* interspersed throughout the book are a valuable reference that simplifies the understanding of constructing modular and reusable structures.
- *Programming code* featuring precise instructions helps the reader implement practical data structures, thereby enhancing program reliability.
- *Review Yourself, Multiple Choice Questions and Programming Exercises* are included at the end of every chapter to reinforce the understanding of concepts.
- *Applications of each data structure* are explained through *concepts and programming examples.*
- *Web supplements* are a valuable resource for students and instructors. The online learning centre contains Additional Problems, Sample Tests, Web Links and Reference Titles for the students, and Solution Manual and chapter-wise PowerPoint Slides for the instructors.

This Book is Outstanding Because...

DATA STRUCTURES USING C is unique, in the sense that it deals with both theoretical and programming aspects of different data structures. The novelty of this book is that it not only covers all the concepts of data structures but also explains the implementation issues with tested programs in all the chapters.

Acknowledgements...

The authors wish to acknowledge the services rendered by their students in testing the sample programs. Sincere thanks are also due to the colleagues who have provided constructive criticism and feedback on the concepts presented in this book. The authors are grateful to Mr. S. Raja Vel (M/S Vel Raj Computer Centre) for his valuable assistance. The authors extend their appreciation to the editorial and publishing team of McGraw-Hill Education for their support in bringing out this book. The in-depth feedback of the following reviewers has been invaluable.

Gursharan	Dept. of Physics & Computer Science Dayalbagh Educational Institute, Agra
D. M. Dalgade	Rajiv Gandhi Institute of Technology RGIT Campus, Mumbai
Bindu Aggarwal	Dept. of Computer Engineering Sikkim Manipal Institute of Technology, Sikkim
Amitava Nag	Dept. of Information Technology Academy of Technology, West Bengal
S. Sridhar	Dept. of Computer Science and Engineering College of Engineering, Anna University, Chennai

Suggestions for improvement are welcome.

AUTHORS

CONTENTS

Preface xi

1. Data Structures—An Overview 1

1.1 Introduction *1*
1.2 Data Types *3*
1.3 Program Modules *4*
1.4 Control Structures *7*
1.5 Looping Structures *13*
1.6 Arrays *16*
1.7 Structures *24*
1.8 Pointers *28*
1.9 Recursion *31*
Review Yourself 35
Multiple Choice Questions 36
Programming Exercises 38

2. Strings and Character Manipulation 39

2.1 Introduction *39*
2.2 Primitive Functions or Operations on Strings *39*
2.3 Representation of Strings *41*
2.4 String Manipulation in C *42*
2.5 String Manipulation Applications *58*
Review Yourself 66
Multiple Choice Questions 67
Programming Exercises 67

3. Stacks 68

3.1 Introduction *68*
3.2 Definition *68*
3.3 Primitive Operations *69*
3.4 An abstract Data Type (ADT) *70*
3.5 Implementation *70*

3.6 Applications of Stack 76
Review Yourself 99
Multiple Choice Questions 99
Programming Exercises 101

4. Queues 102
4.1 Introduction 102
4.2 Definition 102
4.3 Operations on a Queue 103
4.4 ADT for Queues 104
4.5 Representation of Queue 104
4.6 Various Other Queue Structures 112
4.7 Applications 152
Review Yourself 165
Multiple Choice Questions 165
Programming Exercises 165

5. Linked Lists 168
5.1 Introduction 168
5.2 Definition 168
5.3 ADT for Linked List 169
5.4 Singly Linked List 169
5.5 Doubly Linked List 184
5.6 Circular Linked Lists 198
5.7 Sparse Matrices 210
5.8 Applications 232
5.9 Additional Programs 254
Review Yourself 272
Multiple Choice Questions 272
Programming Exercises 273

6. Trees 274
6.1 Introduction 274
6.2 Definition 274
6.3 Terminologies Used 274
6.4 Binary Tree 276
6.5 Threaded Binary Trees 296
6.6 Heap Trees 317
6.7 Deaps 327
6.8 Huffman Algorithm 337

6.9 Decision Trees 344
6.10 Game Tree 351
6.11 Applications 358
Review Yourself 362
Multiple Choice Questions 362
Programming Exercises 363

7. Graphs **364**
7.1 Introduction 364
7.2 Definition 365
7.3 Terminologies Used 366
7.4 Representation of Graphs 369
7.5 ADT for Graphs 372
7.6 Extra Information that can be Retrieved from the Adjacency Matrix of the Graph 373
7.7 Operations on Graphs 374
7.8 Applications 410
7.9 Unweighted Shortest Path for Graphs Using Adjacency Matrix 439
7.10 Introduction to NP-completeness 444
Review Yourself 445
Multiple Choice Questions 445
Programming Exercises 446

8. Sorting **447**
8.1 Introduction 447
8.2 Definition 448
8.3 Internal Sorting 448
8.4 External Sorting 474
Review Yourself 483
Multiple Choice Questions 483
Programming Exercises 484

9. Searching **485**
9.1 Introduction 485
9.2 Quantity Dependent Search Techniques 485
9.3 Density Dependent Search Techniques 490
9.4 Indexed Search Techniques 509
Review Yourself 512
Multiple Choice Questions 512
Programming Exercises 513

10. Search Trees 514

10.1 Introduction 514

10.2 Binary Search Tree (BST) 514

10.3 AVL Trees 530

10.4 B - Trees 544

10.5 B+ - Trees 562

10.6 Tries 572

Review Yourself 587

Multiple Choice Questions 587

Programming Exercises 588

11. File Structures 589

11.1 Files 589

Review Yourself 599

Multiple Choice Questions 600

Programming Exercises 600

Index 602

DATA STRUCTURES—AN OVERVIEW

1

1.1 INTRODUCTION

Computer A computer is an electronic machine that accepts data and instructions (called programs), manipulates the data using the program and gives the information as result.

Data Data is a value or a set of values which does not give any meaning. It is generally a raw fact.

Examples i) 34 ii) C iii) 11/2/2000 iv) RAMA

Entity An entity is a 'thing' or 'object' in the real world that is distinguishable from all other objects. The entity has a set of properties or attributes and the values of some sets of these attributes may uniquely identify an entity. An entity set is a collection of entities.

Example

Entity	*Student*			
Attributes	Roll No.	Name	DOB	% of marks scored
Values	123	RAJA	11/12/1980	78%

All the students of a particular class constitute an entity set.

Domain Each attribute of an entity set has a range of values and is called the domain of the attribute. In other words, a domain is the set of all possible values that could be assigned to a particular attribute. For example, for the attribute *percentage-of-marks scored* the domain is {0 to 100}.

Information Information can be defined as meaningful data or processed data. When the raw facts are subjected to processing, we get a relevant piece of information as its result. Information also relates an entity and the values of the attributes of that entity.

Example Data (11/12/1980) becomes information if the entity RAJA is related to the Date of Birth attribute (11/12/1980) as follows:

Date of Birth of the student RAJA is 11/12/1980.

Fig. 1.1 shows the interrelation between data and information.

Data Structure A data structure is an arrangement of data in a computer's memory (or sometimes on a disk). Depending upon the arrangement of data, data structures can be classified as arrays, records, linked lists, stacks, trees, etc., We also require to have algorithms to manipulate the data in these structures.

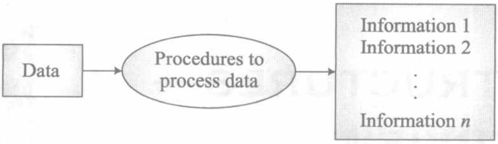


Fig. 1.1 *Relation between data and information*

A general understanding of the data structures is important in developing efficient algorithms in all phases of advanced data processing and computer science. A few of the applications of various data structures can be as follows:

1. Compiler design
2. Operating systems
3. Database management systems
4. Statistical analysis packages
5. Numerical analysis
6. Graphics
7. Artificial Intelligence
8. Simulation
9. Network analysis

That is, in many applications, different data structures are used to do the operations on the data structures. In such a situation, there is a tradeoff between memory utilization and run time. That is, one data structure sacrifies memory compactness for speed; another utilizes memory efficiently but results in a slow run time. So each data structure has its own strengths and weaknesses. They will be discussed fully as we study each data structure. Table 1.1 shows the characteristics of various data structures.

Algorithms for Data Structures Once a data structure for a particular application is chosen, an algorithm must be developed that manipulates the related data items stored in it. Such an algorithm should have the following features.

1. It should be free of ambiguity.
2. It should be efficient.
3. It should be concise and complex.

Classification of Data Structures In computer science, several data structures are available and are used depending on the area of applications. But a few data structures are used frequently almost in all application areas and they may be used to construct a complex data structure. These data structures are known as *fundamental data structures* or *classic data structures*. Fig. 1.2 shows the classification of fundamental data structures.

In a linear data structure, all the elements are arranged in a sequence (or) maintained in a linear ordering. In non-linear data structures, no such sequence is maintained for the elements and the elements are distributed over a plane. Fig. 1.3 shows the diagrammatic representation of various data structures.

Table 1.1 Characteristics of various data structures

Data Structure	*Advantages*	*Disadvantages*
1. Array	Quick insertion, very fast access if index is known	Fixed size, slow speed in searching, insertion and deletion
2. Stack	Provides Last in First out Access	Slow access to other items
3. Queue	Provides First in First out access	Slow access to other items
4. Linked list	Quick insertion, quick deletion, waste of main memory is less	Slow search
5. Binary tree	Quick search, insertion, deletion (if tree remains balanced)	Deletion algorithm is complex
6. Red–black tree	Quick search, insertion, deletion, tree always balanced	Complex
7. 2-3-4 tree	Quick search, insertion, deletion, tree always balanced, similar trees good for disk storage	Complex
8. Hash table	Very fast access if key is known, fast insertion	Slow deletion, access slow if key is not known, inefficient memory usage
9. Heap	Fast insertion, deletion access to largest	Slow access to other items
10. Graph	Models real-world situations	Some algorithms are slow and complex

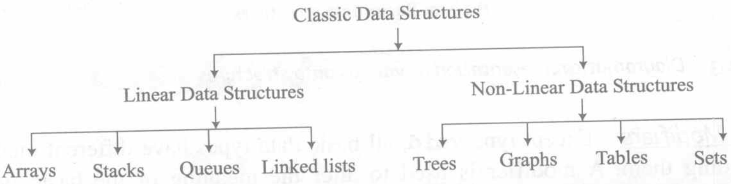


Fig. 1.2 *Classification of data structures*

1.2 DATA TYPES

A data type is a term which refers to the kind of data. Every programming language has its own set of built-in data types. In C, the following are the basic data types.

`int`, `long`, `char` and `void`

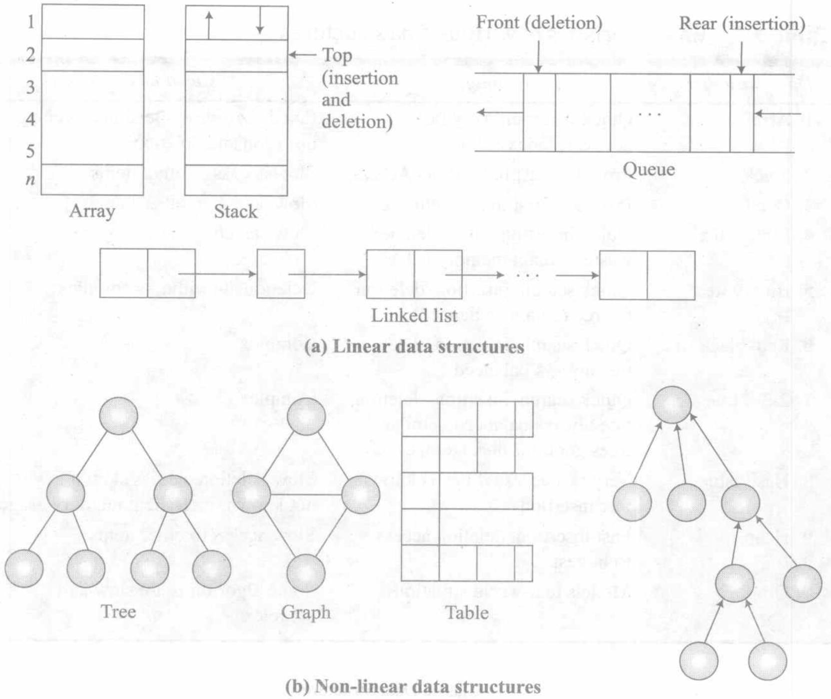
Declaring Variables The syntax used to declare the variables is as follows

```
<data type> variable(s)
```

where variables are separated by a comma.

Example

```
int a,b,c
char d,e
float g,h
```



(a) Linear data structures

(b) Non-linear data structures

Fig. 1.3 *Diagramatic representation of various data structures*

Type Modifiers Except type **void**, all basic data types have different modifiers preceding them. A modifier is used to alter the meaning of the basic type to appropriately suit the requirements of various situations. Now let us discuss the different basic data types and their modifiers.

Integer These are numbers without fractional parts that may optionally be prefixed with a minus sign. The type modifiers that may be used with this type are

```
long, short, unsigned long, unsigned short
```

Float These are numbers with decimal or fractional parts. The type modifiers that may be used with this type are

```
float, double, long double
```

Character Any single letter enclosed within single quotes is called a character data type. Table 1.2 shows in general the number of bytes allocated in a computer's memory for each data type.

1.3 PROGRAM MODULES

A program module is nothing but a set of statements used to achieve a particular task. It may be the main program itself or procedures and functions. In C, the program

Table 1.2 *Storage requirements of the various data types*

Data type	*Number of bytes allocated*	
char	1 byte	0 to 255
int	2 bytes	–32768 to +32767
float	4 bytes	stores up to 6 decimal places
double	8 bytes	stores up to 12 decimal places
long double	10 bytes	stores up to 16 decimal places

module may be the main function or any other function. The syntax of any program module is as follows.

```
<Return data type> function name (Declaration of parameter 1,
   parameter 2, …., parameter n)
{
   Declaration of local variables;
   <function body>
   return <value>
}
```

A program module may return any value to the invoking function or calling function by means of the return statement. Sometimes a program module may be used to achieve a particular task. In that case, the invoked program module (function) need not return any value to the calling or invoking function. A program module can pass arguments to another program module either by reference or by value. If the arguments are passed by value then any change that takes place for the arguments in the called function will not get reflected in the calling function. Some sample C programs are given below to illustrate these concepts.

Example 1 /*calling functions with arguments and return values*/

```
main ( )
{
   int i, f ;
   int fact (int);
   printf (“Enter a number (only positive number):”);
   scanf (“% d”, & i) ;
   if (i = = 0)
   printf (“Factorial of % d is 1”, i) ;
   else
   {f = fact (i) ;
   printf (“Factorial of % d is % d”, i, f) ;
   }
}
   int fact (int n)
{
   int i, f = 1 ;
   for (i = 1; i < = n; i++)
   f=f*i;
   return (f);
}
```

Sample Input and Output

Enter a number (only positive number) : 5

Factorial of 5 is 120

Example 2 /*calling functions with arguments but no return values*/

```
void main ( )
{
   int a, b, c ;
   printf ("Enter 3 numbers a, b, c :") ;
   scanf ("%d %d %d", & a, &b, &c);
   big (a, b, c) ;
}
   big (int x, int y, int z)
{
   if ((x > y) && (x > z))
   printf ("\n a is the biggest”) ;
   else if ((y > z) && (y > x))
   printf ("\n b is the biggest”) ;
   else
   printf ("\n c is the biggest”) ;
}
```

Sample Input and Output

Enter 3 numbers a, b, c : 5 10 15

c is the biggest

Example 3 /*calling function by value */

```
main ( )
{
   int a, b;
   printf (“Enter values for a and b:/n”);
   scanf (“%d %d”, &a, &b);
   fun (a, b);
   printf (“The values of a and b after executing the function are
   %d %d\n”, a, b);
}
fun (int d, int e)
{
   d = d*5;
   e = e*7;
   printf (“The value of a and b inside the function is %d %d\n”, d, e);
}
```

Sample Input and Output

Enter values for a and b: 2 3

The value of a and b inside the function is: 10 21

The values of a and b after executing the function are: 2 3

If the arguments are passed to a function by reference then any change that takes place for the arguments in the called function will get reflected in the calling function. The following program illustrates this idea.