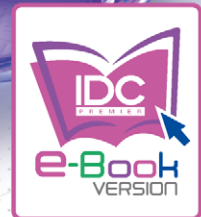
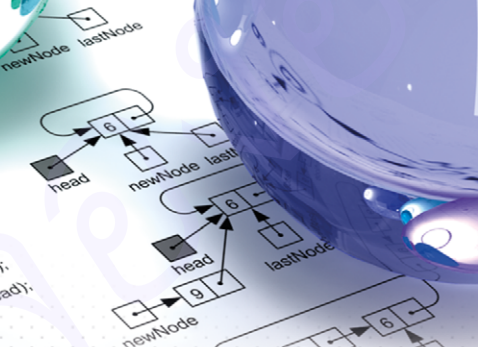




```

Node head = new Node(1);
struct Node *head =
[Java] Node newNode = new Node(1);
Node lastNode = newNode;
struct Node *newNode = NULL;
[c] newNode = insertNode(6, newNode);
struct Node *lastNode = newNode;
[Java] head = newNode;
last.setNext(newNode);
head = newNode;
last->next = newNode;
node = insertNode(9, head);

```



คู่มือเรียน โครงสร้างข้อมูล



อัลกอริทึม

Data Structure & Algorithm



อ่านเข้าใจง่าย
อธิบายอย่างละเอียด เป็นขั้นตอน
มีตัวอย่างประกอบทุกหัวข้อ

เรียนรู้การทำงาน
ด้วยการเขียนโปรแกรม
ภาษา C และภาษา Java

เหมาะสำหรับ
นักเรียน นักศึกษา
และผู้ที่สนใจ

Contents

บทที่ 1 รู้จักกับโครงสร้างข้อมูลและอัลกอริทึม 1

โครงสร้างข้อมูลและอัลกอริทึมคืออะไร.....	1
ประโยชน์ของโครงสร้างข้อมูลและอัลกอริทึม	2
ผังงาน (Flowchart)	3
โค้ดรหัสเทียม (Pseudo code).....	5
Abstract Data Type	6
ประเภทของอัลกอริทึม	8
สรุปเนื้อหาบทที่ 1.....	10
แบบฝึกหัดท้ายบทที่ 1	10

บทที่ 2 การวิเคราะห์ประสิทธิภาพของอัลกอริทึม (Performance Analysis) ... 13

คณิตศาสตร์พื้นฐานสำหรับการวิเคราะห์ประสิทธิภาพของอัลกอริทึม	13
การวัดประสิทธิภาพอัลกอริทึม.....	14
การวิเคราะห์หน่วยความจำที่ใช้ในการประมวลผล (Space Complexity Analysis).....	14
การวิเคราะห์เวลาที่ใช้ในการประมวลผล (Time Complexity Analysis).....	15
อัตราการเติบโตของอัลกอริทึม (Algorithm Growth Rates)	17
อัตราการเติบโต Big-O (O).....	17
อัตราการเติบโต Big-Omega (Ω).....	18
อัตราการเติบโต Big-Theta (Θ).....	19
อัตราการเติบโต Little-o (o).....	20
อัตราการเติบโต Little-omega (ω).....	20
เปรียบเทียบอัตราการเติบโตของอัลกอริทึม.....	21
การนับตัวดำเนินการ (Operation Counts).....	21
นับตัวดำเนินการแบบค่าคงที่ (Constant)	21
นับตัวดำเนินการแบบลูปลำดับ (Linear loops).....	22
นับตัวดำเนินการแบบลูปลอการิทึม (Logarithmic loops)	24
นับตัวดำเนินการแบบลูปล้อน (Nested loops).....	26
ฟังก์ชันอัตราการเติบโตตามการวัดประสิทธิภาพของอัลกอริทึม.....	29
การวิเคราะห์ Best-case, Worst-case และ Average-case	30
สรุปเนื้อหาบทที่ 2.....	30
แบบฝึกหัดท้ายบทที่ 2	31

บทที่ 3 อาร์เรย์ (Array)..... 33

รู้จักกับอาร์เรย์ (Array).....	33
อาร์เรย์ 1 มิติ.....	34
ประกาศอาร์เรย์ 1 มิติแบบมีขนาดเท่ากับจำนวนข้อมูลที่ต้องการจัดเก็บ	34
ประกาศอาร์เรย์ 1 มิติแบบกำหนดขนาดอาร์เรย์	35

อาร์เรย์หลายมิติ.....	37
การอ้างอิงข้อมูลในหน่วยความจำ.....	39
การอ้างอิงข้อมูลในหน่วยความจำในภาษา Java.....	39
การอ้างอิงข้อมูลในหน่วยความจำภาษา C.....	41
สรุปเนื้อหาบทที่ 3.....	42
แบบฝึกหัดท้ายบทที่ 3.....	42

บทที่ 4 ลิงค์ลิสต์ (Link-list)..... 45

ลิงค์ลิสต์ทิศทางเดียว (Singly Link-List).....	45
การสร้างและใช้งานลิงค์ลิสต์ทิศทางเดียวในภาษา Java.....	46
การสร้างและใช้งานลิงค์ลิสต์ทิศทางเดียวในภาษา C.....	48
การจัดการลิงค์ลิสต์ทิศทางเดียว.....	50
การสร้างส่วนหัวของลิงค์ลิสต์ทิศทางเดียว.....	50
การค้นหาตำแหน่งโหนดที่ต้องการลบ หรือแทรกโหนดในลิงค์ลิสต์ทิศทางเดียว.....	51
การลบโหนดในลิงค์ลิสต์ทิศทางเดียว.....	53
การเพิ่มโหนดในลิงค์ลิสต์ทิศทางเดียว.....	55
การนำข้อมูลในลิงค์ลิสต์ทิศทางเดียวออกมาแสดงผล.....	58
การเปลี่ยนโครงสร้างลิงค์ลิสต์ทิศทางเดียว.....	58
การอ้างอิงส่วนท้าย (Tail References).....	58
ดัมมี่โหนด (Dummy Node).....	59
ลิงค์ลิสต์แบบวงกลม (Circular Linked-List).....	59
ตัวอย่างการสร้างลิงค์ลิสต์ทิศทางเดียว.....	61
ลิงค์ลิสต์แบบสองทิศทาง (Doubly Link-List).....	65
โครงสร้างลิงค์ลิสต์แบบสองทิศทางในภาษา Java.....	66
โครงสร้างลิงค์ลิสต์แบบสองทิศทางในภาษา C.....	68
การจัดการลิงค์ลิสต์แบบสองทิศทาง.....	69
การค้นหาตำแหน่งโหนดในลิงค์ลิสต์แบบสองทิศทาง.....	69
การลบโหนดในลิงค์ลิสต์แบบสองทิศทาง.....	70
การเพิ่มโหนดในลิงค์ลิสต์แบบสองทิศทาง.....	72
ตัวอย่างการสร้างลิงค์ลิสต์แบบสองทิศทาง.....	75
การนำลิงค์ลิสต์ไปใช้งาน.....	80
สรุปเนื้อหาบทที่ 4.....	82
แบบฝึกหัดท้ายบทที่ 4.....	82

บทที่ 5 สแตก (Stack)..... 85

ตัวอย่างการนำหลักการของสแตกไปใช้งาน (Simple Application of the Stack).....	86
เครื่องมือที่ใช้ในการสร้างสแตก.....	87
การสร้างสแตกด้วยโครงสร้างอาร์เรย์.....	88
สร้างสแตกด้วยโครงสร้างอาร์เรย์ในภาษา Java.....	88
สร้างสแตกด้วยโครงสร้างอาร์เรย์ในภาษา C.....	90
การสร้างสแตกด้วยโครงสร้างลิงค์ลิสต์.....	93
สร้างสแตกด้วยโครงสร้างลิงค์ลิสต์ในภาษา Java.....	93
สร้างสแตกด้วยโครงสร้างลิงค์ลิสต์ในภาษา C.....	95

การนำสแตกไปใช้งาน	97
การเปลี่ยนรูปแบบ infix ให้เป็น postfix	97
การคำนวณทางคณิตศาสตร์จากรูปแบบของ Postfix	100
สรุปเนื้อหาบทที่ 5	103
แบบฝึกหัดท้ายบทที่ 5	104

บทที่ 6 คิว (Queue)..... 105

การสร้างคิวด้วยโครงสร้างลิงค์ลิสต์	107
การสร้างคิวด้วยโครงสร้างลิงค์ลิสต์แบบทิศทางเดียว	107
การสร้างคิวด้วยโครงสร้างลิงค์ลิสต์แบบวงกลม	107
การเพิ่มข้อมูลในคิวโครงสร้างลิงค์ลิสต์แบบวงกลม	108
การเพิ่มข้อมูลในคิวในกรณีที่คิวไม่มีข้อมูล	108
การเพิ่มข้อมูลในคิวในกรณีที่มีข้อมูล	108
การนำข้อมูลออกจากคิวโครงสร้างลิงค์ลิสต์แบบวงกลม	110
อัลกอริทึมจัดการคิวโครงสร้างลิงค์ลิสต์แบบวงกลม	110
การจัดการคิวโครงสร้างลิงค์ลิสต์แบบวงกลมด้วยภาษา Java	110
การจัดการคิวโครงสร้างลิงค์ลิสต์แบบวงกลมด้วยภาษา C	113
การสร้างคิวด้วยโครงสร้างอาร์เรย์	114
การค้นหาค่าตำแหน่งในการเพิ่มข้อมูลในคิวอาร์เรย์แบบวงกลม	116
การค้นหาค่าตำแหน่งในการนำข้อมูลออกจากคิวอาร์เรย์แบบวงกลม	116
การตรวจสอบคิวอาร์เรย์แบบวงกลมว่าคิวเต็มหรือคิวว่างเปล่า	117
อัลกอริทึมจัดการคิวโครงสร้างอาร์เรย์แบบวงกลม	117
การจัดการคิวโครงสร้างอาร์เรย์แบบวงกลมด้วยภาษา Java	117
การจัดการคิวโครงสร้างอาร์เรย์แบบวงกลมด้วยภาษา C	119
การนำคิวไปใช้งาน	120
สรุปเนื้อหาบทที่ 6	121
แบบฝึกหัดท้ายบทที่ 6	121

บทที่ 7 นรี (Tree) 123

รู้จักกับทรี (Tree)	123
คุณสมบัติเฉพาะของทรี (Terminology of Tree)	124
รู้จักกับไบนารีทรี (Binary Tree)	125
คุณสมบัติของไบนารีทรี	125
การสร้างและการจัดการไบนารีทรี	126
การสร้างไบนารีทรีด้วยโครงสร้างอาร์เรย์	128
การสร้างไบนารีทรีด้วยโครงสร้างลิงค์ลิสต์	130
การจัดการข้อมูลในไบนารีทรี	130
การค้นหาข้อมูลในไบนารีทรี	131
การเพิ่มโหนดข้อมูลในไบนารีทรี	132
การท่องเข้าไปในไบนารีทรี	133
การลบโหนดข้อมูลในไบนารีทรี	142
รู้จักกับ AVL Tree	144
การสร้าง AVL Tree	146

การเพิ่มข้อมูลใน AVL Tree	146
การปรับไบนารีทรีให้สมดุลด้วยการหมุน 1 ครั้งใน AVL Tree	146
การปรับไบนารีทรีให้สมดุลด้วยการหมุน 2 ครั้งใน AVL Tree	147
การลบโหนดใน AVL Tree	150
การปรับไบนารีทรีให้สมดุลด้วยการหมุน 1 ครั้ง หลังจากลบโหนดใน AVL Tree	151
การปรับไบนารีทรีให้สมดุลด้วยการหมุน 2 ครั้ง หลังจากลบโหนดใน AVL Tree	151
รู้จักกับทรีสมดุลแบบ 2-3 Trees	153
กฎของ 2-3 Trees	154
โครงสร้างข้อมูล 2-3 Trees	155
การท่องเข้าไปใน 2-3 Trees	155
การค้นหาข้อมูลใน 2-3 Trees	156
การเพิ่มข้อมูลใน 2-3 Trees	157
สรุปขั้นตอนการเพิ่มข้อมูลในตำแหน่งโหนดใบของ 2-3 Trees	159
สรุปขั้นตอนการเพิ่มข้อมูลในตำแหน่งโหนดแม่ของ 2-3 Trees	160
สรุปขั้นตอนการเพิ่มข้อมูลในตำแหน่งโหนดรากของ 2-3 Trees	160
การลบข้อมูลใน 2-3 Trees	162
สรุปขั้นตอนวิธีในการลบข้อมูลใน 2-3 Trees	164
รู้จักกับทรีสมดุลแบบ 2-3-4 Trees	167
กฎของ 2-3-4 Trees	168
โครงสร้างข้อมูล 2-3-4 Trees	169
การเพิ่มข้อมูลใน 2-3-4 Trees	170
สรุปการเพิ่มข้อมูลใน 2-3-4 Trees	172
การลบข้อมูลใน 2-3-4 Trees	172
รู้จักกับ red-black Tree	173
การค้นหาและการท่องเข้าไปใน red-black Tree	174
การเพิ่มโหนดใน red-black Tree	175
การปรับโครงสร้างโหนดที่เป็นสีแดงทั้งคู่	175
การลบโหนดใน red-black Tree	178
รู้จักกับ B-Tree	181
การเพิ่มโหนดใน B-Tree	182
การลบโหนดใน B-Tree	185
การวิเคราะห์ B-Tree	187
สรุปเนื้อหาบทที่ 7	187
แบบฝึกหัดท้ายบทที่ 7	188

บทที่ 8 แฮช (Hash) 191

แฮชฟังก์ชัน (Hash functions)	192
การเลือกหลัก (Selection digits)	192
การบวกหลัก (Folding)	193
การหารเอาเศษ (Modulate arithmetic)	193
การเปลี่ยนข้อความเป็นตัวเลข (Converting a character string to an integer)	193
การแก้ปัญหาการชนกันของแฮชคีย์ (Resolving Collision)	194
แก้ปัญหาการชนกันด้วยการหาแฮชถัดไปที่ใกล้เคียงที่สุด	195

การแก้ปัญหาการชนกันด้วยวิธีการปรับโครงสร้างตารางแฮช	197
สรุปเนื้อหาบทที่ 8	199
แบบฝึกหัดท้ายบทที่ 8	199
บทที่ 9 ไนส์ (Tries).....	201
Simple Tries	202
Full Tries	205
Compressed Tries.....	207
สรุปเนื้อหาบทที่ 9	209
แบบฝึกหัดท้ายบทที่ 9	209
บทที่ 10 ลำดับความสำคัญของคิวและฮีพ (Priority Queue and Heap)...	211
ลำดับความสำคัญของคิว (Priority Queue)	211
ค่าของลำดับความสำคัญ (Priority value).....	211
เครื่องมือที่ใช้สร้างคิวของลำดับความสำคัญ	212
โครงสร้างอาร์เรย์	212
โครงสร้างลิงค์ลิสต์	213
โครงสร้างโบนารีทรี	213
ฮีพ (Heap).....	213
การสร้างข้อมูลในฮีพ	214
การลบคีย์ในฮีพ	214
การเพิ่มคีย์ในฮีพ.....	216
โครงสร้างคลาสฮีพ.....	218
สรุปเนื้อหาบทที่ 10.....	220
แบบฝึกหัดท้ายบทที่ 10.....	220
บทที่ 11 การจัดเรียงและการค้นหาข้อมูล	221
อัลกอริทึมรับข้อมูล	221
การจัดเรียงข้อมูลแบบ Selection sort	223
วิเคราะห์หาประสิทธิภาพการจัดเรียงข้อมูลแบบ Selection sort	225
การจัดเรียงข้อมูลแบบ Bubble sort.....	226
วิเคราะห์หาประสิทธิภาพการจัดเรียงข้อมูลแบบ Bubble sort.....	227
การจัดเรียงข้อมูลแบบ Insertion sort.....	229
วิเคราะห์หาประสิทธิภาพการจัดเรียงข้อมูลแบบ Insertion sort	230
การจัดเรียงข้อมูลแบบ Merge sort	231
วิเคราะห์หาประสิทธิภาพการจัดเรียงข้อมูลแบบ Merge sort.....	234
การจัดเรียงข้อมูลแบบ Quick sort.....	236
วิเคราะห์หาประสิทธิภาพการจัดเรียงข้อมูลแบบ Quick sort.....	238
การจัดเรียงข้อมูลแบบ Radix sort.....	240
วิเคราะห์หาประสิทธิภาพการจัดเรียงข้อมูลแบบ Radix sort.....	242
การจัดเรียงข้อมูลแบบ Heap sort	242
วิเคราะห์หาประสิทธิภาพการจัดเรียงข้อมูลแบบ Heap sort	245
การจัดเรียงข้อมูลแบบ Shell sort.....	246

วิเคราะห์การจัดเรียงข้อมูลแบบ Shell sort.....	248
การค้นหาข้อมูลแบบลำดับ (Sequential search หรือ Linear search).....	250
วิเคราะห์การค้นหาข้อมูลแบบลำดับ	250
การค้นหาข้อมูลด้วย Binary search หรือ Half-Interval search	251
วิเคราะห์การค้นหาข้อมูลแบบไบนารี	253
สรุปเนื้อหาบทที่ 11.....	254
แบบฝึกหัดท้ายบทที่ 11	254

บทที่ 12 กราฟ (Graph)..... 255

โครงสร้างของกราฟ.....	255
การสร้างกราฟ	258
เมตริกซ์ประชิด	258
รายการประชิด	259
การท่องเที่ยวในกราฟ (Graph Traversals).....	259
Depth-first Search.....	260
Breadth-first Search.....	261
การนำกราฟไปใช้งาน.....	262
Topological sorting	263
Possible Spanning Tree.....	264
DFS Spanning Tree.....	265
BFS Spanning Tree.....	266
Minimum Spanning Tree	266
Shortest Paths	268
Kruskal's Algorithm.....	271
Dijkstra's Algorithm	272
สรุปเนื้อหาบทที่ 12.....	276
แบบฝึกหัดท้ายบทที่ 12	276

รู้จักกับโครงสร้างข้อมูลและอัลกอริทึม

หนังสือเล่มนี้เป็นหนังสือที่สอนในเรื่องโครงสร้างข้อมูลและอัลกอริทึม มุ่งเน้นในเรื่องวิธีการและขั้นตอนการแก้ไขปัญหาต่างๆ โดยใช้คุณสมบัติและความสามารถของโครงสร้างข้อมูล ซึ่งถือได้ว่าเป็นรายวิชาหลักในการเรียนในสาขาคอมพิวเตอร์ ดังนั้น ผู้อ่านควรศึกษาเนื้อหาและตัวอย่างต่างๆ ในหนังสือให้เข้าใจ เพื่อใช้เป็นพื้นฐานในการพัฒนาโปรแกรมต่อไป

โครงสร้างข้อมูลและอัลกอริทึมคืออะไร

โครงสร้างข้อมูล (Data Structure) คือ การจัดการข้อมูลในหน่วยความจำภายในเครื่องคอมพิวเตอร์ หรือในบางครั้งเป็นการจัดการข้อมูลในดิสก์ ให้ความสัมพันธ์กันภายในกลุ่มข้อมูลให้มีรูปแบบและข้อกำหนดที่ชัดเจนในการกำหนดคุณสมบัติเพื่อสร้างความสัมพันธ์ภายในกลุ่มข้อมูล รูปแบบการเก็บข้อมูลที่อยู่ในรูปแบบโครงสร้างข้อมูล เช่น อาร์เรย์ (Array), ลิงค์ลิสต์ (Link-list), สแตก (Stack), ไบนารีทรี (Binary tree) เป็นต้น

อัลกอริทึม (Algorithm) หรือเรียกอีกอย่างว่า **ขั้นตอนวิธี** เป็นวิธีการแสดงลำดับขั้นตอนในการทำงานหรือการแก้ไขปัญหาอย่างใดอย่างหนึ่ง เช่น การกำหนดขั้นตอนเพื่อแก้ไขปัญหาการจัดเรียงเอกสารในแฟ้มข้อมูล หรือการกำหนดอัลกอริทึมในการค้นหาข้อมูลในแฟ้มข้อมูลทั้งหมด เป็นต้น

อัลกอริทึมหรือขั้นตอนวิธีเป็นสิ่งที่พบได้ในชีวิตประจำวัน เช่น ขั้นตอนการเติมเงินในโทรศัพท์มือถือผ่านตู้เติมเงินอัตโนมัติ, ขั้นตอนการใช้งานตู้กดเงินอัตโนมัติ (ATM) เพื่อทำธุรกรรมทางการเงิน เป็นต้น

ตัวอย่างที่ 1.1 แสดงอัลกอริทึมหรือขั้นตอนวิธีการใช้ตู้กดเงินอัตโนมัติ (ATM) เพื่อโอนเงินดังนี้

1. ใส่บัตร ATM
2. บอกรหัสผ่านของบัตร ATM
3. ในหน้าบริการ เลือกบริการรายการโอนเงิน
4. เลือกรูปแบบการโอนเงินว่าจะโอนเงินเข้าบัญชีอื่นธนาคารเดียวกัน หรือธนาคารอื่นๆ
5. กดหมายเลขบัญชีที่ต้องการโอนเงินเข้าบัญชี
6. ตรวจสอบเลขที่บัญชีที่ป้อนถูกต้องหรือไม่ ถ้าถูกต้องให้กดตกลง
7. กรอกจำนวนเงินที่ต้องการโอนเงิน แล้วกดตกลง
8. ชื่อบัญชีและจำนวนเงินที่ต้องการโอนจะปรากฏขึ้นเพื่อยืนยันความถูกต้องในการโอนเงิน เมื่อตรวจสอบบัญชีและจำนวนเงินถูกต้อง ให้กดตกลง เป็นการเสร็จสิ้นการโอนเงิน
9. รับบัตร ATM
10. รับใบสลิปการโอนเงิน

ตัวอย่างที่ 1.2 แสดงอัลกอริทึมการค้นหาข้อมูลในอาร์เรย์ขนาด n ข้อมูลดังนี้

1. รับข้อมูลตัวเลขที่ต้องการค้นหา
2. เปรียบเทียบข้อมูลในอาร์เรย์ทีละตัวตั้งแต่ข้อมูลในตำแหน่งที่ 0 จนถึงตำแหน่งที่ $n-1$
3. ถ้าข้อมูลในอาร์เรย์ตรงกับข้อมูลที่ต้องการค้นหา แสดงว่าเจอข้อมูล และจบการค้นหาข้อมูล
4. ถ้าเปรียบเทียบข้อมูลจนถึงตำแหน่งที่ $n-1$ แล้วไม่พบข้อมูลตัวใดในอาร์เรย์เลย แสดงว่าไม่มีข้อมูลที่ต้องการค้นหาในอาร์เรย์

ประโยชน์ของโครงสร้างข้อมูลและอัลกอริทึม

โครงสร้างข้อมูลและอัลกอริทึม เป็นวิธีการจัดการกับข้อมูลที่ทำให้คอมพิวเตอร์สามารถจัดการกับข้อมูลเพื่อนำมาใช้งานได้อย่างมีประสิทธิภาพ ตัวอย่างเช่น

- การจัดเก็บข้อมูลจำนวนมากในรูปแบบของโครงสร้างข้อมูลที่เหมาะสม จะทำให้สามารถนำข้อมูลไปใช้ได้อย่างมีประสิทธิภาพ โดยใช้ทรัพยากรที่มีได้อย่างคุ้มค่าที่สุด (ใช้พื้นที่หน่วยความจำน้อยที่สุด) อีกทั้งยังใช้เวลาในการประมวลผลน้อยที่สุดด้วย ส่งผลให้คอมพิวเตอร์ทำงานเร็วขึ้นนั่นเอง
- การอ่านข้อมูลเพื่อใช้ในการประมวลผล คอมพิวเตอร์จะอ่านข้อมูลมาเก็บในหน่วยความจำหลัก (หน่วยความจำแรม) และในกรณีที่ข้อมูลมีขนาดมากกว่าขนาดของหน่วยความจำ คอมพิวเตอร์จะใช้โครงสร้างข้อมูลและอัลกอริทึมเข้ามาช่วยในการจัดการกับข้อมูลเหล่านั้น เพื่อให้สามารถจัดสรรหน่วยความจำได้อย่างเหมาะสมและสามารถทำงานได้อย่างต่อเนื่อง
- การพัฒนาโปรแกรมโดยใช้โครงสร้างข้อมูลและอัลกอริทึมที่เหมาะสม สามารถเพิ่มประสิทธิภาพการทำงานของโปรแกรมได้เช่นกัน (ใช้หน่วยความจำน้อยและประมวลผลได้รวดเร็ว)

ทั้งนี้ประโยชน์ของโครงสร้างข้อมูลและอัลกอริทึมที่กล่าวมาเป็นเพียงตัวอย่างเล็กน้อยเท่านั้น ซึ่งโครงสร้างข้อมูลและอัลกอริทึมยังสามารถประยุกต์เพื่อใช้งานได้อีกมากมาย ไม่ว่าจะเป็นการเรียงลำดับข้อมูล การค้นหาข้อมูล เป็นต้น

ผังงาน (Flowchart)

ผังงาน หรือเรียกว่า **โฟลว์ชาร์ต (Flowchart)** เป็นเครื่องมือที่ใช้ออกแบบระบบงานด้วยสัญลักษณ์ ช่วยให้โครงสร้างของระบบงานที่เป็นลำดับขั้นตอนและเข้าใจได้ง่าย

การนำผังงานมาใช้ในการออกแบบโปรแกรม สามารถตรวจสอบได้ว่ามีลำดับขั้นตอนการทำงานถูกต้องหรือไม่ และสามารถเปลี่ยนแปลงแก้ไขข้อผิดพลาดของระบบงานภายในผังงานได้ง่ายกว่าการหาข้อผิดพลาดที่เกิดจากการเขียนโปรแกรม อีกทั้งยังช่วยลดความสับสนในการพัฒนาโปรแกรมอีกด้วย

วัตถุประสงค์ที่สำคัญที่สุดในการนำผังงานมาเป็นเครื่องมือในการพัฒนาโปรแกรมคือ เพื่อให้บุคคลอื่นสามารถทำความเข้าใจถึงลำดับขั้นตอนการทำงานของโปรแกรม และผลลัพธ์ที่ได้จากการทำงานของโปรแกรมที่พัฒนาขึ้นได้

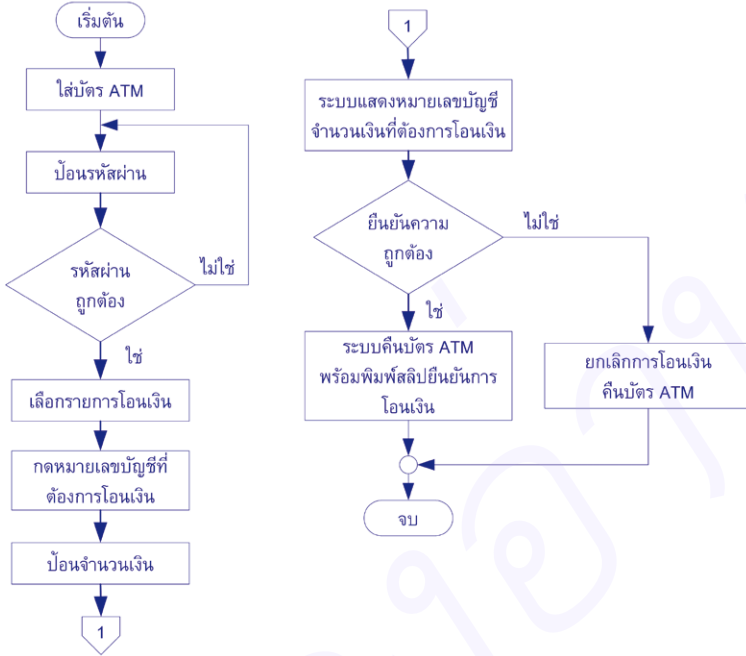
สัญลักษณ์ที่นำมาใช้ในการเขียนผังงานมีอยู่ 8 สัญลักษณ์ดังนี้

สัญลักษณ์	ความหมายสัญลักษณ์ในการใช้งานในผังงาน
 Terminator	จุดเริ่มต้นและจุดสิ้นสุดของโปรแกรม
 Process	การประมวลผลหรือการคำนวณของโปรแกรม
 Data	รับข้อมูลเข้ามาในโปรแกรม หรือส่งค่าออกไปจากโปรแกรม
 Decision	ตรวจสอบเงื่อนไข แล้วเลือกการทำงานของโปรแกรม
 Document	แสดงผลออกทางเอกสาร
 On-page reference	จุดเชื่อมต่อหลายเส้นทางของโปรแกรมให้เหลือการเข้ามาเพียงเส้นทางเดียว
 Off-page reference	ขึ้นหน้าใหม่ในกรณีที่ผังงานมีความยาวเกินกว่าที่จะแสดงพอนในหนึ่งหน้า
	ลูกศรแสดงทิศทางการทำงานของโปรแกรมและการไหลของข้อมูล

ตัวอย่างที่ 1.3 แสดงผังงานขั้นตอนการใช้ตู้กดเงินอัตโนมัติ (ATM) เพื่อโอนเงินดังนี้

รูปที่ 1-1

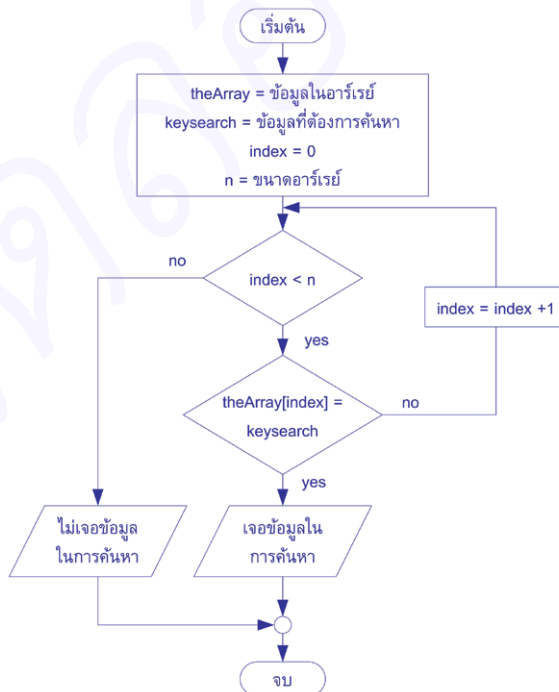
ผังงานขั้นตอนการใช้ตู้กดเงินอัตโนมัติ (ATM) เพื่อโอนเงิน



ตัวอย่างที่ 1.4 แสดงผังงานอัลกอริทึมค้นหาข้อมูลในอาร์เรย์ขนาด n ข้อมูลดังนี้

รูปที่ 1-2

ผังงานอัลกอริทึมค้นหาข้อมูลในอาร์เรย์ขนาด n ข้อมูล



โค้ดรหัสเทียม (Pseudo code)

โค้ดรหัสเทียม (Pseudo code) เป็นโครงสร้างรหัสที่รวมทั้งภาษาเขียนกับภาษาคอมพิวเตอร์เข้าไว้ด้วยกัน เพื่อใช้ในการอธิบายโครงสร้างและลำดับขั้นตอนการทำงานของโปรแกรมโดยไม่อ้างอิงภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่ง เพื่อเป็นสื่อกลางแทนการเขียนด้วยโค้ดโปรแกรม (Code Program) ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1.5 แสดง Pseudo code การค้นหาข้อมูลในอาร์เรย์ดังนี้

```
1 +searching(in theArray:arrayType,in KeySearch:keyType,in MaxData:integer):boolean
2   for (n = 0 to MaxData-1){
3     if (theArray[n] == KeySearch){
4       data is found data then return true
5     }
6   }
7   if (n == MaxData-1){
8     data is not found then return false
9   }
```

อธิบายการทำงานของโปรแกรม

บรรทัดที่ 1 ประการฟังก์ชันหรือเมธอด searching โดยกำหนดให้ตัวแปรอินพุต 3 ตัวคือ

- theArray เป็นข้อมูลทั้งหมดที่เก็บในอาร์เรย์
- KeySearch เป็นข้อมูลที่ต้องการค้นหา
- MaxData เป็นขนาดของอาร์เรย์ที่เก็บข้อมูล

และกำหนดให้ส่งคืนค่าผลการค้นหาข้อมูลเป็นข้อมูลชนิด boolean

บรรทัดที่ 2 วนลูปตั้งแต่ตัวแปร n เท่ากับ 0 จนถึงค่าในตัวแปร MaxData-1

บรรทัดที่ 3-4 ถ้าข้อมูลใน theArray ตำแหน่งที่ n เท่ากับค่าข้อมูล KeySearch แสดงว่าเจอข้อมูลที่ต้องการค้นหา ให้ฟังก์ชันคืนค่าเป็น true

บรรทัดที่ 7-8 ถ้าค่าตัวแปร n เท่ากับ MaxData แสดงว่าไม่พบข้อมูลในอาร์เรย์ให้คืนค่าเป็น false

ตัวอย่างที่ 1.6 แสดง Pseudo code แสดงผลการเรียน (Grade) จากคะแนนที่รับเข้ามาดังนี้

```
1 +showgrade(in score:double)
2   if (score >= 80)
3     output "A"
4   else if (score >= 75)
5     output "B+"
6   else if (score >= 70)
7     output "B"
8   else if (score >= 65)
9     output "C+"
10  else if (score >= 60)
11  output "C"
12  else if (score >= 55)
13  output "D+"
14  else if (score >= 50)
15  output "D"
16  else output "F"
```

อธิบายการทำงานของโปรแกรม

บรรทัดที่ 1	ประกาศฟังก์ชันหรือเมธอด showgrade กำหนดให้รับค่าคะแนนไว้ในตัวแปร
บรรทัดที่ 2-3	ถ้า score มากกว่าเท่ากับ 80 ให้แสดงเกรด “A”
บรรทัดที่ 4-5	แต่ถ้า score มากกว่าเท่ากับ 75 ให้แสดงเกรด “B+”
บรรทัดที่ 6-7	แต่ถ้า score มากกว่าเท่ากับ 70 ให้แสดงเกรด “B”
บรรทัดที่ 8-9	แต่ถ้า score มากกว่าเท่ากับ 65 ให้แสดงเกรด “C+”
บรรทัดที่ 10-11	แต่ถ้า score มากกว่าเท่ากับ 60 ให้แสดงเกรด “C”
บรรทัดที่ 12-13	แต่ถ้า score มากกว่าเท่ากับ 55 ให้แสดงเกรด “D+”
บรรทัดที่ 14-15	แต่ถ้า score มากกว่าเท่ากับ 50 ให้แสดงเกรด “D”
บรรทัดที่ 16	ถ้า score น้อยกว่า 50 ให้แสดงเกรด “F”

Abstract Data Type

Abstract Data Type หรือเรียกว่า **ADT** เป็นการประกาศถึงคุณสมบัติของโครงสร้างข้อมูลและกลุ่มตัวดำเนินการที่กระทำกับโครงสร้างข้อมูล

คุณสมบัติของโครงสร้างข้อมูลที่กำหนดใน ADT เป็นการแสดงถึงลักษณะของโครงสร้างข้อมูลที่น่าสนใจ และกลุ่มตัวดำเนินการจะเป็นฟังก์ชันที่กำหนดการทำงานของโปรแกรมที่กระทำกับโครงสร้างข้อมูล ซึ่งแต่ละกลุ่มตัวดำเนินการจะทำงานอย่างอิสระไม่เกี่ยวเนื่องกัน (ไม่มีการส่งค่าระหว่างกลุ่มตัวดำเนินการ) และมีลักษณะการทำงานที่ชัดเจน

รายการ ADT คือ การรวบรวมลำดับการทำงานที่ชัดเจน โดยใช้ตัวเลขในการอ้างอิงตำแหน่งดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1.7 รายการ ADT แสดงการจัดการรายการสิ่งของในร้านค้าดังนี้

1. Create an empty list. (สร้างรายการว่างเปล่า)
2. Determine whether a list is empty. (สนใจรายการที่ว่างเปล่า)
3. Determine the number of items on a list. (สนใจตัวเลขสิ่งของในรายการ)
4. Add an item at a given position in the list. (เพิ่มสิ่งของในตำแหน่งที่ให้มีในรายการ)
5. Remove the item at a given position in the list. (ลบสิ่งของในตำแหน่งที่ให้มีในรายการ)
6. Remove all the items from the list. (ลบสิ่งของทั้งหมดในรายการ)
7. Retrieve (get) the item at a given position in the list. (นำสิ่งของกลับคืนมาในตำแหน่งที่ให้มีในรายการ)

จากรายการ ADT นำไปเขียนเป็น Pseudo code ได้ดังนี้

```
+createList()
//สร้างรายการว่างเปล่า
+isEmpty():boolean{query}
//สนใจรายการที่ว่างเปล่า
+size():integer{query}
//คืนค่าตัวเลขรายการทั้งหมด
+add(in index:integer, in item:ListItemType)
//เพิ่มข้อมูล (item) ในตำแหน่ง index ในรายการ ถ้า 0 <= index < size()
//ถ้า index = size()+1 ไม่เพิ่มข้อมูลในรายการเนื่องจากรายการเต็ม
+remove(in index:integer)
//ลบข้อมูลในตำแหน่งของ index ในรายการ ถ้า 0 <= index < size()
//ถ้า index < 0 ไม่ลบข้อมูลในรายการเนื่องจากรายการว่างเปล่า
+removeAll()
//ลบข้อมูลทั้งหมดในรายการ
+get(in index:ListItemType){queue}
//คืนค่า item ในตำแหน่ง index ของรายการ ถ้า 0 <= index < size()
//ให้เลื่อน index ไปทางซ้าย (ลดค่า index ลงหนึ่งตำแหน่ง) และไม่คืนค่าถ้า index เกินขอบเขตที่กำหนด
```

การออกแบบ ADT เป็นการพัฒนามาจากกระบวนการแก้ไขปัญหา ตัวอย่างเช่น ต้องการตรวจสอบวันหยุดทั้งหมดในหนึ่งปี ซึ่งมีวิธีการคือ ดูปฏิทินว่ามีวันหยุดวันใดบ้างในหนึ่งปี ด้วยการตรวจสอบทีละวันว่าวันใดเป็นวันหยุด เป็นต้น

เพื่อให้เกิดความเข้าใจให้ศึกษาตัวอย่างต่อไปนี้

ตัวอย่างที่ 1.8 การออกแบบรายการ ADT จากฟังก์ชันตรวจสอบวันหยุดที่มีคำสั่งดังนี้

```
+listHolidays(in year:integer)
//แสดงวันหยุดทั้งหมดในหนึ่งปีของ year ให้มา
date = วันที่ของวันแรกของปี
while(date ไม่ใช่วันแรกของปีถัดไป)
    if(date = วันหยุด){
        write(date + "คือวันหยุด")
    } //end if
    date = วันที่ถัดไป
} //end while
```


จากฟังก์ชันสามารถเขียนในรูปแบบ ADT เพื่อใช้ในการแก้ไขปัญหาการตรวจสอบวันหยุดในหนึ่งปีได้ดังนี้

1. Determine the date of the first day of a given year. (สนใจวันแรกของปีที่ให้มา)
2. Determine whether a date is before another date. (สนใจวันที่ในการตรวจสอบก่อนวันที่อื่นๆ)
3. Determine whether a date is a holiday. (สนใจวันที่ที่เป็นวันหยุด)
4. Determine the date of the day that follows a given date. (สนใจวันที่ถัดไปของวันที่กำหนดให้มา)

จาก ADT การตรวจสอบวันหยุดในหนึ่งปีนำไปเขียนเป็น Pseudo code ได้ดังนี้

```
+firstDay(in year:integer):Date{query}
//คืนค่าวันแรกของปีที่ให้มา
+isBefore(in date1:Date,date2:Date):boolean{query}
//คืนค่า true ถ้า date1 เป็นวันที่มาก่อน date2 นอกเหนือจากนั้นให้คืนค่า false
+isHoliday(in aDate:Date):boolean{query}
//คืนค่า true ถ้า aDate เป็นวันหยุด นอกเหนือจากนั้นให้คืนค่า false
+nextDay(in aDate:Date):Date
//คืนค่าวันที่ถัดไปจากวันที่กำหนดให้มาในตัวแปร aDate
```

ดังนั้น การออกแบบ ADT จะเป็นการแสดงข้อมูลและการเลือกตัวดำเนินการให้เหมาะสมกับการแก้ไข ปัญหา โดยที่ตัวดำเนินการนั้นมีการทำงานที่อิสระในการกำหนดคุณสมบัติของ ADT

ประเภทของอัลกอริทึม

การแยกประเภทของอัลกอริทึมเหมือนกับการแยกประเภทขั้นตอนการแก้ไขปัญหา โดยอธิบายการทำงานของอัลกอริทึมแต่ละประเภทได้ดังนี้

- **Brute Force algorithm** เป็นการแก้ไขปัญหาโดยสั่งให้ทำงานไปเรื่อยๆ จนกระทั่งได้คำตอบของทุกปัญหา การแก้ไขในรูปแบบนี้ไม่เหมาะสำหรับแก้ไขปัญหามีข้อมูลจำนวนมาก ตัวอย่างอัลกอริทึมที่ใช้หลักการ Brute Force algorithm เช่น การจัดเรียงข้อมูลแบบ Bubble sort, Selection sort เป็นต้น
- **Divide and Conquer algorithm** เป็นอัลกอริทึมที่มีหลักการคิดโดยแยกปัญหาออกเป็นสองส่วนคือ ส่วนที่หนึ่งแบ่งปัญหาออกเป็นส่วนเล็กๆ แล้วแก้ไขปัญหานั้นในส่วนเล็กๆ นั้นก่อน และอีกส่วนนำผลที่ได้จากการแก้ไขปัญหานั้นในส่วนเล็กๆ กลับมารวมกันใหม่ ตัวอย่างอัลกอริทึมที่ใช้หลักการ Divide and Conquer algorithm เช่น การจัดเรียงข้อมูลแบบ Quick sort, Merge sort เป็นต้น
- **Decrease and Conquer algorithm** เป็นอัลกอริทึมที่แก้ไขปัญหาวัยด้วยการลดขนาดของปัญหาลง และเลือกขนาดของกลุ่มปัญหาที่ต้องการแก้ไขปัญหานั้น โดยละเว้นปัญหาบางส่วนไว้ก่อนเพื่อจะแก้ปัญหามีขนาดเล็กกว่าเดิม เนื่องจากการแก้ไขปัญหามีขนาดเล็กกว่าจะสามารถแก้ไขปัญหานั้นได้ง่ายกว่า ตัวอย่างอัลกอริทึมที่ใช้หลักการ Decrease and Conquer algorithm เช่น การค้นหาข้อมูลแบบไบนารี เป็นต้น
- **Transform and Conquer algorithm** เป็นการแก้ไขปัญหาวัยด้วยการเปลี่ยนรูปแบบของปัญหาที่ต้องการแก้ไขให้อยู่ในรูปแบบอื่นก่อน ด้วยคาดหวังว่าเมื่อเปลี่ยนรูปแบบของปัญหาแล้วจะสามารถแก้ไขปัญหานั้นได้ง่ายและรวดเร็วขึ้น ตัวอย่างในการเปลี่ยนโครงสร้างข้อมูลก่อนการแก้ไข

สรุปเนื้อหาบทที่ 1

- โครงสร้างข้อมูลคือ การจัดการข้อมูลในหน่วยความจำภายในเครื่องคอมพิวเตอร์
- อัลกอริทึมคือ ลำดับขั้นตอนการทำงานเพื่อใช้ในการแก้ไขปัญหา
- ผังงาน เป็นเครื่องมือที่ช่วยออกแบบขั้นตอนการทำงานด้วยสัญลักษณ์
- โค้ดรหัสเทียม เป็นโครงสร้างรหัสที่มีการรวมกันทั้งภาษาเขียนกับภาษาคอมพิวเตอร์ เพื่อใช้ในการอธิบายโครงสร้างและลำดับขั้นตอนการทำงานของโปรแกรม โดยไม่อ้างอิงภาษาการเขียนโปรแกรมภาษาใดภาษาหนึ่ง
- Abstract Data Type เป็นการบอกถึงคุณสมบัติของโครงสร้างข้อมูลและกลุ่มตัวดำเนินการที่กระทำกับโครงสร้างข้อมูล
- ประเภทของอัลกอริทึมแยกได้เหมือนกับแยกรูปแบบในการแก้ไขปัญหาของโปรแกรม การแยกประเภทของอัลกอริทึมเพื่อแยกรูปแบบอัลกอริทึมในการแก้ปัญหานั้นเอง

แบบฝึกหัดท้ายบทที่ 1

ให้เขียนขั้นตอนวิธี ผังงาน และโค้ดรหัสเทียมของโปรแกรมต่อไปนี้

1. คำนวณเลขยกกำลังกำหนดให้รับตัวเลข 2 ค่าคือ ค่าเลขฐานและเลขยกกำลัง เพื่อนำมาคำนวณเลขยกกำลัง
2. เปลี่ยนค่าหน่วยเวลาจากหน่วยวินาทีที่รับเข้ามาให้เป็นหน่วยชั่วโมง นาที และวินาที
3. คำนวณอัตราแลกเปลี่ยนจากข้อมูลตัวเลขที่ป้อนเข้ามา กำหนดให้เป็นสกุลเงินบาทเปลี่ยนเป็นสกุลเงินประเทศสหรัฐอเมริกา ญี่ปุ่น และจีน กำหนดให้อ้างอิงอัตราแลกเปลี่ยนปัจจุบันในการซื้อขาย
4. รับข้อมูลตัวเลขจำนวน 20 ตัว กำหนดให้นำเฉพาะข้อมูลที่เป็นเลขคู่มาบวกกันและแสดงผลการบวก
5. คำนวณการถอนเงิน โดยให้รับข้อมูลตัวเลขเข้ามา 2 จำนวนคือ ยอดเงินที่ถูกค้าซื้อและจำนวนเงินที่ลูกค้าจ่าย แล้วให้คำนวณว่าต้องถอนแบงค์ 1,000 500 100 50 20 เหรียญ 10 5 2 1 .50 และ .25 บาท จำนวนอย่างละเท่าไร
6. รับคะแนนสอบของนักศึกษาจำนวน 20 คนว่ามีนักศึกษาที่ได้เกรดแต่ละเกรดจำนวนกี่คน และหาค่าเฉลี่ยว่านักศึกษากลุ่มนี้มีค่าคะแนนเฉลี่ยอยู่ที่เกรดอะไร โดยกำหนดค่าคะแนนในแต่ละช่วงเกรดดังนี้

คะแนน	เกรด
ต่ำกว่า 50	F
50-54	D
55-59	D+
60-64	C
65-69	C+
70-74	B
75-79	B+
80 ขึ้นไป	A

7. รับข้อมูลอายุจำนวน 20 คน กำหนดให้นับจำนวนคนในแต่ละช่วงวัยและหาค่าเฉลี่ยจากอายุที่รับเข้ามาว่าอยู่ในช่วงวัยไหน โดยแยกแต่ละช่วงวัยดังนี้
- วัยเด็กอายุน้อยกว่า 15 ปี
 - วัยรุ่นอายุมากกว่าและเท่ากับ 15 ปี ถึง 20 ปี
 - วัยผู้ใหญ่ตอนต้นอายุมากกว่า 20 ปี ถึง 40 ปี
 - วัยกลางคนอายุมากกว่า 40 ปี ถึง 60 ปี
 - วัยชราอายุมากกว่า 60 ปีขึ้นไป
8. รับข้อมูลตัวเลขจำนวนชั่วโมงการทำงานในแต่ละสัปดาห์จำนวน 4 สัปดาห์และประเภทของงานที่ทำงานเพื่อคำนวณค่าจ้าง โดยกำหนดให้ประเภทของงานและอัตราค่าจ้างดังนี้

ลำดับประเภทงาน	ชื่อประเภทงาน	อัตราค่าจ้าง/ชั่วโมง
0	แรงงานระดับใช้กำลัง	20
1	แรงงานระดับฝีมือ	25
2	แรงงานระดับชำนาญเฉพาะด้าน	30
3	แรงงานระดับฝีมือชำนาญเฉพาะด้าน	35

9. หาผลคูณจากเมตริกซ์ 2 เมตริกซ์ กำหนดเมตริกซ์ที่ 1 มีขนาด $n \times m$ และเมตริกซ์ที่ 2 มีขนาด $m \times n$ ผลลัพธ์ของการคูณเมตริกซ์ทั้งสองมีขนาด $n \times n$

คู่มือเรียน โครงสร้างข้อมูล

และ อัลกอริทึม

Data Structure & Algorithm

หนังสือเล่มนี้เป็นหนังสือประกอบการเรียนการสอนในรายวิชาโครงสร้างข้อมูลและอัลกอริทึม เน้นอธิบายการใช้งานโครงสร้างข้อมูลและหลักการทํางานของอัลกอริทึม ด้วยภาษา C และภาษา Java พร้อมคำอธิบายการทํางานอย่างละเอียดเป็นขั้นตอน เหมาะสำหรับนักเรียน นักศึกษาที่จะนำไปใช้ประกอบการเรียน

เนื้อหาภายในเล่ม

- รู้จักกับโครงสร้างข้อมูลและอัลกอริทึม
- การวิเคราะห์ประสิทธิภาพของอัลกอริทึม (Performance Analysis)
- รู้จักกับอาร์เรย์ (Array)
- รู้จักกับลิงคิสต์ทิศทางเดียว (Singly Link-list)
- รู้จักกับลิงคิสต์แบบสองทิศทาง (Doubly Link-list)
- การสร้างและใช้งานสแตค (Stack)
- การสร้างและใช้งานคิว (Queue)
- รู้จักและใช้งาน Binary Tree, AVL Tree, 2-3 Trees, red-black Tree, B-Tree
- เรียนรู้การทํางานของ Hash
- รู้จักกับไทร (Tries)
- เรียนรู้การสร้างและใช้งานฮีพ (Heap)
- เรียนรู้การจัดเรียงข้อมูลแบบ Selection sort, Insertion sort, Merge sort, Quick sort, Heap sort
- เรียนรู้การค้นหาข้อมูลแบบ Binary search
- เรียนรู้การท่องกราฟแบบ Depth-First Search (DFS) และ Breadth-First Search (BFS)
- รู้จักกับ Minimum Spanning Tree
- เรียนรู้การหาเส้นทางที่สั้นที่สุด (Shortest Paths)
- เรียนรู้การทํางานของ Kruskal's Algorithm
- เรียนรู้การทํางานของ Dijkstra's Algorithm

ผู้แต่ง : **วิษณุ ช้างเนียม**

Sun Certified Java Programmer (SCJP)

ปริญญาตรีวิศวกรรมคอมพิวเตอร์ ศูนย์กลางสถาบันเทคโนโลยีราชมงคล

ปริญญาโทวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเชียงใหม่

ปัจจุบันเป็นอาจารย์ประจำสาขาวิชาระบบสารสนเทศทางคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ลำปาง

e-Book
VERSION

จัดจำหน่ายโดย **IDC**
ISBN 978-616-200-652-4

สงวนลิขสิทธิ์
200
บาท



9 786162 006524

• FREE CD

Slide
Source Code

วิษณุ ช้างเนียม
บรรณาธิการ
กิตตินันท์ พลสวัสดิ์

DEV BOOK