บทที่ 1 โครงสร้างข้อมูลเบื้องต้น

1.1 ข้อมูล
1.2 การจัดการข้อมูล
1.3 โครงสร้างข้อมูล
1.4 ตัวอย่าง : แบบชนิดข้อมูล
1.5 การประกาศโครงสร้างข้อมูลในภาษาโปรแกรม
1.6 การแปลงส่งไปหน่วยเก็บ : จำนวนเต็ม
1.7 การแปลงส่งไปหน่วยเก็บ : ตัวอักขระ
1.8 การแปลงส่งไปหน่วยเก็บ : สายอักขระ
1.9 การเลือกการแปลงส่งที่เหมาะสม
บทสรุป
แบบฝึกหัด

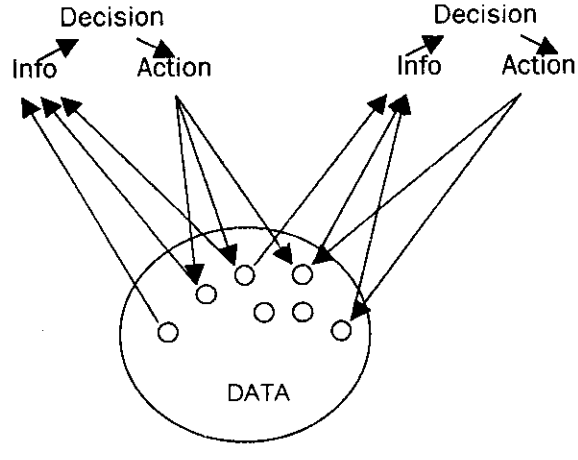
IT 204 1

บทที่ 1
โครงสร้างข้อมูลเบื้องต้น
(Introduction to Data Structures)

ส่วนแรกของหนังสือเล่มนี้อภิปรายโครงสร้างข้อมูลชนิดที่สำคัญและบอกว่ามันถูกจัดการโดยโปรแกรมคอมพิวเตอร์อย่างไร ความรู้เบื้องต้นบทนี้จะทำให้นักศึกษามีความเข้าใจว่าโครงสร้างข้อมูลคืออะไรและทำไมโครงสร้างข้อมูลจึงมีความสำคัญในโลกของระบบสารสนเทศเช่นเดียวกับความรู้ทั่วไปของโครงสร้างข้อมูลอย่างง่ายหลายชนิด

1.1 ข้อมูล (Data)

ก่อนที่จะเริ่มต้นอภิปรายเรื่องโครงสร้างข้อมูล สิ่งแรกให้พิจารณาว่า ข้อมูลคืออะไรและมันสนับสนุนระบบสารสนเทศอย่างไร รูปที่ 1 แสดงให้เห็นบทบาทซึ่งข้อมูลมีอยู่ในเหตุการณ์ต่าง ๆขององค์กร วงกลมใหญ่ที่เขียนว่า "data" แทนแหล่งข้อมูลทั้งหมดขององค์กร แหล่งข้อมูลนี้คือ ผู้ที่มีส่วนทำงาน (active participant) ในการดำเนินงานและการวางแผนขององค์กร วงกลมเล็กแทนสมาชิกแต่ละบุคคลหรือ items ของข้อมูล สมาชิกเหล่านี้ถูกพิจารณาว่าเป็นข้อเท็จจริง(raw facts) เมื่อนำมารวมเข้าด้วยกันและทำสรุปในวิธีที่มีความหมายต่าง ๆ เพื่อประกอบกันเป็นสารสนเทศ การตัดสินใจกระทำบนมูลฐานของสารสนเทศที่เราหวังว่าการตัดสินใจที่กระทำคือการกล่าวถึงสิ่งหนึ่ง การทำให้เกิดผลในทางปฏิบัติของผลลัพธ์ของการตัดสินใจ คือ การกระทำ (actions) ซึ่งก่อให้เกิดข้อมูลมากขึ้น หลังจากนั้นข้อมูลเหล่านี้สามารถนำมารวามกันเพื่อให้เป็นวงกลมอีกว่าหนึ่งของกระบวนการตัดสินใจ



รูปที่ 1 การใช้ข้อมูลโดยเหตุการณ์ต่าง ๆ ในองค์กร

โปรดสังเกตว่า สมาชิกข้อมูลที่กำหนดให้ใด ๆ ก็ตามอาจมีส่วนร่วมในการก่อกำเนิดของชิ้นต่างๆ มากมายของสารสนเทศสิ่งที่มีความสำคัญซึ่งมีความคล่องตัวในวิธีต่าง ๆ ซึ่ง data item ถูกนำมารวมกันและทำสรุป เพื่อให้สารสนเทศมีความหมายสามารถถูกผลิตให้สนับสนุนการตัดสินใจที่มีอยู่ในมือ โปรดสังเกตด้วยว่าสมาชิกข้อมูลบางตัว ในแหล่งข้อมูลขององค์กรอาจให้ผลลัพธ์จาก actions ที่นำมาโดยการบังคับหรือจากแหล่งภายนอกไปยังองค์กร การกระทำของผู้แข่งขัน พฤติกรรมของลูกค้า และถูกบังคับโดยความต้องการที่ถูกต้องทั้งหมดที่เข้ามายังกระบวนการทำการตัดสินใจขององค์กร

แหล่งข้อมูลต้องสนับสนุนชนิดต่างๆ ของการทำการตัดสินใจ มีข้อดีที่จะให้รู้จักระดับการตัดสินใจ 3 ระดับดังนี้ (It is convenient to recognize three levels of decisions:)

1. การตัดสินใจเชิงปฏิบัติการ (operational decisions) ซึ่งควบคุมการดำเนินงานทุกวันขององค์กร การตัดสินใจเหล่านี้กระทำโดยการจัดการระดับล่าง
2. การตัดสินใจควบคุม (Control decisions) ซึ่งกำหนดวิธีที่องค์กรกระทำการ หน้าที่ที่กำหนดของมัน การตัดสินใจเหล่านี้บางครั้งเรียกว่า การตัดสินใจตามยุทธวิธี (tactical decisions) และกระทำโดยการจัดการระดับกลาง
3. การตัดสินใจวางแผน (Planning decisions) ซึ่งพัฒนาและนิยาม หน้าที่ขององค์กร การตัดสินใจเหล่านี้บางครั้งเรียกว่า การตัดสินใจเชิงยุทธศาสตร์ (Strategic decision) และกระทำโดยระดับบนของการจัดการ

สมาชิกตัวเดียวกันของข้อมูลอาจมีส่วนร่วมในการผลิตของสารสนเทศเพื่อสนับสนุนทั้งสามระดับของการตัดสินใจตัวอย่างเช่น สมาชิกข้อมูลหนึ่งตัว เลขที่ของชุดโทรศัพท์มือถือสีเขียวผลิตโดยเครื่องประทับตราพลาสติก #12345-78 อาจนำมาใช้ในการตอบแต่ละคำถามข้างล่างนี้

1. จำนวนชุดโทรศัพท์มือถือสีเขียวซึ่งเครื่อง #12345-78 ตั้งให้ผลิตในวันนี้มีกี่เครื่อง ? (Operational)
2. จากข้อมูลการผลิต ความล้มเหลวและการซ่อมแซมเราควรจะซื้อเครื่องประทับตราพลาสติกสีเขียวเพิ่มอีก 3 ชุดสำหรับทำชุดโทรศัพท์มือถือหรือไม่ (Control)
3. บริษัทควรจะเข้ายึดบริษัทสาขาซึ่งผลิตเครื่องประทับตราพลาสติก ดังนี้การเพิ่มความสามารถของมันเพื่อโต้ตอบการบังคับการเปลี่ยนแปลงตลาดหรือไม่ (Planning) การตัดสินเหล่านี้แต่ละข้อต้องขึ้นอยู่กับข้อมูลที่ถูกต้อง

1.2 การจัดการข้อมูล (Data Management)

ข้อมูลมีราคาแพงจึงต้องถูกจัดการในวิธีซึ่งให้ข้อมูลถูกต้องและเอาไปใช้ได้เพื่อผลิตสารสนเทศที่ต้องการ ชนิดต่าง ๆของการจัดกระทำข้อมูลได้แก่

การวัด (measurement)	การเก็บ (Storage)
การรวบรวม (Collection)	การรวมกัน (Aggregation)
การคัดลอก (Transcription)	การปรับให้ทันกาล (Update)
ความถูกต้อง (Validation)	การค้นคืน (Retrieval)
การจัดระเบียบ (Organization)	การป้องกัน (Protection)

ระบบการจัดการข้อมูลเกี่ยวข้องกับการทำสิ่งต่าง ๆข้างต้นเหล่านี้กับข้อมูล วัตถุประสงค์สุดท้ายคือทำให้แหล่งข้อมูลยืดหยุ่นได้ , คล่องตัว และสามารถดัดแปลงง่ายเพื่อสนับสนุนกระบวนการต่างๆ ของการทำการตัดสินใจขององค์กร

แนวทางที่เป็นประโยชน์ 4 ข้อสำหรับระบบการจัดการข้อมูลได้แก่ (Four useful guidelines for a data management system are :)

1. ข้อมูลต้องถูกแทนที่และเก็บเพื่อให้สามารถถูกเข้าถึงได้ในภายหลัง (Data must be represented and stored so that they can be accessed later.)
2. ข้อมูลต้องถูกจัดระเบียบเพื่อให้มันสมารถถูกเลือกและเข้าถึงได้อย่างมีประสิทธิภาพ (Data must be organized so that they can be selectively and efficiently accessed.)
3. ข้อมูลต้องถูกประมวลผลและนำเสนอเพื่อให้มันสนับสนุนสิ่งแวดล้อมผู้ใช้ได้อย่างมีประสิทธิผล (Data must be processed and presented so that they support the user environment effectively)
4. ข้อมูลต้องถูกปกป้องและจัดการเพื่อให้มันเก็บรักษาค่าของมัน (Data must be protected and managed so that they retain their value.)

แนวทางเหล่านี้คือการแรงจูงใจมากของสาระที่ครอบคลุมในหนังสือเล่มนี้ เหตุผลหลักสำหรับความสำคัญของโครงสร้างข้อมูลคือมันช่วยเราเพื่อให้แนวทางข้อ(1) และ (2) ประสบผลสำเร็จ

เราจะได้อภิปรายการแทนที่การเก็บและการจัดระเบียบข้อมูลทั้งบนอุปกรณ์หน่วยเก็บหลักและอุปกรณ์หน่วยเก็บรองสำหรับหน่วยเก็บหลัก เวลาที่ใช้เข้าถึงข้อมูลที่เก็บในหนึ่งตำแหน่งจะเหมือนกับเวลาที่จำเป็นต้องใช้เพื่อเข้าถึงข้อมูลที่เก็บในตำแหน่งอื่นบนอุปกรณ์

หน่วยเก็บหลัก(primary storage) ปกติเรียกว่า main memory สำหรับหน่วยเก็บรอง เวลาที่ใช้เข้าถึงข้อมูลในหนึ่งตำแหน่งอาจจะแตกต่างจากเวลาที่ต้องใช้เพื่อเข้าถึงข้อมูลเก็บในอีกหนึ่งตำแหน่งบนอุปกรณ์ตัวเดียวกัน

หน่วยเก็บรอง (secondary storage) อาจเรียกอีกชื่อหนึ่งว่าหน่วยความจำช่วย (auxiliary memory) ได้แก่จานบันทึกชนิด movable-head, จานบันทึกชนิด fixed-head, แถบบันทึกแม่เหล็ก, บัตรเจาะรู และอื่น ๆ

การจัดการข้อมูลจะมีความแตกต่างกันสำหรับข้อมูลซึ่งเก็บในหน่วยเก็บหลักและหน่วยเก็บรอง ส่วนแรกของหนังสือเล่มนี้จะพิจารณาการจัดการข้อมูลในหน่วยเก็บหลักบนโครงสร้างข้อมูล และในส่วนที่สองการจัดการข้อมูลในหน่วยเก็บรองบนการจัดระเบียบและการประมวลผลแฟ้ม

1.3 โครงสร้างข้อมูล (Data Structures)

เราเริ่มต้นศึกษาเรื่องการแทนที่ข้อมูล การจัดระเบียบและการประมวลผลจากแง่คิดของโครงสร้างข้อมูล

โครงสร้างข้อมูลหมายถึงประเภทของข้อมูลซึ่งถูกจำแนกคุณสมบัติโดยการจัดระเบียบและการปฏิบัติการที่ให้นิยามให้กับประเภทข้อมูลนั้น

(A data structure is a class of data that can be characterized by its organization and the operation that are defined on it)

โครงสร้างข้อมูลบางครั้งเรียกว่าแบบชนิดข้อมูล (data types)

โครงสร้างข้อมูลมีความสำคัญมากในระบบคอมพิวเตอร์ ในโปรแกรมตัวแปรทุกตัวคือโครงสร้างข้อมูลซึ่งถูกนิยามอย่างชัดแจ้งหรือโดยนัย ซึ่งจะบอกถึงเซตของการดำเนินงานซึ่งถูกต้องบนตัวแปรนั้น

โครงสร้างข้อมูลซึ่งเราอภิปรายในที่นี้คือเทอม โครงสร้างข้อมูลเชิงตรรกะ (logical data structures) ในแง่ที่ว่ามีการแทนที่เชิงกายภาพแตกต่างกันหลายชนิดบนหน่วยเก็บที่เป็นไปได้สำหรับโครงสร้างข้อมูลเชิงตรรกะที่กำหนดให้ สำหรับโครงสร้างข้อมูลแต่ละชนิดที่เราจะพิจารณาการแปลงส่งที่เป็นไปได้หลายชนิดกับหน่วยเก็บจะได้นำมาแนะนำด้วย

ตารางที่1 การจำแนกโครงสร้างข้อมูลซึ่งอภิปรายในหนังสือเล่มนี้

DaTa Types:	Primitive	Integer Boolean Character
	Compound	String

Data Structures:	Simple	Array Record
	Linear:	Stack Queue Linear linked list
	Compound	Binary tree
	Nonlinear:	Binary search tree M-way search tree B-tree,B*_tree,b+_tree Trie General tree Graph
File Structure :	Sequential organization Relative organization Indexed Sequential organization Multi-key organization	

ตารางที่1 จำแนกชนิดของโครงสร้างข้อมูลหลากหลายซึ่งจะได้อภิปรายในหนังสือเล่มนี้
โครงสร้างข้อมูลแบบเก่าหมายถึงโครงสร้างข้อมูลซึ่งไม่ได้ประกอบด้วยโครงสร้างข้อมูล
อื่น ๆ
(Primitive data structures are not composed of other data structures.)
ปกติเรียกว่าชนิดข้อมูลแบบเก่า (Primitive data types) แบ่งออกเป็น 3 ชนิดคือ integers, booleans และ characters

ชนิดข้อมูลแบบประกอบสร้างจากชนิดข้อมูลแบบเก่าหนึ่งชนิดหรือมากกว่าขึ้นไป (A compound data type is constructed from one or more primitive data types.)

ตัวอย่างของชนิดข้อมูลประกอบได้แก่สายอักขระ(string) ภาษาโปรแกรมจำนวนมากให้การสนับสนุนสำหรับแบบชนิดข้อมูลเหล่านี้

แบบชนิดข้อมูลสามารถจำแนกได้หลายวิธีเพื่อประกอบเป็นโครงสร้างข้อมูล เส้นแบ่งระหว่างแบบชนิดข้อมูลกับโครงสร้างข้อมูลยังไม่ชัดเจน และคำศัพท์ได้วิวัฒนาการไป โดยเรียกการจัดองค์กรของแบบชนิดข้อมูลบางอย่างว่า "โครงสร้างข้อมูล" และเรียกแบบอื่น ๆว่า " compound data types"

โครงสร้างข้อมูลอย่างง่ายสองชนิดที่จะได้พิจารณาคือ แถวลำดับ (array) และระเบียน (records) โครงสร้างข้อมูลเหล่านี้สามารถรวมเข้าด้วยกันหลายวิธีเพื่อประกอบเป็นโครงสร้างข้อมูลซับซ้อนมากขึ้นซึ่งบางครั้งเรียกว่า โครงสร้างข้อมูลเชิงประกอบ (compound data structures) ในหนังสือเล่มนี้ส่วนแรกจะเกี่ยวข้องกับโครงสร้างข้อมูลเชิงประกอบเป็นส่วนใหญ่โครงสร้างข้อมูลเชิงประกอบมีสองชนิดคือ linear และ nonlinear ขึ้นอยู่กับความซับซ้อนของความสัมพันธ์เชิงตรรกะที่มันแทนที่

โครงสร้างข้อมูลแบบเชิงเส้น (linear data structures) ซึ่งเราจะได้อภิปรายได้แก่ stacks, queues, และ linear linked lists.

ส่วนโครงสร้างข้อมูลไม่ใช่แบบเชิงเส้น (nonlinear data structures) ได้แก่ trees และ graphs เราจะพบว่ามีโครงสร้างรูปต้นไม้หลายชนิดซึ่งเป็นประโยชน์ในการจัดการข้อมูล โครงสร้างข้อมูลซึ่งประยุกต์กับการรวบรวมของข้อมูลในหน่วยเก็บรองเรียกว่า การจัดระเบียบแฟ้ม (The data structures applied to collection of data in secondary storage are called file organizations.)
การจัดระเบียบแฟ้มแบ่งเป็น 4 ชนิดที่สำคัญคือ
sequential, relative, indexed sequential, และ multi-key file organization

การมีความเข้าใจเกี่ยวกับเทคนิคเหล่านี้สามารถนำไปสู่การศึกษาของวิธีสำหรับจะจัดการระบบฐานข้อมูลสากลโดยตรง

1.4 ตัวอย่าง : แบบชนิดข้อมูล (Examples : Data types)

แบบชนิดข้อมูลอย่างง่ายชนิดแรก คือ จำนวนเต็ม (Integer) จำนวนเต็มคือสมาชิกของเซตของจำนวนต่อไปนี้

{ . . . , -(n+1) , -n , . . . , -2 , -1 , 0 , 1 , 2 , . . . , n , n+1 , . . . }

การปฏิบัติการพื้นฐานบนจำนวนเต็มที่คุ้นเคยได้แก่การบวก การลบ การคูณ การหาร การยกกำลัง

และ อื่น ๆ การดำเนินการเหล่านี้ทั้งหมดทำงานบนคู่ของจำนวน จึงเป็นตัวดำเนินการแบบทวิภาค (binary operators)

ตัวดำเนินการเอกภาค (unary operator) จะมีตัวถูกกระทำ (operand) เพียงหนึ่งตัวเท่านั้น

นิเสธ(Negation) ซึ่งเปลี่ยนเครื่องหมายของจำนวนคือตัวอย่างของตัวดำเนินการเอกภาค

ตัวอย่างที่สองของโครงสร้างข้อมูลอย่างง่ายได้แก่ แบบชนิดข้อมูล boolean หรือเรียกชื่ออีกอย่างหนึ่งว่า logical data type

สมาชิกข้อมูลชนิดนี้มีค่าใดค่าหนึ่งในสองค่าคือ true หรือ false (A data element of boolean type can have one of two values : true or false)

การให้นิยามเซตของการดำเนินการบนโครงสร้างข้อมูลชนิดนี้ แตกต่างจากเซตของการดำเนินการที่ถูกต้องสำหรับ integers ตัวดำเนินการแบบบลูที่เป็นพื้นฐานมี 3 ชนิดคือ not, and และ or

ตารางที่ 2 Boolean Operators

Value of First Operand	Value of Second Operand	Operator		
		and	or	not *
true	true	True	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

*ประยุกต์กับ operand ตัวแรก

ตารางที่ 2 แสดงให้เห็นผลลัพธ์ของการประยุกต์ใช้ตัวดำเนินการเหล่านี้ให้กับแต่ละตัวของค่า boolean ที่ถูกต้องโปรดสังเกตว่า or และ and เป็นตัวดำเนินการแบบทวิภาคทั้งคู่ในขณะที่ not เป็นตัวดำเนินการแบบเอกภาค not มีการทำก่อน(precedence) สูงกว่า or และ and นั้นคือ เมื่อไม่มีวงเล็บกำกับซึ่งเปลี่ยนแปลงอันดับของการประเมินผล not จะถูกประเมินผลก่อน and หรือ or

ตัวอย่าง การประเมินผลนิพจน์

A and not B

อันดับแรก B มีค่าเป็นนิเสธและผลลัพธ์ที่ได้ and กับ A

ค่า true และ false อาจเป็นผลลัพธ์จากตัวดำเนินการสัมพันธ์ (relational operators) ได้เช่นกัน

ตัวดำเนินการสัมพันธ์ไม่จำเป็นต้องมีตัวถูกดำเนินการแบบบลูแต่มันเป็นตัวดำเนินการที่ให้ผลลัพธ์มีชนิดเป็นแบบบลู ตัวดำเนินการเหล่านี้ได้แก่ $>$, $<$, $=$, $\leq$, $\not>$, $\not<$, $\neq$, $=$ ตัวอย่างเช่น

$<$ ดำเนินการบน integer สองจำนวนให้ผลลัพธ์เป็นค่าข้อมูลแบบบลู :

$6 < 12$ คือ true

แบบชนิดข้อมูลเก่าแก่ ชนิดที่ 3 ซึ่งเราน่าจะมีความคุ้นเคยมาแล้วได้แก่ ตัวอักขระ (Character)

ตัวอักขระคือสมาชิกซึ่งเอามาจากเซตของสัญลักษณ์ (A character is an element taken from a set of symbols.)

เซตของสัญลักษณ์ที่เป็นไปได้ชุดหนึ่งได้แก่

{ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , A , B , C , D , . . . , X , Y , Z , ? , . , * , + , - , / }

ซึ่งประกอบด้วยเลขโดด ตัวอักษรและตัวอักขระพิเศษอื่น ๆ คอมไพเลอร์หลากหลายจะมีเซตของอักขระที่แตกต่างกันซึ่งปกติแตกต่างในตัวอักขระพิเศษที่ใช้

แบบชนิดข้อมูลเชิงประกอบ (Compound data type) ซึ่งสร้างขึ้นมาจากแบบชนิดข้อมูลตัวอักขระเรียกว่า **สายอักขระ (String)**

สายอักขระหมายถึงลำดับจำกัดของสัญลักษณ์เอามาจากชุดอักขระ (A string is a finite sequence of symbols taken from a character set.)

ชุดอักขระที่ใช้ก่อกำเนิดสายอักขระต่าง ๆ เรียกว่า**ชุดตัวอักษร** (The character set used to generate strings is called an alphabet.) เซตของสายอักขระต่าง ๆ ซึ่งสามารถได้มาจากชุดตัวอักษร A = { C , D , 1}

ได้แก่ ' CD1' , ' CD ' , ' BDC ' , ' 1d111' และอื่น ๆ รวมทั้งสายอักขระว่าง (null or empty string) เราวิเคราะห์การเริ่มต้นและจบของสายอักขระด้วยเครื่องหมายคำพูด

สายอักขระเป็นแบบชนิดข้อมูลที่มีความสำคัญมากส่วนหนึ่งเป็นเพราะว่ามันถูกนำมาใช้อย่างแพร่หลายมาก

ตัวอย่างเช่น สายอักขระคือสื่อพื้นฐานซึ่งโปรแกรมเมอร์เขียนและส่งไปยังคอมพิวเตอร์ มันเป็นสื่อหลักของการแลกเปลี่ยนสารสนเทศกับผู้ใช้ต่าง ๆ

สายอักขระถูกนำมาใช้ในเขตข้อมูลของระเบียนภายในแฟ้มเพื่อเก็บสารสนเทศ มันถูกนำมาใช้ในภาษาโปรแกรมให้เป็นชื่อตัวแปร ลาเบล และ โปรซีเดอร์

ในบริบทโดยทั่วไปมากกว่านี้มันเป็นสายอักขระซึ่งมนุษย์ใช้สื่อสารกับบุคคลอื่น

เซตของสายอักขระทั้งหมดที่เป็นไปได้ซึ่งได้มาจากชุดตัวอักษร เรียกว่า **คำศัพท์** (The set of all possible strings that can be derived from an alphabet is called a **vocabulary**.)

คำศัพท์ V ซึ่งได้มาจากชุดตัวอักษร A บางครั้งใช้สัญลักษณ์

$$V_A = A^*$$

ชุดตัวอักษรไม่จำเป็นต้องเป็นตัวอักขระจากชุดตัวอักษรที่เราคุ้นเคย { A , B , C , . . . , X , Y , Z } มันอาจจะเป็นสัญลักษณ์ถูกต้องใด ๆ ก็ได้ ถ้าชุดตัวอักษรคือ { 0 , 1 } เพราะฉะนั้นสายอักขระได้มาปกติเรียกว่า **สายบิต (bit strings)** เพื่อให้เป็นทางการมากขึ้นให้แทนสายอักขระ S ซึ่งได้มาจากชุดตัวอักษร A ดังนี้

$$S := ' a_1 , a_1 , . . . , a_N \qquad (1)$$

เมื่ออักขระแต่ละตัว $a_1 \in A$ for $1 \leq I \leq N$

สายอักขระทุกชุดมีลักษณะประจำเรียกว่า**ความยาว**ของมัน ซึ่งหมายถึง จำนวนตัวอักขระทั้งหมดที่ประกอบกันเป็นสายอักขระ

(Every string has an attribute called its length which is the number of characters in the string.)

การดำเนินการต่าง ๆซึ่งนิยามถูกต้องบนสายอักขระแตกต่างจากการดำเนินการซึ่งถูกนิยามอย่างถูกต้องสำหรับจำนวนเต็ม การดำเนินการที่สำคัญบนสายอักขระมี 3 ชนิด คือ

1. ความยาว (Length)
2. การต่อกัน (Concatenation)
3. สายอักขระย่อย (Substring)

ตัวดำเนินการความยาวให้แทนด้วย LENGTH ซึ่งให้ค่าของลักษณะประจำความยาวของ

สายอักขระ ตัวดำเนินการนี้มีตัวถูกดำเนินการหนึ่งตัวมีชนิดเป็นสายอักขระและให้ผลลัพธ์ซึ่งเป็นชนิด integer เราสามารถประยุกต์ LENGTH ให้กับสายอักขระ S ซึ่งนิยามในสมการ (1) ให้ผลลัพธ์ N โดยเขียนดังนี้

N := LENGTH (S)

การต่อกันกระทำบนสายอักขระสองชุด ใส่สายอักขระชุดหนึ่งต่อท้ายสายอักขระอีกชุดหนึ่งให้ผลลัพธ์เป็นสายอักขระตัวดำเนินการต่อกันใช้สัญลักษณ์ CONCAT ซึ่งมีตัวถูกกระทำสองตัว ทั้งคู่มีชนิดเป็น string และให้ผลลัพธ์เป็นชนิด string ถ้า S ถูกนิยามโดยสม การที่ (1) และ S^0 นิยามดังนี้

$$S^0 := b_1, b_1, \ldots, b_M \qquad (2)$$

เมื่ออักขระแต่ละตัว $b_j \in B$, for $j \leq I \leq m$ บนชุดตัวอักษร B เพราะฉะนั้น

CONCAT (S, S^0) คือ '$a_1\ a_2, \ldots, a_N\ b_1\ b_2, \ldots, b_m$'

ความยาวของสายอักขระผลลัพธ์คือผลรวมของความยาวของสายอักขระที่เป็นส่วนประกอบของมัน :

LENGTH (CONCAT(S, S^0)) = LENGTH (S) + Length (S^0)

ตัวอย่าง การต่อกันของสายอักขระที่มีค่า 'STOVE' กับสายอักขระที่มีค่า 'PIPE' ผลลัพธ์คือสายอักขระ 'STOVEPIPE'

สายอักขระย่อยมีสายอักขระหนึ่งชุดเป็นตัวถูกดำเนินการของมันและสร้างสายอักขระใหม่อีกชุดหนึ่งเป็นผลลัพธ์ เพื่อให้การกำหนดอย่างครบถ้วนกับการดำเนินการสายอักขระย่อย ไม่เพียงแต่ต้องการกำหนดสายอักขระที่เป็นตัวถูกดำเนินการเท่านั้นแต่ยังต้องกำหนดจุดเริ่มต้นภายในสายอักขระชุดนั้นด้วยและจำนวนตัว อักขระที่จะเอาจากสายอักขระนั้นมาประกอบเป็นสายอักขระชุดใหม่ตัว ดำเนินการสายอักขระย่อยให้แทนด้วย SUBSTR ตัวดำเนินการนี้มีตัวถูกดำเนินการ 1 ตัวมีชนิดเป็น string และมีตัวถูกดำเนินการอีกสองตัวมีชนิดเป็น integer ให้ผลลัพธ์ ซึ่งมีชนิดเป็น string ถ้า S ถูกนิยามโดยสมการที่ (1) ดังนั้น

SUBSTR (S, i , j) คือ '$a_i\ a_{i+1}, \ldots, a_{i+j-1}$'

เมื่อ i คือจุดเริ่มต้น , $0 \leq i \leq$ LENGTH (s)

j คือจำนวนตัวอักขระที่ต้องการ $0 \leq j \leq$ LENGTH(s)

และ $0 \leq i+j-1 \leq$ LENGTH (S)

สำหรับ i , j ที่มีชนิดเป็น integer

LENGTH (SUBSTR(S , i , j)) เท่ากับ j

ตัวดำเนินการ SUBSTR สามารถนำมาใช้เพื่อ **ลบล้าง** (undo) ผลกระทบของตัวดำเนินการ CONCAT :

SUBSTR (CONCAT (S,S^o), LENGTH (S)) คือ S

และ

SUBSTR (CONCAT (S,S^o) , LENGTH (S)) + 1 , LENGTH S^o คือ S^o

Composite Operations

ตัวดำเนินการสายอักขระสามตัวนี้ – length , concatenation และ substring จะไม่มีความหมายแต่อย่างใด เมื่อนำไปประยุกต์ใช้กับตัวแปรซึ่งมีแบบชนิดข้อมูลเป็น integer ถ้าเราต้องการใส่จำนวนเต็ม 12 เข้ากับจำนวนเต็ม 34 ไม่น่าจะหมายถึงการต่อกันให้ผลลัพธ์เป็น 1234 แต่ควรจะเป็นการบวกให้ผลลัพธ์เท่ากับ 46 อย่างไรก็ตามมันถูกต้องแน่นอนเมื่อต่อสายอักขระ '12' กับสายอักขระ '34' ซึ่งให้ผลลัพธ์เป็นสายอักขระชุดที่สาม '1234' ที่มีความยาวเท่ากับ 4

จำนวนเต็มบางครั้งมีการแปลงผัน (convert) ให้เป็นสายอักขระและจัดการในแบบสายอักขระหลังจากนั้นจึงแปลงผันกลับมาเป็นจำนวนเต็ม มีการดำเนินการผลประกอบอื่น ๆ (other composite operations) ซึ่งถูกต้องบนสายอักขระ ตัวอย่างเช่นตัวดำเนินการใส่ให้แทนด้วย INSERT มันมีตัวถูกดำเนินการสองชุดมีชนิดเป็น string และตัวดำเนินการตัวที่สามมีชนิดเป็น integer ให้ผลลัพธ์เป็นชนิด string

INSERT(S, S^o,i) ใส่สายอักขระ S^o เข้าไปในสายอักขระ S โดยที่อักขระตัวแรกของ S^o คือ อักขระตัวที่ i ของผลลัพธ์

INSERT(S, S^o,i) คือ CONCAT(CONCAT (SUBSTR(S,i,i-1), S^o),

SUBSTR(S,i,LENGTH(S) – (i-1)))

เมื่อ $1 \leq i \leq$ LENGTH (S) +1

ตัวดำเนินการลบทิ้งให้แทนด้วย DELETE มันมีตัวถูกดำเนินการหนึ่งตัวมีชนิดเป็น string และมีตัวถูกดำเนินการอีกสองตัวมีชนิดเป็น integer ให้ผลลัพธ์เป็นชนิด string

DELETE(S , i , j) ลบทิ้งจาก S ด้วยสายอักขระย่อยความยาว j ซึ่งเริ่มต้นที่ตัวอักขระตำแหน่งที่ i ตัวอย่าง DELETE ('MAGIC' , 2 ,2) = 'MIC'

(DELETE(S , i , j) deletes from S the substring of length j that starts at the i^{th} character.)

(DELETE(S , i , J) is CONCAT (SUBSTR(S , i , J), (SUBSTR(S , i+J), LENGTH(S) – (i+j–1)))

where $1 \leq i \leq$ LENGTH (S)

$0 \leq j \leq$ LENGTH (S)

$0 \leq i+j-1 \leq$ LENGTH (S)

for i,j of type integer

เราแยกความแตกต่างระหว่างค่า integer กับค่า string โดยการกั้นเขตของตัวหลังด้วยเครื่องหมายคำพูดเพื่อให้แตกต่างกันระหว่างค่า integer และค่า string ในภาษาโปรแกรมปกติใช้ข้อตกลงเดียวกันนี้

ถ้า 1234 เป็น integer ให้เขียน 1234

ถ้า 1234 คือ string ให้เขียน '1234' หรือ "1234"

ในวิธีเดียวกันภาษาโปรแกรมแยกความแตกต่างกันค่าของ string กับชื่อของตัวแปร ตัวอย่างเช่น SUMMER ที่เป็น string ดังนั้นสายอักขระ SUMMER เขียนดังนี้ 'SUMMER' หรือ " SUMMER" ส่วนชื่อตัวแปร SUMMER เขียนดังนี้ SUMMER

1.5 การประกาศของโครงสร้างข้อมูลในภาษาโปรแกรม (Declaration of Data Structures in Programming Languages)

ภาษาโปรแกรมให้โปรแกรมเมอร์เป็นผู้ให้ความหมายที่กำหนดชนิดโครงสร้างข้อมูลให้กับตัวแปร

ตัวแปรต้องมีค่าซึ่งเอามาจากเซตของค่าต่าง ๆ ที่ถูกนิยามแล้วสำหรับแบบชนิดข้อมูลของมันและเฉพาะการดำเนินการต่างๆ ซึ่งนิยามแล้วเป็นสิ่งถูกต้องสำหรับชนิดข้อมูลนั้นที่ควรจะกระทำบนตัวแปร

ภาษาโปรแกรมต่าง ๆ มีวิธีแตกต่างกันของการกำหนดชนิดให้กับตัวแปร คอมไพเลอร์บางตัวถือเป็นเรื่องสำคัญมากกว่าทำสิ่งอื่น ๆ เพื่อรับผิดชอบการตรวจทานที่ติดมากับกฎซึ่งจำแนกโครงวสร้างข้อมูลโดยเฉพาะ ตัวอย่างเช่นคอมไพเลอร์บางตัวไม่สนใจว่าโปรแกรมย้ายตัวอักขระที่เป็นตัวอักษรเช่น 'Q' ไปยังเขตข้อมูลชนิด integer มันจะให้โปรแกรมทำงานต่อไปและบวก 'Q' กับสิ่งอื่นโดยไม่มีการเตือนหรือให้ข้อสังเกตของผลลัพธ์

ที่ไม่มีความหมายใด ๆ ที่น่าจะเป็นไปได้ ส่วนคอมไพเลอร์อื่น ๆ อาจจะสร้างข้อผิดพลาดเวลาคอมไพล์ (compile-time error) หรือ ข้อผิดพลาดเวลากระทำการ (execution-time error) ขึ้นอยู่กับว่าเหตุการณ์เกิดเมื่อใด

บางภาษาเช่น Pascal และ COBOL ต้องการให้โปรแกรมเมอร์ประกาศอย่างชัดแจ้งถึงชนิดของตัวแปรทุกตัวที่ใช้ในโปรแกรม

ภาษาอื่น ๆ เช่น FORTRAN มีการประกาศชนิดข้อมูลอย่างโดยนัยตัวอย่างเช่น ตัวแปรซึ่งมีชื่อขึ้นต้นด้วยตัวอักษร I,J,K,L,M หรือ N มีแบบชนิดข้อมูลเป็น integer เว้นแต่ว่ามีการประกาศเป็นอย่างอื่นโดยชัดแจ้ง

ในภาษา COBOL มีส่วนเฉพาะของโปรแกรมซึ่งเรียกว่า Data division มีวัตถุประสงค์ตั้งแต่แรกคือเขียนบทนิยามของแบบชนิดข้อมูล ในภาษาอื่น ๆ ตัวอย่างเช่น ภาษา Pascal ชนิดของตัวแปรต้องมีการประกาศก่อนการใช้ครั้งแรกของตัวแปรนั้น ภาษา PL/1 มีข้อความสั่ง DECLARE สำหรับบทนิยามข้อมูลภาษา Pascal ใช้ข้อความสั่ง var ภาษา FORTRAN โปรแกรมเมอร์ใช้ข้อความสั่ง INTEGER , REAL และ DIMENSION

ขณะนี้ภาษาโปรแกำส่วนใหญ่ให้โปรแกรมเมอร์นิยามก่อน (predefind) ถึงเซตของแบบชนิดข้อมูลสำหรับใช้ในตัวแปรต่างๆ

สังเกตได้จาก Pascal มีข้อควาามสั่ง type ของมันเองทำให้โปรแกรมเมอร์ให้นิยามและให้ชื่อโครงสร้างข้อมูลบางครั้งคุณสมบัตินี้ใช้ programmer-defined name กับผลประกอบชนิด system-defined แต่แบบข้อมูลที่โปรแกรมเมอร์เป็นคนตั้งขึ้นมีมากกว่า ต่อไปนี้คือวิธีต่าง ๆซึ่งโปรแกรมเมอร์ให้นิยามตัวแปรซึ่งมีโครงสร้างข้อมูลเป็น integer , boolean และ character ในภาษา COBOL และ pascal ตัวอย่างเช่นสมมติว่าตัวแปรชนิด integer ชื่อว่า COUNT และมีค่ามากที่สุดเป็นเลขสามหลัก, ตัวแปรชนิด boolean ชื่อ SWITCH, ตัวแปรชนิด character ชื่อ BETA ภาษา COBOL เขียนดังนี้

```
DATA DIVISION.
01    COUNT PICTURE S999.
01    FLDA   PICTURE X.
      88 SWITCH    VALUE 'Y'.
01    BETA   PICTURE X.
```

ในที่นี้ S ที่อยู่ใน PICTURE clause ของชื่อ COUNT แทนค่าที่เป็นเครื่องหมายและมีความจำเป็นเพื่อให้ค่าลบเก็บในตัวแปรชนิด numeric ได้ ภาษา COBOL ไม่มีแบบชนิดข้อมูลชื่อ boolean ดังนั้นคุณสมบัติของชื่อมีเงื่อนไข (condition-name) จึงถูกนำมาใช้แทนเพื่อเกี่ยวข้องกับค่าต่าง ๆ ซึ่ง data items ถูกสมมติขึ้น หลังจากนั้น มีเงื่อนไข ตัวอย่างเช่นในที่นี้

88-level ชื่อมีเงื่อนไข SWITCH ถูกนิยามให้มีค่า boolean เป็น true เมื่อ FLDA เท่ากับ 'y' และค่า boolean เป็น false สำหรับค่าอื่น ๆ ทั้งหมดของ FLDA ดังนั้น โปรแกรมเมอร์สามารถเขียนรหัส

```
OF SWITCH THEN . . .
        ELSE . . .
```

แทนที่จะเป็น

```
IF FLDA = 'Y' THEN. . .
        ELSE. . .
```

ตัวแปรชื่อ FLDA และ BETA มี picture เป็น X แสดงว่าค่านั้นเอามาจากชุดอักขระเต็ม ตัวแปร COUNT มี picture เป็น 9 แสดงว่าด้านนี้เอามาจาก numerics

ภาษา Pascal เขียนดังนี้

```
var     count : integer;
        switch : boolean;
        beta : char;
```

จำนวนของเลขโดดใน count ถูกควบคุมโดยรูปแบบในคำสั่งงาน read และ write ส่วนตัวแปร switch สามารถกำหนดค่าเป็น true หรือเป็น false สายอักขระสามารถให้นิยามได้อย่างง่ายใน COBOL การประกาศตัวแปรชื่อ ADDRESS ให้เป็นสายอักขระที่มีความยาวเท่ากับ 25 ตัวอักขระเขียนดังนี้

```
01      ADDRESS       PICTURE X(25).
```

ชุดตัวอักษรสำหรับการได้มาของสายอักขระต่าง ๆ คือชุดอักขระเต็มสนับสนุนโดยคอมไพเลอร์การประกาศของตัวแปรชนิด string ในภาษา Pascal ดูเหมือนยุ่งยากกว่าเชียนดังนี้

```
var address : packed array [1..25] of char;
```

ในที่นี้สมาชิกแต่ละตัวของแถวลำดับเรียกว่า address คืออักขระหนึ่งตัวของสายอักขระแถวลำดับต้องถูกอัดแน่น (packed) ถ้าต้องการให้สายอักขระหมายถึงรวมทั้งหมดเป็นหนึ่งหน่วยกรณีอื่น ๆ เฉพาะอักขระตัวประกอบของสายอักขระที่อ้างถึงได้จะได้พิจารณาแถวลำดับอย่างละเอียดในบทถัดไป

แบบชนิดข้อมูลซึ่งนิยามโดยโปรแกรมเมอร์ของ Pascal ยอมให้โปรแกรมเมอร์ให้ชื่อกับโครงสร้างข้อมูลผลประกอบตัวใหม่ หลังจากนั้นกำหนดให้ชนิดใหม่กับตัวแปร

ตัวอย่างเช่น โครงสร้างข้อมูลผลประกอบซึ่งประกอบจากข้อมูลพื้นฐานชนิด char กำหนดชื่อเป็น string 25 เขียนดังนี้

```
type string25 = packed array [1..25] of char;
```

หลังจากนั้นโปรแกรมเมอร์สามารถกำหนดตัวแปรชื่อ name และ address ของชนิดข้างต้นดังนี้

```
var name, address : string25;
```

บางภาษามีตัวดำเนินการในตัว (built-in operators) สำหรับคุมแต่งตัวแปรชนิด string ภาษา PL/1 มี LENGTH, SUBSTR และ || (concatenation)
ภาษา COBOL มี STRING และ UNSTRING ภาษา Pascal ไม่มีตัวดำเนินการพื้นฐานเหล่านี้ โปรแกรมเมอร์ซึ่งจำเป็นต้องกระทำการดำเนินการสายอักขระต้องเขียนรู้ทีน(routine)ที่สมนัยหรือเอามาจากคลังของรหัส (library of the code)

บางภาษาโดยเฉพาะ SNOBOL และ UCSD Pascal เป็นภาษาเฉพาะด้านที่ทำให้โปรแกรมเมอร์สามารถจัดกระทำข้อมูลสายอักขระได้โดยง่าย

1.6 การแปลงส่งให้กับหน่วยเก็บ : จำนวนเต็ม (Mapping to storage : Integer)

คุณสมบัติอย่างหนึ่งของโครงสร้างข้อมูลตรรกะคือแต่ละชนิดมีการแปลงส่งหน่วยเก็บหรือการแทนที่เชิงกายภาพที่เป็นไปได้หลายวิธีในหัวข้อนี้จะอภิปรายทางเลือกบางอย่างสำหรับแปลงส่งจำนวนเต็มให้หน่วยเก็บ

(1)การเปลงส่งหน่วยเก็บที่เป็นไปได้วิธีหนึ่งสำหรับจำนวนเต็มเรียกว่า **รูปแบบเครื่องหมาย-และ-ขนาด(sign-and-magnitude form)**สิ่งนี้คือรูปแบบธรรมดาสำหรับการแทนที่จำนวนโดยมนุษย์เมื่อจำนวนเต็มบวกแสดงโดยเครื่องหมายบวกและสายของเลขโดดซึ่งแทนขนาดจำนวนเต็มลบแสดงโดยเครื่องหมายลบและสายของเลขโดดซึ่งแทนขนาดเครื่องหมายบวกบ่อยครั้งที่ไม่ใส่ไว้ ในการใช้จำนวนทุกวันนี้แต่ต้องเก็บมันไว้กับจำนวนสำหรับประมวลผลด้วยคอมพิวเตอร์ขนาดของจำนวน (magnitude of a number) คือค่าธรรมชาติของสายเลขโดด ทุกวันนี้ใช้โดยมนุษย์ขนาดของจำนวนคือการแทนด้วยรูปแบบฐาน 10 ของมันในหน่วยความจำของคอมพิวเตอร์เราแทนขนาดในรูปแบบฐานสอง (binary) ของมัน

มนุษย์เกือบจะตัดการใช้รูปแบบ sing-and-magnetude ออกไป ถ้าเราพัฒนาคอมพิวเตอร์อัลกอริทึมสำหรับการบวกจำนวนในรูปแบบเครื่องหมาย-และขนาด จะพบว่าเมื่อตัวถูกกระทำมีเครื่องหมายไม่เหมือนกัน การบวกจริง ๆแล้วต้องใช้การลบ ซึ่งกลายเป็นเรื่องซับซ้อน

เพื่อแก้ปัญหานี้การแปลงส่งหน่วยเก็บอีกวิธีหนึ่งเรียกว่า**การแทนที่ส่วนเติมเต็ม (complement representation)** ได้ถูกพัฒนาขึ้น

กำหนดจำนวนเต็มไม่ใช่ลบ x, x' และ R เรานิยมให้ x' เป็นส่วนเติมเต็มของ x ขึ้นอยู่กับ R หรือ R' s complement of X หรือ $X + X' = R$ ในการแทนที่ส่วนเติมเต็ม จำนวนเต็ม X เรียกว่ารูปแบบจริง (true form) และจำนวนเต็ม $X' = R-X$ เรียกว่า รูปแบบส่วนเติมเต็ม

รูปแบบส่วนเติมเต็ม (Complement forms)

ในคอมพิวเตอร์แบบทวิภาค ทางเลือกหนึ่งในทางปฏิบัติของส่วนเติมเต็มตัวคงที่ R คือกำลังของ 2

$$R = 2^N$$

ในที่นี้ N คือพิสัยของจำนวนเต็มซึ่งจะถูกแทนที่ค่าใหญ่สุดที่สามารถแทนที่ได้ในรูปแบบส่วนเติมเต็มของ 2^N คือ 2^{N-1}

(The largest value representable in 2^N s 'complement form is 2^{N-1}.)

(2)ระบบการแปลงส่งซึ่งใช้ $R = 2^N$ เรียกว่าระบบส่วนเติมเต็มของ สอง (two's complement system)

ตารางที่ 3 แสดงให้เห็นการแทนที่ของ binary sign-and magnitude และการแทนที่ของ two's-complement (โดยใช้ $R = 2^4$) ของจำนวนเต็มในพิสัย −7 ถึง +7 รูปแบบส่วนเติมเต็ม $X' = R - X$ แทนจำนวนเต็ม −X ส่วนรูปแบบจริง X แทนจำนวนเต็มบวก X

ตาราง3 Two Methods for Representation of Integer Value : Binary Sign-and-Magnitude and Two's-Complement

Integer	Binary sign-and magnitude	Two's-Complement	
-7	-0111	1001	Complement forms
-6	-1110	1010	
-5	-1101	1011	
-4	-1100	1100	
-3	-1011	1101	
-2	-1010	1110	
-1	-1001	1111	
+0	+0000	0000	True forms
+1	+0001	0001	
+2	+0010	0010	
+3	+0011	0011	
+4	+0100	0100	
+5	+0101	0101	
+6	+0110	0110	
+7	+0111	0111	

อัลกอริทึมที่ใช้กระทำการคำนวณบนจำนวนเต็มซึ่งแทนที่ในรูปแบบส่วนเติมเต็มจะสะดวกสำหรับคอมพิวเตอร์ มากกว่าอัลกอริทึมที่จัดการจำนวนเต็มซึ่งแทนที่ในรูปแบบเครื่องหมาย-ขนาดเพราะฉะนั้นถ้ามีการคำนวณมากกระทำบนตัวแปรจำนวนเต็มมีข้อแนะนำว่าให้แทนจำนวนเต็มในหน่วยเก็บโดยใช้รูปแบบส่วนเติมเต็ม

ทางเลือกที่สองในทางปฏิบัติของส่วนเติมเต็มคือให้กำหนดตัวคงที่ R ดังนี้

$$R = 2^N - 1$$

ระบบการแปลงส่งเหล่านี้เรียกว่า **ระบบส่วนเติมเต็มของหนึ่ง (one's complement system)**

ในแบบฝึกหัดท้ายบทให้สร้างการแทนที่ส่วนเติมเต็มของหนึ่ง (โดยใช้ $R = 2^4 - 1$) ของจำนวนเต็มในพิสัย -7 ถ้า +7

ระบบส่วนเติมเต็มของหนึ่งมีข้อดีหนือกว่ารูปแบบเครื่องหมาย-ขนาด เช่นเดียวกับระบบส่วนเติมเต็มของสองอัลกอริทึมคำนวณบางอย่างสำหรับรูปแบบส่วนเติมเต็มของสองมีความซับซ้อนมากกว่า รูปแบบส่วนเติมเต็มของหนึ่งส่วนรูปแบบอื่น ๆ มีความซับซ้อนมากกว่ารูปแบบส่วนเติมเต็มของหนึ่ง

โดยพื้นฐานทั้งรูปแบบเครื่องหมาย-ขนาด และรูปแบบส่วนเติมเต็มแทนจำนวนเต็มโดยหัวเรื่องและความหลากหลายบนขนาดของจำนวน

อีกทางเลือกหนึ่งคือการแปลงส่งจำนวนเต็มให้กับหน่วยเก็บโดยใช้การเข้าถึงทีละหลัก (digit-by-digit approach) ปฏิบัติราวกับว่าเลขโดดแต่ละตัวคืออักขระหนึ่งตัว

1.7 การแปลงส่งให้กับหน่วยเก็บ : ตัวอักขระ (Mapping to storage : Characters)

ทุกวันนี้มีโครงสร้างมากมายที่ใช้สำหรับการแทนที่ข้อมูลตัวอักขระ โครงสร้างการเข้ารหัสที่โดดเด่นมากที่สุด 2 ชนิดได้แก่

- รหัสสับเปลี่ยนเลขฐานสิบเข้ารหัสฐานสองแบบขยาย (เอบซีดิก) (Extended Binary Coded Decimal Interchange Code (EBCDIC))
- รหัสมาตรฐานของสหรัฐอเมริกาเพื่อการสับเปลี่ยนสารสนเทศ(แอสกี) (American standard Code for Information Interchange (ASCII))

EBCDIC เป็นรหัสชนิด 8 บิต หมายความว่าใช้ 8 บิตแทนอักขระแต่ละตัวในชุดอักขระทำให้มีจำนวน 2^8 (=256) วิธีจัดกลุ่มที่เป็นไปได้ และรหัส EGBCDIC จึงใช้กันมาก ชุดอักขระซึ่งรหัส EBCDIC สามารถสร้างได้มีทั้งตัวอักษรตัวใหญ่ และตัวเล็ก เลขโดด และตัวอักขระพิเศษจำนวนมาก

ASCII เป็นรหัสชนิด 7 บิต จึงมีการจัดกลุ่มที่เป็นไปได้ 2^7 (=128) วิธีซึ่งเป็นจำนวนตัวอักขระเพียงครึ่งหนึ่งของรหัส EBCDIC เมื่อต่อรองแล้ว (trade-off)

อักขระแต่ละตัวใช้หน่วยเก็บน้อยกว่าและสามารถส่งได้รวดเร็วกว่าในระหว่างโครงร่างการเข้ารหัสที่มีให้ใช้ ได้แก่รหัสที่มีวัตถุประสงค์เฉพาะด้านเช่น กลุ่มของรหัสฮัฟแมน (Huffman code) รหัสชนิดนี้ตัวอักขระถูกแทนที่ด้วยจำนวนบิตที่ผันแปรขึ้นอยู่กับความถี่สัมพัทธ์ของการเกิดอักขระตัวนั้นในคำศัพท์ของงาน ให้แทนที่ตัวอักขระที่เกิดขึ้นบ่อยที่สุดด้วยรูปแบบบิตสั้นกว่าและให้แทนที่ตัวอักขระที่มีความถี่น้อยกว่าด้วยรูปแบบบิตยาวกว่า ตัวอย่างแสดงให้เห็นในตารางที่ 4 ตัวอักขระ 0 แทนที่ด้วยหนึ่งบิต

TABLE 1–4 HUFFMAN CODE FOR A PARTICULAR APPLICATION

Charac-ter	Frequency of occurrence (%)	Code	Number of bits	
0	55.5	0	1	
1	6.7	1000	4	
2	4.5	1100	4	
8	3.5	10010	5	
3	3.3	10100	5	
A	3.2	10101	5	
5	3.0	10110	5	
6	2.7	11100	5	
4	2.7	11101	5	
9	2.2	11110	5	
7	1.9	100110	6	
F	1.5	101110	6	
B	1.2	111110	6	
Blank	1.1	110110	6	
D	1.0	110100	6	
E	0.9	110101	6	
Z	0.7	1011110	7	
P	0.6	1111110	7	
N	0.5	1101110	7	
u	0.4	10011110	8	
C	0.4	10011100	8	
H	0.4	10011101	8	
R	0.3	10111110	8	
M	0.3	11111110	8	
L	0.3	11111111	8	
S	0.25	11011110	8	Average character length =
I	0.20	100111110	9	0.555 × 1 + 0.112 × 4
T	0.15	110111110	9	+ 0.206 × 5 + 0.76 × 6
K	0.15	110111111	9	+ 0.018 × 7 + 0.24 × 8
Y	0.13	1001111110	10	+ 0.005 × 9 + 0.0045 × 10
X	0.12	1001111111	10	+ 0.0017 × 11 + 0.0003 × 12
G	0.10	1011111100	10	+ 0.0001 × 16
J	0.10	1011111101	10	= 2.91 bits per character
O	0.06	10111111100	11	
Q	0.03	10111111101	11	
V	0.03	10111111110	11	
W	0.03	101111111110	12	
.	0.01	1011111111110000	16	
-	|	1011111111110001	16	
?	|	1011111111110010	16	
&	|	1011111111110011	16	
/	|	1011111111110100	16	
+	|	1011111111110101	16	
<	below	1011111111110110	16	
)	0.001	1011111111110111	16	
(	|	1011111111111000	16	
%	|	1011111111111001	16	
=	|	1011111111111010	16	
#	|	1011111111111011	16	
?	|	1011111111111100	16	
'	|	1011111111111101	16	
@	|	1011111111111110	16	
	|	1011111111111111	16	

Source: J. Martin, *Computer Data-Base Organization*, 2nd ed. (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1977), Fig. 32.8.

ตัวอักขระ A แทนที่ด้วยรูปแบบ 5 บิตคือ 10101 ตัวอักขระ % แทนที่ด้วยรูปแบบ 16 บิต คือ

1011111111111001

การ trade-off เพื่อให้ผลลัพธ์ใช้หน่วยเก็บอัดแน่นคือการประมวลผลที่จำเป็นเพื่อเข้ารหัสและเพื่อให้รู้จักตัวอักขระ

คอมไพเลอร์จำนวนมากจะยอมให้โปรแกรมเมอร์กำหนดรหัสได้หลากหลายสำหรับใช้ภายในหนึ่งโปรแกรม ตัวอย่างเช่น โปรแกรม COBOL อาจจะอ่านหรือเขียนข้อมูลจากแฟ้มโดยใช้ EBCDIC บางครั้งสำหรับหน่วยขับแถบบันทึกที่เข้ากันได้ข้อมูลอื่น ๆ อาจจะอ่านหรือเขียนในรหัส ASCII ซึ่งน่าจะเป็นอินพุต/เอาต์พุตบนอุปกรณ์ปลายทาง ข้อมูลอื่น ๆ อาจแทนที่โดยใช้รหัสแสดงภายในถ้ามันไม่ได้ถูกใช้ร่วมกันด้วยการประมวลผลบนอุปกรณ์อื่น ๆ สมาชิกข้อมูลเฉพาะด้านสามารถถูกเข้ารหัสโดยใช้เพียงหนึ่งโครงร่างเท่านั้น

เมื่อโครงร่างการเข้ารหัสตัวอักขระ (a character-encoding scheme) ถูกนำมาใช้แทนจำนวนเต็ม เครื่องหมายบวกหรือลบต้องถูกเก็บพร้อมกับการแทนที่เพื่อแสดงว่าจำนวนเต็มเป็นบวกหรือเป็นลบ โดยทั่วไปเครื่องหมายนี้เก็บทางขวามือของเลขโดดที่มีอันดับต่ำของจำนวน

(Generally this sign is stored to the right of the low-order digit of the number.)

คอมไพเลอร์จำนวนมากเสนอทางเลือกอื่น ๆ และความหลากหลายบนรหัสเหล่านี้สำหรับการแทนที่จำนวนเต็ม บางทีทางเลือกที่ใช้แพร่หลายมากที่สุดคือ **รูปแบบทศนิยมอัดแน่น (packed-decimal format)** โครงร่างนี้แทนข้อมูลตัวเลขกระชับโดยเก็บเลขโดด 2 ตัวต่อ 8 บิตไม่ใช่เลขโดดหนึ่งตัวต่อ 8 บิต (EBCDIC) ทางขวามือ 8 บิต ไม่ใช่เลขโดดอันดับต่ำสุดของจำนวนแต่เป็นเครื่องหมายของจำนวน ตารางที่ 5 แสดงให้เห็นการแทนที่ของจำนวนเต็ม +903 และ -903 โดยใช้โครงร่างเชิงตัวอักขระหลายแบบ

ตารางที่ 5 ตัวอย่างของ Character-Oriented Numberic Representation

Code	+903
EBCDIC	11111001 11110000 11110011 01001110
ASCII	0111001 0110000 0110011 0101011
Packed decimal	10010000 00111100
Code	-903
EBCDIC	11111001 11110000 11110011 01100000
ASCII	0111001 0110000 0110011 0101101
Packed decimal	10010000 00111101

1.8 การแปลงส่งไปยังหน่วยเก็บ : สายอักขระ (Mapping to Storage : Strings)

ขณะนี้เราอภิปรายว่าตัวอักขระแต่ละตัวถูกแทนที่อย่างไรให้พิจารณาทางเลือกต่าง ๆ สำหรับการแทนที่สายของตัวอักขระการอภิปรายในหัวข้อนี้จะจำกัดการแทนที่ของสายอักขระในเนื้อที่ติดกัน (contiguous space);

หน่วยเก็บของสายอักขระความยาวเท่ากับ N จะใช้เนื้อที่ประชิดกันเชิงกายภาพสำหรับอักขระ N ตัว เพื่ออธิบายการแปลงส่งหน่วยเก็บเนื้อที่ติดกันของสายอักขระจำเป็นที่จะต้องแสดงทั้งตำแหน่งที่เนื้อที่ของสายอักขระเริ่มต้นและตำแหน่งที่สายอักขระ

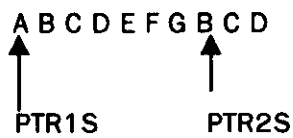
ในข้อตกลงแต่ละข้อต่อไปนี้ต้องใช้ตัวแปรช่วยอย่างน้อยที่สุดหนึ่งตัวของข้อมูลชนิดพอยน์เตอร์ (pointer) รายละเอียดมากกว่านี้จะกล่าวถึงภายหลังเกี่ยวกับแบบข้อมูลชนิด pointer ขณะนี้พูดเพียงว่าค่าของตัวแปรของข้อมูลชนิดพอยน์เตอร์คือเลขที่อยู่ (address) กล่าวคือพอยน์เตอร์แสดงถึงตำแหน่งหน่วยเก็บมีวิธีง่าย ๆ อย่างน้อย 3 วิธี เพื่ออธิบายตัวแปรสายอักขระเก็บในเนื้อที่ติดกัน เราจะใช้ตัวอย่างสายอักขระสองชุดดังนี้

ให้ STRING1 = 'ABCDEFG'
และ STRING2 = 'BCD'

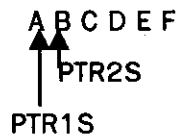
1) เก็บตารางของสารสนเทศข้างล่างนี้สำหรับตัวแปรสายอักขระแต่ละชุด : ชื่อของมัน , เลขที่อยู่เริ่มต้นของมัน, ความยาวของมัน
ตัวอย่างเช่น

NAME	START	LENGTH
STRING1	PTR1S	7
STRING2	PTR2S	3

รูปแบบหน่วยเก็บที่สมนัยเป็นดังนี้ :



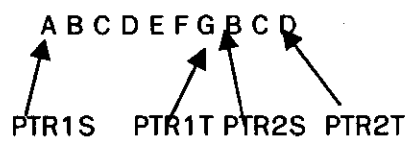
หรือ การซ้อนแทนสายอักขระ(Overlaying the strings);



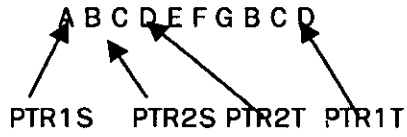
2) เก็บตารางสารสนเทศข้างล่างนี้สำหรับตัวแปรสายอักขระแต่ละชุด : ชื่อของมัน , เลขที่อยู่เริ่มต้นของมัน เลขที่อยู่สิ้นสุดของมัน
ตัวอย่างเช่น

NAME	START	TERM
STRING1	PTR1S	PTR1T
STRING2	PTR2S	PTR2T

รูปแบบหน่วยเก็บที่สมนัยเป็นดังนี้



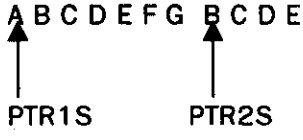
หรือการซ้อนแทนสายอักขระ



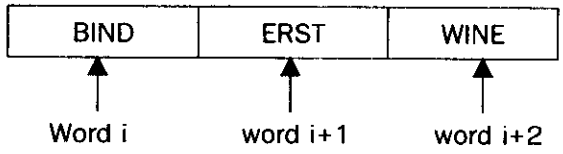
3) เก็บตารางของสารสนเทศข้างล่างนี้สำหรับตัวแปรสายอักขระแต่ละชุด : ชื่อของมัน และเลขที่อยู่เริ่มต้นของมัน

NAME	START
STRING1	PTR1S
STRING2	PTR2S

รูปแบบหน่วยเก็บที่สมนัยจะเป็นดังนี้



การพิจารณาอีกวิธีหนึ่งในการแทนที่ของสายอักขระในหน่วยเก็บคือไม่ว่าจะเป็นการแทนที่แบบอัดแน่นหรือแบบขยายออกก็ตาม (packed or unpacked)
กำหนดโครงร่างการเข้ารหัสเฉพาะให้และสมมติว่าเป็นเครื่องชนิดคำ
(a word-oriented machine)
การแทนที่แบบอัดแน่นของสายอักขระ ประกอบด้วยการใส่รหัสสืบเนื่องแต่ละตัวสำหรับตัวอักขระในสายในหน่วยเก็บตัวสืบเนื่อง ด้วยจำนวนตัวอักขระมากเท่าที่เป็นไปได้ในคำที่กำหนด
(The packed representation of a string consists of placing each of the successive codes for the characters in the string in successive storage words, with as many of the characters as possible in any given word.)
ตัวอย่างเช่นสายอักขระ 'BINDERSTWINE' การอัดแน่นสี่ตัวอักขระต่อคำกำหนดในรูปที่2



รูปที่2

โดยทั่วไปกรณีอักขระ K ตัวต่อหนึ่งคำ จำนวนของคำที่ต้องใช้เก็บสายอักขระ S เท่ากับ

$\lceil \text{LENGTH}(S) \rceil$, เมื่อ $\lceil\ \rceil$ คือฟังก์ชัน ceiling *

บางครั้งรหัสตัวอักขระใส่พอดีใน 1 คำ (word) ถ้าเป็น 32 บิตต่อคำ หรือ 16 บิตต่อคำดังนั้นรหัส EBCDIC ชนิด 8 บิตต่อตัวหนึ่งอักขระจึงเหมาะพอดีมากแต่รหัส ASCII ชนิด 7 บิตต่อหนึ่งตัวอักขระจึงต้องยอมให้อักขระสองตัวใน 16-บิตต่อคำกับบิตที่เหลืออีกสองบิตซึ่งไม่ได้ถูกใช้หรืออักขระสี่ตัวใน 32 บิตต่อคำกับบิตที่เหลือไม่ได้อีก 4 บิต

* ฟังก์ชัน ceiling ให้ผลลัพธ์คือจำนวนเต็มตัวแรกที่มีค่ามากกว่าหรือเท่ากับตัวถูกกระทำ ตัวอย่างเช่น $\lceil 3.16 \rceil = 4$ หรือ $\lceil 3 \rceil$ เท่ากับ 3

การแทนที่แบบขยายออกของสายอักขระ ประกอบด้วยการเก็บรหัสหนึ่งตัวอักขระในหนึ่งคำสำหรับตัวอักขระสืบเนื่องแต่ละตัว
(The unpacked representation for a string consists of storing one to a word the character codes for each of the succesive character)

อักขระ 'BINDERSTWINE'เก็บในรูปแบบขยายถูกกำหนดให้ในรูปที่3

B	I	N	D	E	R	S	T	W	I	N	E

จำนวนคำที่ต้องใช้สำหรับเก็บสายอักขระ S ในรูปแบบขยายเท่ากับ LENGTH(S) การแทนที่สายอักขระทั้งสองชนิดนี้ แบบใดดีกว่า ? ทั้งหมดขึ้นอยู่กับวัตถุประสงค์ในการทำงาน การแทนที่แบบอัดแน่นเหมาะที่สุดสำหรับสถานะการณ์ซึ่งการใช้ประโยชน์หน่วยเก็บต่ำสุดเป็นเรื่องสำคัญที่สุดแต่การดำเนินการการแทนที่แบบอัดแน่นจะโดยปกติจะช้ากว่าการแทนที่แบบขยาย

การแทนที่แบบขยายไม่ต้องมีการเลื่อน (shifting), การปิดบัง (masking), หรือการแปลงผัน(conversion) เมื่อมีการคลุมแต่ง contents ของคำ

1.9 การเลือกการแปลงส่งที่เหมาะสม (Selection of Appropriate Mappings)

หัวข้อที่ผ่านมาคงจะทำให้เราเชื่อมั่นได้ว่าที่จริงมีการแปลงส่งที่เป็นไปได้หลายวิธีของตัวอย่างโครงสร้างข้อมูลให้กับหน่วยเก็บ การที่จะบอกว่าทางเลือกใดการแปลงส่งที่เหมาะสมที่สุดที่จะนำมาใช้ขึ้นอยู่กับว่า

(1) ตัวแปรจะมีการคลุมแต่งอย่างไร (How the variable will be manipuated)
(2) พิสัยของค่าต่าง ๆ ซึ่งตัวแปรจะมี (The range of values that the variable will have,)
(3) คุณสมบัติของคอมพิวเตอณ์และหน่วยความจำซึ่งจะประมวลผลและเก็บตัวแปร (The characteristics of the computer and memory that will process and store the variable.)

การเลือกรูปแบบส่วนเติมเต็มของหนึ่งและรูปแบบส่วนเติมเต็มของสอง โดยนัยคืออัลกอริทึมการคำนวณที่ใช้สำหรับคอมพิวเตอร์ซับซ้อนน้อยกว่าการใช้การแทนที่แบบเครื่องหมาย-และ-ขนาด

อัลกอริทึมให้คำนวณกับจำนวนเต็มแทนที่ในรหัสทีละหนึ่งหลัก (digit-by-digit code) จะมีความซับซ้อนมากกว่าอัลกอริทึมที่แทนจำนวนเต็มแบบเครื่องหมาย-และ-ขนาดจริง ๆ แล้วคอมไพเลอร์จำนวนมากแปลงผันจำนวนจากรูปแบบทีละหลักให้เป็นการแทนที่แบบเครื่องหมาย-และ-ขนาดหรือการแทนที่แบบส่วนเติมเต็มก่อนทำการคำนวณถ้าการคำนวณด้วยการแทนที่แบบทีละหนึ่งหลักมีประสิทธิภาพน้อยกว่ามากทำไมการคำนวณเช่นนี้ จึงยังคงมีอยู่ ? เหตุผลหลักสำหรับความสำคัญของโครงร่างการเข้ารหัสเช่น EBCDIC และ ASCII คือความสามารถของมันที่จะแทนตัวอักขระไม่ใช้ตัวเลขได้นอกจากนี้แล้ว ถ้าข้อมูลตัวเลขถูกแสดงผลบนจอภาพของ เทอร์มินอลหรือพิมพ์ออกบนรายงานเพื่อให้มนุษย์นำไปใช้ข้อมูลเล่านี้ต้องถูกแทนที่ในรหัสที่สามารถแสดงผลได้ ทำไมจึงมีโครงร่างการเชิงรหัสมากมาย ? (Why are there so many coding schemes?.) เหตุผลข้อที่หนึ่งคือ โครงร่างเข้ารหัสเหล่านี้ได้มีการพัฒนาขึ้นมาโดยหลายกลุ่มที่แตกต่างกัน สำหรับวัตถุประสงค์หลากหลาย

ตัวอย่างเช่น คอมพิวเตอร์ IBM ใช้ 32 บิตต่อหนึ่งคำ รหัสตัวอักขระชนิด 8 บิต (EBCDIC) จะสะดวกอย่างมากสำหรับการแทนที่ข้อมูลที่มีขนาดเช่นนั้น

คอมพิวเตอร์ของ DEC system 10 มี 36-บิตต่อหนึ่งคำรหัสตัวอักขระชนิด 7 บิต (เช่น ASCII) จึงสะดวกมากเมื่อแทนอักขระ 5 ตัวต่อหนึ่งคำและมีอีกหนึ่งช่องสำหรับบิตเครื่องหมาย

คอมพิวเตอร์ CDC ขนาดใหญ่ เช่น CYBER 175และ CDC7600 มี 60 บิตต่อหนึ่งคำ เครื่องเหล่านี้จึงเหมาะสมอย่างมากสำหรับการคำนวณทางวิทยาศาสตร์เพราะว่าความถูกต้องมากกว่ากระทำการสัมพันธ์กับ ความยาวของคำที่ยาวกว่า รหัสชนิด 6-บิตต่อหนึ่งตัวอักขระ (เช่น BCD) สะดวกอย่างมากที่จะแทนอักขระสองตัวต่อหนึ่งคำ

รหัสประหยัดเนื้อที่บางชนิดนำมาใช้เพื่อที่หน่วยความจึงมีจำกัดหรือความสามาคพของการสื่อสารถูกจำกัดอย่างมากทุกวันี้รหัสที่ใช้กันแพร่หลายมากที่สุดสำหรับแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์คือ EBCDIC และ ASCII ผู้ขายคอมไพเลอร์จำนวนมากเสนอทางเลือกทั้งคู่ในคอมไพเลอร์ของเขานอกเหนือจากนี้ผู้ขายเทอร์มินอลเกือบทั้งหมดเสนอขายอุปกรณ์กับตัวประมวลผลซึ่งยอมรับหนึ่งรหัสหรือทั้งคู่ของรหัสสองชนิดนี้

บทสรุป

บทนี้เริ่มต้นด้วยการทบทวนความสำคัญของข้อมูลกับกระบวนการทำการตัดสินใจขององค์กร มีแนวทาง 4 ข้อซึ่งเป็นเป้าหมายสำหรับระบบจัดการข้อมูล แนวคิดของโครงสร้างข้อมูล หลังจากนั้นแนะนำโครงสร้างข้อมูลหลากหลายซึ่งได้ถูกแบ่งประเภทซึ่งจะอภิปรายในหนังสือเล่มนี้ Integer, boolean และ characters เป็นตัวอย่างของแบบชนิดข้อมูลเก่าแก่ซึ่ง

แบบชนิดข้อมูลเหล่านี้ผู้อ่านน่าจะมีความคุ้นเคยมาแล้ว แบบชนิดข้อมูลเชิงประกอบเรียกว่า string และตัวดำเนินการพื้นฐานของมัน (length, concatenation และ substring) ได้ยกขึ้นมาแนะนำ

ความคล่องตัวสำหรับการประกาศของ integers, boolenas, characters และ strings ในภาษา COBOL และ Pascal ได้นำมาทบทวนสองภาษานี้จะนำมาใช้ตลอดในหนังสือเพื่อแสดงให้เห็นชนิดต่าง ๆ ของสิ่งสนับสนุนซึ่งจัดให้กับโปรแกรมเมอร์สำหรับบทนิยามโครงสร้างข้อมูลและการคลุมแต่ง

มีทางเลือกหลายวิธีเพื่อแปลงส่งโครงสร้างข้อมูลเชิงตรรกะให้กับหน่วยเก็บคอมพิวเตอร์เชิงกายภาพ การอภิปรายการแทนที่เครื่องหมาย-ขนาด และส่วนเติมเต็มของ integers

EBCDIC และ ASCII นำมาใช้เป็นเทคนิคสำหรับการแปลงส่งข้อมูลตัวอักขระไปยังหน่วยเก็บจากนั้นขยายให้ใช้การแปลงส่งเชิงตัวอักขระของ integer ไปยังหน่วยเก็บ

สุดท้ายแนะนำการแทนที่เลขฐานสิบอัดแน่นของจำนวนเต็มการแปลงส่งของสายอักขระไปยังหน่วยเก็บเกี่ยวข้องกับทั้งสองสิ่งนี้

(1) การแทนที่ของอักขระแต่ละตัวของสายอักขระและ

(2) การแทนที่ของการรวมกันของตัวอักขระประกอบขึ้นเป็นสายอักขระ

ทางเลือกสำหรับการเก็บสารสนเทศจำเป็นต้องหาสัญลักษณ์เริ่มต้นและสัญลักษณ์สิ้นสุดของสายอักขระ จากนั้นทำให้เห็นความแตกต่างระหว่างการแทนที่สายอักขระอัดแน่นและแบบขยาย

การเลือกโครงร่างการแปลงส่งที่เหมาะสมที่สุดสำหรับโครงสร้างข้อมูลใด ๆ ก็ตามควรจะขึ้นอยู่กับการคลุมแต่งที่กระทำบนข้อมูลนั้น คุณสมบัติของค่าต่าง ๆ ซึ่งโครงสร้างข้อมูลจะต้องมี และคุณสมบัติของคอมพิวเตอร์และหน่วยความจำซึ่งจะประมวลผลและเก็บโครงสร้างข้อมูลนั้นมีข้อแนะนำบางอย่างของพื้นฐานการ trade-off ระหว่างการแปลงส่งโครงสร้างข้อมูลซึ่งมีความสำคัญสำหรับระบบสารสนเทศ เพื่อให้สามารถออกแบบสนับสนุนที่มีประสิทธิภาพสำหรับแหล่งข้อมูลของระบบสารสนเทศ เราต้องมีความเข้าใจว่าข้อมูลถูกแทนที่และมีการจัดการอย่างไร

แบบชนิดข้อมูลซึ่งใช้ในโปรแกรมช่วยบอกให้ทราบว่าหน่วยเก็บต้องการอะไรของโปรแกรมนั้น การประมวลผลที่ต้องการของมันคืออะไร ต้องการอินพุต-เอาต์พุตอะไรเวลาโต้ตอบจะเป็นอะไรและอื่น ๆ

โครงสร้างข้อมูลซับซ้อนมีให้ใช้แต่แรกเพราะว่าแบบชนิดแตกต่างกันถูกนำมาใช้ให้วิธีแตกต่างกัน โครงสร้างข้อมูลจริง ๆ แล้วใช้นิยามเส้นทางเข้าถึงซึ่งถูกสนับสนุนระหว่างสมาชิกข้อมูล

การเปิดตัวของหนังสือเล่มนี้คือให้มีความเข้าใจว่าโครงสร้างข้อมูลชนิดต่าง ๆ นำมาใช้ในการแก้ปัญหาอย่างมีประสิทธิภาพและมีประสิทธิผลได้อย่างไร

แบบฝึกหัด

1. จงเขียนรายการของการจัดกระทำข้อมูลด้านต่าง ๆซึ่งจำเป็นต้องมีในระบบสารสนเทศ ฟังก์ชันเหล่านี้ ชุดใดซึ่งดูเหมือนว่าราคาแพงที่สุด จงบอกเหตุผล
2. จงยกตัวอย่างค่าจำนวนเต็ม 5 ชุด
3. จงยกตัวอย่างของค่าแบบบลู (boolean values)
4. จงหาผลลัพธ์ของการดำเนินงานแต่ละชุดข้างล่างนี้เมื่อตัวแปรแบบบลู FIRST มีค่าเป็น true, ตัวแปร SECOND มีค่าเป็น false และตัวแปร THIRD มีค่าเป็นtrue
   (a) not FIRST
   (b) FIRST and SECOND
   (c) FIRST and THIRD
   (d) FIRST or SECOND
   (e) FIRST or THIRD
   (f) not FIRST or THIRD
   (g) not(FIRST or THIRD)
   (h) not(first and SECOND)
   (i) not(FIRST and SECOND)
   (j) FIRST and not SECOND
   (k) not(FIRST and not SECOND)

5 . จงหาผลลัพธ์ของการดำเนินงานแต่ละชุดข้างล่างนี้เมื่อตัวแปรสายอักขระ S1 มีค่าเป็น 'PIE' ตัวแปร S2 มีค่าเป็น 'MAGIC' และตัวแปร S3 มีค่าเป็น ' WHEEL'
   (a) LENGTH(S2)
   (b) LENGTH(S3)
   (c) CONCAT(S2,S1)
   (d) SUBSTR(S3,4,3)
   (e) CONCAT(SUBSTR(S2,1,3),S1)
   (f) CONCAT(SUBSTR(S2,1,3),S3)
   (g) SUBSTR(CONCAT(S1,S2),1,LENGTH(S1))
   (h) SUBSTR(CONCAT(S2,S1),LENGTH(S2)+1,LENGTH(S1))
   (i) INSERT(S1,SUBSTR(S3,6,1),3)
   (j) DELETE(S2,2,2)
   (k) INSERT(S1,DELETE(S3,1,3),1)

6. จงสร้างการแทนที่ส่วนเติมเต็มของหนึ่ง (ให้ R = 2^4-1 ของจำนวนเต็มในพิสัย -7 ถึง +7

(Construct the one's-complement representation(with R = 2^4-1) of the integers in the range -7 to +7.)

7. จงหาค่าต่ำสุดของ R ซึ่งจำเป็นต้องแทนที่ส่วนเติมเต็มของ R ค่าระหว่าง

(What is the minimum value of R needed to represent the R's complement of values

between :)

(a) -10 และ +10

(b) -100 และ+100

8. จงหาค่าต่ำสุดของ R ซึ่งจำเป็นต้องใช้แทนที่ค่าส่วนเติมเต็มของสองของค่าระหว่าง

(a) -10 และ+10

(b) -100 และ+100

9. จะมีกี่บิตที่จำเป็นเพื่อใช้แทนค่าในรูปแบบส่วนเติมเต็มของสองโดยที่ R = 2^N

10. จะมีกี่บิตที่ซึ่งจำเป็นเพื่อใช้แทนค่าในรูปแบบส่วนเติมเต็มของ 1 โดยที่ R = 2^N-1

11. จงหาค่าใหญ่ที่สุดและค่าเล็กที่สุดซึ่งสามารถแทนในรูปแบบส่วนเติมเต็มโดยที่

(a) R = 32

(b) R = 256

(c) R = R = 2^N

12. จงหาค่าใหญ่ที่สุดและค่าเล็กที่สุดซึ่งสามารถแทนในรูปแบบส่วนเติมเต็มของ 1 โดยที่

(a) R = 31

(b) R = 255

(c) R= R = 2^N-1

13. เทคนิคอะไรที่นำมาใช้แปลงส่งจำนวนเต็มไปยังหน่วยเก็บโดยคอมไพเลอร์ที่มีให้เราใช้ได้

14. เทคนิคอะไรซึ่งนำมาใช้แปลงส่งตัวอักขระไปยังหน่วยเก็บโดยคอมไพเลอร์ที่มีให้เราใช้ได้

15. จงเขียนผังรูปแสดงรหัสตัวอักขระ EBCDIC และรหัส ASCII

16. จงเขียนการแทนที่ EBCDIC ของชื่อและเลขที่อยู่ของนักศึกษา

17. จงเขียนการแทนที่ ASCII ของชื่อและเลขที่อยู่ของนักศึกษา

18. จงแทนที่เลขที่ เลขประกันสังคมของท่าน หรือเลขแสดงสิ่งอื่นโดยใช้ EBCDIC, ASCII และ packed decimal

19. จงใช้การแทนที่หน่วยเก็บที่เป็นไปได้อย่างน้อยที่สุด 3 ชุดของค่าสายอักขระต่อไปนี้

'MAGPIE' , 'PIE' , และ 'MAGENTA'

20. อะไรคือ trade-offs พื้นฐานที่จะต้องพิจารณาในการชั่งแทนการแทนที่สายอักขระในหน่วยเก็บ

21. อะไรคือ trade-offs พื้นฐานซึ่งต้องพิจารณาในการอัดแน่นการแทนที่สายอักขระในหน่วยเก็บ

22. อะไรคือ trade-off พื้นฐานซึ่งจะต้องพิจารณาในการใช้

(a) ตัวแสดงความยาว (a length indicator) เพื่อจบการแทนที่สายอักขระ

(b) พอยน์เตอร์ไปจบการแทนที่สายอักขระ

(c) เครื่องหมายจบสายอักขระ เพื่อจบการแทนที่สายอักขระ

23. จงบอกว่าอุปกรณ์หน่วยเก็บรองและอุปกรณ์หน่วยเก็บหลักชนิดใดซึ่งมีให้ใช้ในสิ่งอำนวยความสะดวกคอมพิวเตอร์ของท่าน ความจุของมันคืออะไร ค่าใช้จ่ายสัมพัทธ์ของการใช้คืออะไร

24. จงหาชุดอักขระเต็ม (full character set) ที่ใช้โดยคอมไพเลอร์ Pascal (หรือ COBOL หรือ FORTRAN หรือ PL/1 หรือ . . .)

25. สายอักขระอินพุตสำหรับคอมไพเลอร์ Pascal จะจัดรูปแบบอย่างไร (Find out how to format character string input for your local Pascal compiler.)

26. จงเขียนโปรแกรมอ่านและพิมพ์สายอักขระ 5 ชุดแต่ละชุดความยาวเท่ากับ 10 (Write a program that reads and writes five character strings, each of length 10.)

27. จงดัดแปรโปรแกรมอ่านและพิมพ์สายอักขระ N ชุดแต่ละชุดไม่ทราบความยาว อะไรคือสิ่งที่ต้องเป็นอินพุตให้กับโปรแกรม (Modify your program to read and write N character strings, each of some unknown length. What are the required inputs to the program?)

28. จงเขียนโปรแกรมทำให้เกิดผลในทางปฏิบัติของตัวดำเนินการ LENGTH, CONCAT, และSUBSTR (Write a program that implements the LENGTH, CONCAT, and SUBSTR operators.)

29.จงขยายโปรแกรมข้อ 28 เพื่อทำให้เกิดผลในทางปฏิบัติกับตัวดำเนินการ INSERT และ DELETE