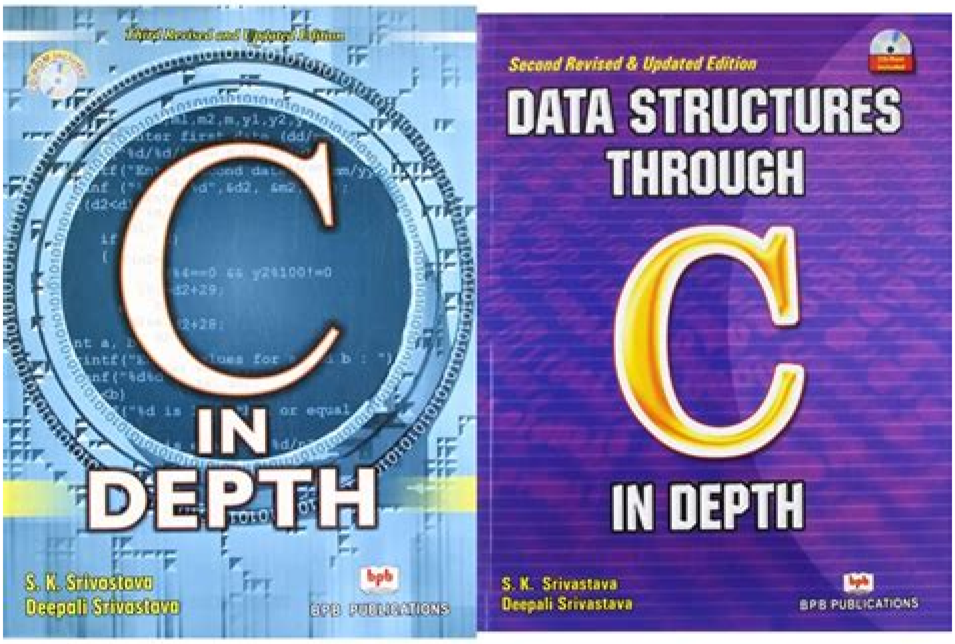
Data Structure Through C In Depth


Third Revised and Updated Edition
C IN DEPTH
S. K. Srivastava
Deepali Srivastava
BPB PUBLICATIONS
Second Revised & Updated Edition
DATA STRUCTURES THROUGH C IN DEPTH
S. K. Srivastava
Deepali Srivastava
BPB PUBLICATIONS

Data structure through C in depth is a fundamental aspect of computer science that enables programmers to efficiently manage and manipulate data. Understanding data structures is essential for writing efficient algorithms and improving the performance of applications. This article will provide an in-depth exploration of various data structures implemented in C, discussing their characteristics, advantages, disadvantages, and practical applications.

Understanding Data Structures

Data structures are specialized formats for organizing, processing, and storing data. They enable efficient access and modification of data, which is crucial for creating scalable and high-performance applications. In C, data structures can be categorized into two types: primitive and non-primitive.

Primitive Data Structures

Primitive data structures are the basic building blocks of data manipulation in C. They include:

1. Integers - Stores whole numbers.
2. Floats - Stores decimal numbers.
3. Characters - Represents single characters.
4. Double - Stores double-precision floating-point numbers.

These data types directly represent data and have a fixed size.

Non-Primitive Data Structures

Non-primitive data structures are more complex and can be classified into two main categories: linear and non-linear.

1. Linear Data Structures: These data structures store data in a sequential manner.
- Arrays: A collection of elements stored at contiguous memory locations.
- Example: `int arr[10];` defines an array of 10 integers.
- Linked Lists: A collection of nodes, where each node contains data and a reference (or link) to the next node.
- Stacks: A collection of elements that follows the Last In First Out (LIFO) principle.
- Queues: A collection of elements that follows the First In First Out (FIFO) principle.

2. Non-Linear Data Structures: These data structures store data in a hierarchical manner.
- Trees: A hierarchical structure consisting of nodes, where each node has a value and references to child nodes.
- Graphs: A collection of nodes (vertices) connected by edges, representing relationships between data elements.

Implementing Data Structures in C

The C programming language provides various ways to implement and manage data structures. Below are examples of how to implement some common data structures.

Arrays

Arrays are one of the simplest data structures. They are used to store fixed-size sequential collections of elements of the same type.

```c
include
```

```c
int main() {
int arr[5] = {1, 2, 3, 4, 5};
for (int i = 0; i < 5; i++) {
printf("%d ", arr[i]);
}
return 0;
}
```

Advantages of Arrays:
- Fast access to elements via indexing.
- Memory efficiency as they store elements in contiguous memory locations.

Disadvantages of Arrays:
- Fixed size; cannot be resized once defined.
- Insertion and deletion of elements can be costly.

Linked Lists

A linked list consists of nodes where each node contains data and a pointer to the next node. This structure allows for dynamic memory allocation.

```c
include
include

struct Node {
int data;
struct Node next;
};

void printList(struct Node n) {
while (n != NULL) {
printf("%d ", n->data);
n = n->next;
}
}

int main() {
struct Node head = (struct Node)malloc(sizeof(struct Node));
head->data = 1;
head->next = NULL;

struct Node second = (struct Node)malloc(sizeof(struct Node));
second->data = 2;
second->next = NULL;

head->next = second; // Link first node with second
printList(head);
return 0;
}
```

Advantages of Linked Lists:
- Dynamic size; can grow or shrink as needed.
- Efficient insertion and deletion of elements.

Disadvantages of Linked Lists:
- Extra memory space for pointers.
- Sequential access; cannot access elements directly like arrays.

Stacks

A stack is a linear data structure that follows the LIFO principle. It can be implemented using arrays or linked lists.

```c
include
include

define MAX 10

struct Stack {
int top;
int items[MAX];
};

void initStack(struct Stack s) {
s->top = -1;
}

int isFull(struct Stack s) {
return s->top == MAX - 1;
}

int isEmpty(struct Stack s) {
return s->top == -1;
}

void push(struct Stack s, int item) {
if (isFull(s)) {
printf("Stack is full\n");
} else {
s->items[++(s->top)] = item;
}
}
```

```
int pop(struct Stack s) {
if (isEmpty(s)) {
printf("Stack is empty\n");
return -1;
} else {
return s->items[(s->top)--];
}
}

int main() {
struct Stack stack;
initStack(&stack);
push(&stack, 1);
push(&stack, 2);
printf("%d popped from stack\n", pop(&stack));
return 0;
}
```

Advantages of Stacks:
- Simple implementation.
- Useful in recursive programming and expression evaluation.

Disadvantages of Stacks:
- Size limitation (in array implementation).
- Difficult to access elements other than the top one.

Queues

A queue is a linear data structure that follows the FIFO principle. It can also be implemented using arrays or linked lists.

```c
include
include

define MAX 10

struct Queue {
int items[MAX];
int front, rear;
};

void initQueue(struct Queue q) {
q->front = -1;
q->rear = -1;
}

int isFull(struct Queue q) {
```

```
return q->rear == MAX - 1;
}

int isEmpty(struct Queue q) {
return q->front == -1 || q->front > q->rear;
}

void enqueue(struct Queue q, int item) {
if (isFull(q)) {
printf("Queue is full\n");
} else {
if (q->front == -1) q->front = 0;
q->items[++(q->rear)] = item;
}
}

int dequeue(struct Queue q) {
if (isEmpty(q)) {
printf("Queue is empty\n");
return -1;
} else {
return q->items[(q->front)++];
}
}

int main() {
struct Queue queue;
initQueue(&queue);
enqueue(&queue, 1);
enqueue(&queue, 2);
printf("%d dequeued from queue\n", dequeue(&queue));
return 0;
}
```

Advantages of Queues:
- Useful for scheduling and managing tasks.
- Simple implementation.

Disadvantages of Queues:
- Size limitation (in array implementation).
- Accessing elements other than the front one can be inefficient.

Advanced Data Structures

In addition to basic data structures, C also allows the implementation of more advanced structures like trees and graphs.

Trees

A tree is a non-linear data structure that consists of nodes connected by edges. Each tree has a root node and sub-nodes.

- Binary Trees: Each node has at most two children.
- Binary Search Trees (BST): A binary tree where the left child is less than the parent and the right child is greater.

```c
include
include

struct TreeNode {
int data;
struct TreeNode left;
struct TreeNode right;
};

struct TreeNode newNode(int data) {
struct TreeNode node = (struct TreeNode)malloc(sizeof(struct TreeNode));
node->data = data;
node->left = node->right = NULL;
return node;
}

void inorder(struct TreeNode root) {
if (root != NULL) {
inorder(root->left);
printf("%d ", root->data);
inorder(root->right);
}
}

int main() {
struct TreeNode root = newNode(1);
root->left = newNode(2);
root->right = newNode(3);
inorder(root);
return 0;
}
```

Advantages of Trees:
- Hierarchical data representation.
- Efficient searching, inserting, and deleting.

Disadvantages of Trees:
- Complexity in implementation.
- Unbalanced trees can lead to inefficiencies.

Graphs

Graphs are a collection of nodes connected by edges. They can be directed or undirected.

- Adjacency Matrix: A 2D array used to represent a graph.
- Adjacency List: A list where each node has a list of its adjacent nodes.

```c
include
include

struct Graph {
int vertices;
int adjMatrix;
};

struct Graph createGraph(int vertices) {
struct Graph graph = (struct Graph)malloc(sizeof(struct Graph));
graph->vertices = vertices;
```

Frequently Asked Questions

What are the fundamental data structures covered in 'Data Structures through C in Depth'?

The fundamental data structures covered include arrays, linked lists, stacks, queues, trees, graphs, and hash tables.

How does 'Data Structures through C in Depth' explain the concept of pointers?

The book explains pointers as variables that store memory addresses, emphasizing their role in dynamic memory allocation and the manipulation of data structures.

What is the importance of time complexity analysis in data structures as discussed in the book?

Time complexity analysis is crucial for evaluating the efficiency of algorithms associated with data structures, helping developers choose the most suitable structure for their needs.

Can you explain how the book approaches the

implementation of linked lists in C?

The book provides a step-by-step guide for implementing singly and doubly linked lists, complete with code examples and explanations of operations like insertion, deletion, and traversal.

What are some real-world applications of data structures covered in 'Data Structures through C in Depth'?

Real-world applications include managing databases (using trees and hash tables), implementing caches (using stacks and queues), and network routing algorithms (using graphs).

How does the book address the concept of recursion in relation to data structures?

The book discusses recursion as a critical technique for traversing complex data structures like trees and graphs, providing examples and code snippets to illustrate its use.

What role do algorithms play in the understanding of data structures in this book?

Algorithms are integral to the understanding of data structures, with the book detailing various algorithms for searching, sorting, and manipulating data within these structures.

Find other PDF article:
https://soc.up.edu.ph/19-theme/pdf?docid=dAe60-0070&title=el-nio-del-globo-rojo-historia-real.pdf

Data Structure Through C In Depth

C盘APPData目录如何清理，目前占用了几十**G**？ - 知乎

C盘APPData目录清理方法，解决占用几十G空间问题，防止C盘飘红。

邓白氏码是干什么用的？我要怎么获得？ - 知乎

DUNS编码：(Data Universal Numbering System)是一个独一无二的9位数字全球编码系统，被广泛应用于企业识别、商业信息的组织及整理。目前不仅美国FDA强制要求受其监管的企业必须 …

手机微信接收的文件存储在哪？ - 知乎

微信8.0版本文档、图片、视频的保存目录还真的改了。 1、接收文件保存目录：Android\Data\com.tencent.mm\MicroMsg\Download 2、点击保存的图片和自己发送的图片保 …

带你一分钟了解编码器基础知识 - 知乎

Mar 8, 2024 · 2.绝对值型编码器 绝对值编码器的输出可直接反映360°范围内的绝对角度，就是对应一圈，每个基准的角度发出一个唯一与该角度对应二进制的数值，通过外部记圈器件可以进行 …

DATAファイルの開き方 -国税庁の**HP**から確定申告を保存しまし …

Feb 20, 2017 · 国税庁のHPから確定申告を保存しましたが拡張子がDATAになっています。これの開き方を誰か教えてください。同じ国税庁のHPから確定申告の作成に進んで「再開する …

C盘里的**Appdata**有哪些是可以删的？ - 知乎

Appdata里是不能断定什么东西“一定可以删”的，但是可以靠经验来人工判断。 Local Local文件夹是常规的本地应用程序数据保存位置，也是最容易出现垃圾的地方。里边大部分的文件夹所属 …

电脑NVIDIA的文件夹有那些是缓存文件可以删除？ - 知乎

可以删除的文件位置具体一点C:\ProgramData\ NVIDIA Corporation \NetService 该目录下是设置自动更新NVIDIA后下载的完整驱动包残留 C:\Program Files\NVIDIA Corporation\Installer2 该目 …

微信新版本聊天记录文件夹改为**xwechat_file**了，老的聊天记 …

今天为这个弄了一下午，还是没搞定。正在经历你所说的问题。差不多200G的记录，真是要了命了。现在为了不影响手头工作，只能还继续用原来的电脑挂着微信，新电脑没办法完全交接老 …

发*SCI*让加数据可用性声明怎么弄？ - 知乎

Dec 3, 2019 · The data that support the findings of this study are available from the corresponding author, [author initials], upon reasonable request. 4. 当稿件中所用的数据来自公共领域资源 …

如何知道一个期刊是不是**sci**？ - 知乎

这里给出一个图文并茂的方法，手把手的教你，核查期刊质量，快速判断期刊是否被SCI收录，国内外两种方案都介绍，保证查得到！ 循证医学杂谈：期刊质量评价·实战篇 在上一篇文章（循 …

C盘**APPData**目录如何清理，目前占用了几十**G**？ - 知乎

C盘APPData目录清理方法，解决占用几十G空间问题，防止C盘飘红。

邓白氏码是干什么用的？我要怎么获得？ - 知乎

DUNS编码：(Data Universal Numbering System)是一个独一无二的9位数字全球编码系统，被广泛应用于企业识别、商业信息的组织及整理。目前不仅美国FDA强制要求受其监管的企业必须 …

手机微信接收的文件存储在哪？ - 知乎

微信8.0版本文档、图片、视频的保存目录还真的改了。 1、接收文件保存目录：Android\Data\com.tencent.mm\MicroMsg\Download 2、点击保存的图片和自己发送的图片保 …

带你一分钟了解编码器基础知识 - 知乎

Mar 8, 2024 · 2.绝对值型编码器 绝对值编码器的输出可直接反映360°范围内的绝对角度，就是对应一圈，每个基准的角度发出一个唯一与该角度对应二进制的数值，通过外部记圈器件可以进行 …

DATAファイルの開き方 -国税庁の**HP**から確定申告を保存しまし …

Feb 20, 2017 · 国税庁のHPから確定申告を保存しましたが拡張子がDATAになっています。これの開き方を誰か教えてください。同じ国税庁のHPから確定申告の作成に進んで「再開する …

*C*盘里的*Appdata*有哪些是可以删的？ - 知乎

Appdata里是不能断定什么东西“一定可以删”的，但是可以靠经验来人工判断。 Local Local文件夹是常规的本地应用程序数据保存位置，也是最容易出现垃圾的地方。里边大部分的文件夹所属 …

电脑*NVIDIA*的文件夹有那些是缓存文件可以删除？ - 知乎

可以删除的文件位置具体一点C:\ProgramData\ NVIDIA Corporation \NetService 该目录下是设置自动更新NVIDIA后下载的完整驱动包残留C:\Program Files\NVIDIA Corporation\Installer2 该目 …

微信新版本聊天记录文件夹改为**xwechat_file**了，老的聊天记 **…**

今天为这个弄了一下午，还是没搞定。正在经历你所说的问题。差不多200G的记录，真是要了命了。现在为了不影响手头工作，只能还继续用原来的电脑挂着微信，新电脑没办法完全交接 …

发**SCI**让加数据可用性声明怎么弄？ **-** 知乎

Dec 3, 2019 · The data that support the findings of this study are available from the corresponding author, [author initials], upon reasonable request. 4. 当稿件中所用的数据来自公共领域资源 …

如何知道一个期刊是不是**sci**？ **-** 知乎

这里给出一个图文并茂的方法，手把手的教你，核查期刊质量，快速判断期刊是否被SCI收录，国内外两种方案都介绍，保证查得到！ 循证医学杂谈：期刊质量评价·实战篇 在上一篇文章（循 …

Master data structures through C in depth with our comprehensive guide. Discover how to implement and optimize key structures for better programming skills. Learn more!

Back to Home