ทุนส่งเสริม
การผลิตตำรา

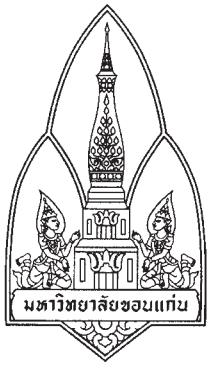
DATA STRUCTURES AND ALGORITHMS

โครงสร้างข้อมูลและขั้นตอนวิธี

COMPUTER SCIENCE

พศ.ดร.อุรฉัตร โคแก้ว
Asst.Prof.Urachart Kokaew,Ph.D.

DATA STRUCTURES AND ALGORITHMS โครงสร้างข้อมูลและขั้นตอนวิธี
พศ.ดร.อุรฉัตร โคแก้ว



โครงสร้างข้อมูลและขั้นตอนวิธี

ผู้ช่วยศาสตราจารย์ ดร.อุรฉัตร โคแก้ว

คณะวิทยาศาสตร์ มหาวิทยาลัยขอนแก่น

ได้รับทุนสนับสนุนการผลิตตำรา มหาวิทยาลัยขอนแก่น

ลำดับที่ 176

จัดพิมพ์โดยมหาวิทยาลัยขอนแก่น พ.ศ. 2563

ISBN 978-616-438-514-6

อุรฉัตร โคแก้ว.
โครงสร้างข้อมูลและขั้นตอนวิธี / อุรฉัตร โคแก้ว. -- พิมพ์ครั้งที่ 1. -- ขอนแก่น : คณะวิทยาศาสตร์ มหาวิทยาลัยขอนแก่น, 2563.
280 หน้า : ภาพประกอบ
1. โครงสร้างข้อมูล (วิทยาการคอมพิวเตอร์). 2. การจัดแฟ้มข้อมูล (วิทยาการคอมพิวเตอร์). (1) มหาวิทยาลัยขอนแก่น. คณะวิทยาศาสตร์. (2) ชื่อเรื่อง.

QA76.9.D35 อ848
ISBN 978-616-438-514-6

พิมพ์ครั้งที่ 1
จำนวน 200 เล่ม
จัดทำโดย ศูนย์นวัตกรรมการเรียนการสอน มหาวิทยาลัยขอนแก่น
พิมพ์ที่โรงพิมพ์มหาวิทยาลัยขอนแก่น พ.ศ. 2563
ราคา 280 บาท

คำนำ

โครงสร้างข้อมูลและขั้นตอนวิธีเป็นองค์ความรู้พื้นฐานสำคัญของวิทยาการคอมพิวเตอร์ วิศวกรรมคอมพิวเตอร์ และอีกหลายสาขาที่เกี่ยวข้องกับการจัดเก็บข้อมูลให้มีระบบ เพื่อเป็นประโยชน์สำหรับผู้นำไปใช้สามารถเลือกโครงสร้างข้อมูลและขั้นตอนวิธีที่มีประสิทธิภาพตามบริบทที่ต้องการ ในการเลือกโครงสร้างข้อมูลที่เหมาะสมจะทำให้ผู้พัฒนาโปรแกรมสามารถเลือกใช้ขั้นตอนวิธีที่มีประสิทธิภาพไปพร้อมกันได้ โดยส่วนมากการเลือกโครงสร้างข้อมูลจะเริ่มต้นจากการเลือกแบบชนิดข้อมูลนามธรรม โครงสร้างข้อมูลที่ออกแบบเป็นอย่างดีที่จะสามารถรองรับการประมวลผลให้ใช้ทรัพยากรที่น้อยที่สุดเท่าที่จะเป็นไปได้ ทั้งในแง่ของหน่วยความจำและเวลาที่ใช้ในการประมวลผล เนื้อหาในเล่มประกอบด้วยทั้งหมด 6 บท ได้แก่ บทที่ 1 แนะนำโครงสร้างข้อมูลและขั้นตอนวิธี บทที่ 2 อธิบายถึงโครงสร้างข้อมูลเชิงเส้นซึ่งเป็นเรื่องเกี่ยวกับแถวลำดับ สแตก คิว และลิงค์ลิสต์ ส่วนเรื่องเกี่ยวกับโครงสร้างข้อมูลชนิดไม่ใช่เชิงเส้น อยู่ในบทที่ 3 เรื่องทรี และ บทที่ 4 เรื่องกราฟ สำหรับการนำโครงสร้างข้อมูลในบทดังกล่าวมาประยุกต์ในขั้นตอนวิธีได้อธิบายใน บทที่ 5 เรื่องขั้นตอนการเรียงลำดับข้อมูล และ บทที่ 6 เรื่องขั้นตอนการค้นหาข้อมูล

หนังสือเล่มนี้ผู้เรียบเรียงขอขอบพระคุณผู้มีพระคุณทุกท่านที่เกี่ยวข้อง ได้แก่ โครงการทุนสนับสนุนการผลิตตำรามหาวิทยาลัยขอนแก่นที่ให้โอกาสได้เรียบเรียงความรู้จากประสบการณ์ในการสอนวิชาโครงสร้างข้อมูล และ วิชาการวิเคราะห์และออกแบบขั้นตอนวิธี โดยได้จากการเตรียมการสอนและปรับปรุงเนื้อหาวิชาอย่างต่อเนื่องที่ได้รับการถ่ายทอดมาจาก ผศ.ไกรสร วงศ์พราวมาศ อาจารย์ผู้มีพระคุณที่ให้ความรู้และโอกาสในการสอนวิชานี้จนเรียบเรียงเนื้อหาออกมาได้ ซึ่งเหมาะสำหรับ นักเรียน นักศึกษา และผู้สนใจทั่วไป หากมีข้อบกพร่องประการใดหรือท่านใดมีความคิดเห็นเกี่ยวกับหนังสือเล่มนี้ ขอน้อมรับคำติชมและกรุณาแจ้งให้ทราบเพื่อจะได้ปรับปรุงแก้ไขในโอกาสต่อไป

ผศ. ดร.อุรฉัตร โคแก้ว
สาขาวิชาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ มหาวิทยาลัยขอนแก่น
E-mail:urachart@kku.ac.th

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

บทที่ 1 แนะนำโครงสร้างข้อมูลและขั้นตอนวิธี

วัตถุประสงค์การเรียนรู้

แนะนำโครงสร้างข้อมูลและขั้นตอนวิธี

1. ความหมายของโครงสร้างข้อมูลและการจำแนกประเภทโครงสร้างข้อมูล
2. ตัวอย่างการจัดเก็บข้อมูลตามการใช้งานได้หลากหลายลักษณะ
3. ลักษณะของขั้นตอนวิธีโดยใช้ผังงานและรหัสเทียมในการแก้ปัญหา

การเลือกใช้โครงสร้างข้อมูลหรือการจัดการข้อมูลในรูปแบบที่เหมาะสมในการแก้ปัญหาทางคอมพิวเตอร์มีความสำคัญ เนื่องจากมีผลต่อการใช้ทรัพยากรอย่างมีประสิทธิภาพ เพื่อนำไปใช้ในกระบวนการแก้ปัญหาที่มีขั้นตอนวิธีอย่างมีประสิทธิภาพ ซึ่งจำเป็นต้องวางแผนอย่างเป็นระบบ เพื่อให้ได้ขั้นตอนที่สามารถตัดทอนส่วนซ้ำซ้อนเกินความจำเป็น หรือเพิ่มเติมขั้นตอนใหม่เข้าไปได้

1.1 นิยามโครงสร้างข้อมูล

โครงสร้างข้อมูล (Data Structures) หมายถึงรูปแบบการจัดระเบียบข้อมูลที่มีความสัมพันธ์กันเป็นชุดของข้อมูลที่มีขั้นตอนวิธี (Algorithm) ในการจัดการข้อมูล

โครงสร้างข้อมูลสามารถจำแนกได้ตามลักษณะจัดการและจัดเก็บ 2 กลุ่ม ได้แก่

1. โครงสร้างข้อมูลทางกายภาพ (Physical Data Structure) หมายถึงชนิดข้อมูลที่สามารถเก็บข้อมูลที่เป็นข้อมูลทั่วไปหรือข้อมูลพื้นฐานแบ่งเป็น 2 ประเภท ได้แก่

1.1 ข้อมูลเบื้องต้นหรือกลุ่มของข้อมูลปฐม (Primitive Data Type) เป็นข้อมูลหนึ่งจำนวนหรือมากกว่าหนึ่งจำนวน โดยข้อมูลปฐมแต่ละตัวจะอยู่ในฟิลด์ (Field) เช่น จำนวนเต็ม (จำนวนเต็มบวกจำนวนเต็มลบจำนวนเต็มศูนย์) จำนวนจริง (มีจุดทศนิยม) และตัวอักขระ

1.2 ข้อมูลแบบโครงสร้าง (Structure Data Type) คือกลุ่มของข้อมูลที่มีความสัมพันธ์กันโดยข้อมูลเหล่านี้อาจมีชนิดข้อมูลที่แตกต่างกัน มีชื่อที่ใช้อ้างถึงข้อมูลแต่ละตัวต่างกัน แต่ข้อมูลกลุ่มนี้จะอยู่ภายใต้ชื่อกลุ่มข้อมูลเดียวกัน ได้แก่ ระเบียนข้อมูล แฟ้มข้อมูล

2. โครงสร้างข้อมูลทางตรรกะ (Logical Data Structure) การจัดเก็บข้อมูลและความสัมพันธ์ต่างๆ ของข้อมูลภายใน และได้มีการประมวลมาแล้วซึ่งบอกถึงความสัมพันธ์และความเกี่ยวโยงกัน แบ่งเป็น

2.1 โครงสร้างข้อมูลเชิงเส้น (Linear Data Structure) คือโครงสร้างข้อมูลที่มีสมาชิกเรียงกันในลักษณะเป็นลำดับ ได้แก่ แถวลำดับ (Array) ลิสต์เชิงเส้น (Linear list) คิว (Queue) สแตก (Stack)

2.2 โครงสร้างข้อมูลแบบไม่เชิงเส้น (Non-linear Data Structure) ได้แก่ ทรี (Tree) กราฟ (Graph)

ตามปกติแล้วโปรแกรมคอมพิวเตอร์ทำงานโดยอาศัยข้อมูลในลักษณะที่หลากหลาย การที่จะให้โปรแกรมคอมพิวเตอร์ทำงานได้อย่างสมบูรณ์จึงจำเป็นที่จะต้องทำความเข้าใจเกี่ยวกับความสัมพันธ์เชิงโครงสร้างที่มีอยู่ของข้อมูล รวมทั้งเทคนิคการแทนข้อมูลและขั้นตอนวิธีการจัดการข้อมูลดังกล่าวด้วย

1.2 ขั้นตอนวิธี (Algorithm)

ขั้นตอนวิธีเป็นสิ่งสำคัญของการเขียนโปรแกรมซึ่งจะแสดงให้ทราบถึงประสิทธิภาพและความถูกต้องของโปรแกรม หากใช้ขั้นตอนวิธีที่ไม่ดีพอแล้วจะได้รับความยุ่งยากในภายหลังได้ เช่น โปรแกรมใช้เวลาทำงานมาก แก้ไขยาก คำตอบไม่น่าเชื่อถือ และมีประสิทธิภาพต่ำ

การเขียนโปรแกรมนั้นควรคำนึงถึงความเชื่อถือได้ (Reliability) รูปแบบการเขียนโปรแกรม (Program style) ประสิทธิภาพ (Efficiency) การแก้ไขข้อผิดพลาด (Debugging) การทดสอบ (Testing) และการบำรุงรักษาโปรแกรม (Maintainability) หากโปรแกรมใดมีคุณสมบัติดังกล่าวแล้ว โปรแกรมนั้นย่อมอำนวยประโยชน์แก่ผู้ใช้ได้อย่างดี อีกทั้งเป็นการขจัดข้อขัดข้องและ

ปัญหาต่าง ๆ ที่จะเกิดขึ้นในขณะทำงาน หรือเมื่อต้องแก้ไขปรับปรุงโปรแกรมให้เหมาะสมกับลักษณะงานที่ต้องพัฒนาต่อไป อย่างไรก็ดี การที่จะให้บรรลุวัตถุประสงค์ดังกล่าว จำเป็นต้องเกี่ยวข้องกับขั้นตอนวิธีที่ดี และโปรแกรมโครงสร้าง (Structured Program) ที่จะช่วยให้โปรแกรมมีคุณภาพ

ขั้นตอนวิธีคือวิธีดำเนินการที่ประกอบด้วยเซตที่แน่นอนของกฎที่ไม่คลุมเครือซึ่งระบุอันดับที่แน่นอนของการปฏิบัติการที่จะเป็นคำตอบให้กับปัญหา

1.2.1 คุณลักษณะของขั้นตอนวิธี

ขั้นตอนวิธีประกอบด้วยคุณลักษณะสำคัญ 5 ประการ ได้แก่

1) ความแน่นอน (Finiteness) ขั้นตอนวิธีจะทำงานตามจำนวนขั้นตอนที่มีความแน่นอนและจะต้องหยุดภายหลังทำงานเสร็จ

2) ความหมายแน่นอน (Definiteness) ทุกขั้นตอนต้องมีความหมายชัดเจนไม่คลุมเครือ

3) อินพุท (Input) ปริมาณที่กำหนดให้ก่อนการเริ่มต้นของขั้นตอนวิธีซึ่งอาจมีหรือไม่ก็ได้

4) เอาท์พุท (Output) ขั้นตอนวิธีจะต้องมีเอาท์พุทอย่างน้อยหนึ่งประเภทหรือหนึ่งจำนวน

5) ความสำเร็จผล (Effectiveness) กล่าวคือ ปฏิบัติการที่เกี่ยวข้องจะต้องเป็นพื้นฐานที่เพียงพอ สามารถกระทำได้อย่างแน่นอน ในระยะเวลาที่จำกัด และสมเหตุสมผล

การออกแบบขั้นตอนวิธีสามารถเขียนได้ในรูปแบบที่นิยม ได้แก่ ผังงาน (Flowchart) และรหัสเทียม (Pseudo Code)

1.2.2 ผังงาน (Flowchart)

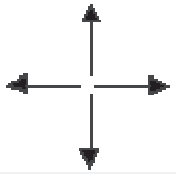
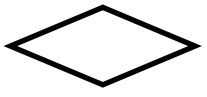
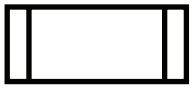
ผังงานหมายถึง ผังงานมีลักษณะเป็นแผนภาพที่มีการใช้สัญลักษณ์รูปภาพและลูกศรที่แสดงถึงขั้นตอนการทำงานของโปรแกรมหรือระบบทีละขั้นตอน รวมไปถึงทิศทางการไหลของข้อมูลตั้งแต่แรกจนได้ผลลัพธ์ตามที่ต้องการประกอบด้วยเส้นและสัญลักษณ์รูปต่างๆ

ซึ่งนักเขียนโปรแกรมใช้ เพื่อให้มองเห็นขั้นตอนการทำงานของโปรแกรม แบ่งเป็น 2 ประเภท คือ ผังงานกว้างๆ (General Flowchart) และผังงานละเอียด (Detailed Flowchart) ผังงานจะช่วยทำให้เข้าใจการทำงาน ทำให้การเขียนโปรแกรมง่ายขึ้น เนื่องจากคอมพิวเตอร์จะทำงานเป็นขั้นเป็นตอน ซึ่งผู้เขียนโปรแกรมจะต้องเป็นผู้กำหนดให้ ผังงานจะช่วยให้ผู้เขียนโปรแกรมมองเห็นขั้นตอนต่างๆ จากสัญลักษณ์ที่กำหนดไว้อย่างง่าย และไม่เกิดความสับสนประโยชน์ของผังงานนั้น ช่วยลำดับขั้นตอนการทำงานของโปรแกรม และสามารถนำไปเขียนโปรแกรมได้โดยไม่สับสน ช่วยในการตรวจสอบ และแก้ไขโปรแกรมได้ง่าย เมื่อเกิดข้อผิดพลาด ช่วยให้การดัดแปลง แก้ไข ทำได้อย่างสะดวกและรวดเร็ว ช่วยให้ผู้อื่นสามารถศึกษาการทำงานของโปรแกรมได้อย่างง่าย และรวดเร็วมากขึ้น

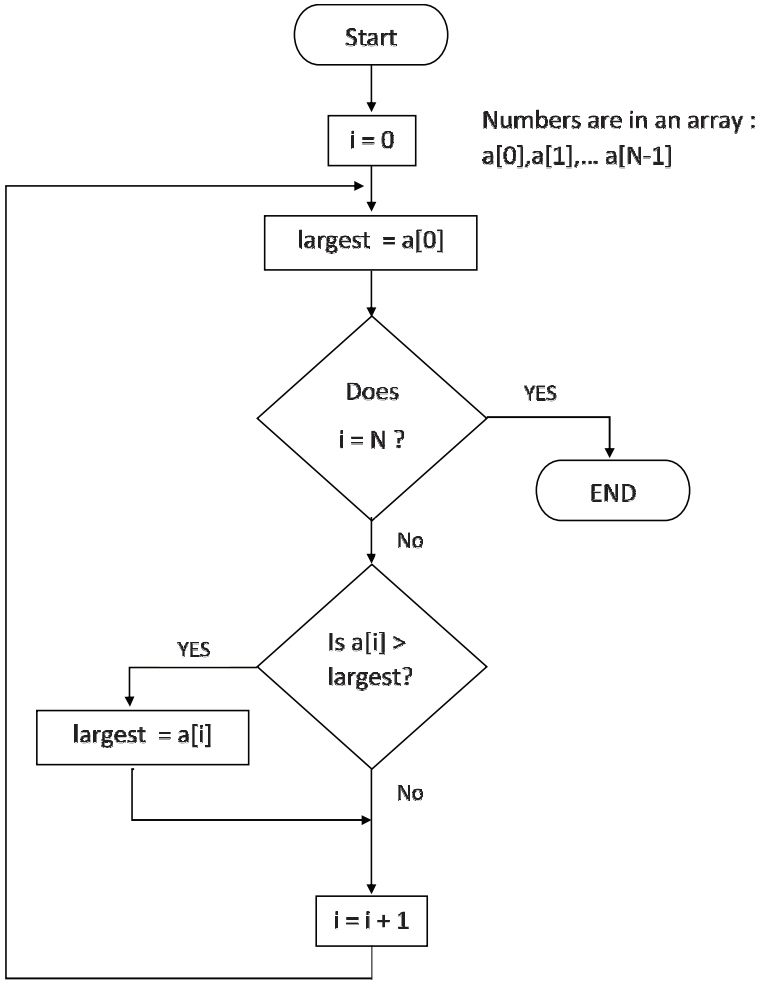
วิธีการเขียนผังงานที่ดีต้องมีลูกศรแสดงทิศทางเข้า - ออกควรใช้สัญลักษณ์ตามที่กำหนดไว้ใช้ลูกศรแสดงทิศทางการไหลของข้อมูลจากบนลงล่าง หรือจากซ้ายไปขวามีคำอธิบายในภาพควรสั้นกะทัดรัด และเข้าใจง่ายไม่ควรโยงเส้นเชื่อมผังงานที่อยู่ไกลมากๆ ควรใช้สัญลักษณ์จุดเชื่อมต่อแทน ผังงานควรมีการทดสอบความถูกต้องของการทำงานก่อนนำไปเขียนโปรแกรม

การเขียนผังโปรแกรมจะประกอบไปด้วยการใช้สัญลักษณ์มาตรฐานต่างๆ ที่เรียกว่าสัญลักษณ์ ANSI (American National Standards Institute) ในการสร้างผังงาน ตามสัญลักษณ์ในตารางที่ 1.1 และดังตัวอย่างที่แสดงในรูป 1.1 ต่อไปนี้

ตารางที่ 1.1 ตารางสัญลักษณ์และความหมายของผังงาน

สัญลักษณ์	ความหมาย
	จุดเริ่มต้น / สิ้นสุดของโปรแกรม
	ลูกศรแสดงทิศทางการทำงานของโปรแกรมและการไหลของข้อมูล
	ใช้แสดงคำสั่งในการประมวลผล หรือการกำหนดค่าข้อมูลให้กับตัวแปร
	รับและแสดงข้อมูลจากหน่วยเก็บข้อมูลหรือการประมวลผลออกมา
	การตรวจสอบเงื่อนไขเพื่อตัดสินใจ โดยจะมีเส้นออกจากรูปเพื่อแสดงทิศทางการทำงานต่อไป เงื่อนไขเป็นจริงหรือเป็นเท็จ
	แสดงผลหรือรายงานที่ถูกสร้างออกมาทางเครื่องพิมพ์
	การทำงานย่อย หรือ โปรแกรมย่อย
	แสดงจุดเชื่อมต่อของผังงานภายใน หรือเป็นที่บรรจบของเส้นหลายเส้นที่มาจากหลายทิศทางเพื่อจะไปสู่การทำงานอย่างใดอย่างหนึ่งที่เหมือนกัน
	การขึ้นหน้าใหม่ ในกรณีที่ผังงานมีความยาวเกินกว่าที่จะแสดงพอในหนึ่งหน้า

ตัวอย่างที่ 1.1 กำหนดให้ค้นหาค่าที่มากที่สุดในชุดของแถวลำดับ a และกำหนดตัวแปร largest เพื่อเก็บค่ามากที่สุด การทำงานเริ่มจากต้องค้นหาด้วยการเปรียบเทียบค่าในทุกตำแหน่งของชุดตั้งแต่ข้อมูลจึงกำหนดให้ค่าในแถวลำดับ a ตำแหน่งแรกสมมุติเป็นค่ามากที่สุดเก็บไว้ที่ตัวแปร largest แล้วนำไปเปรียบเทียบกับข้อมูลใน แถวลำดับ a ในตำแหน่งถัดไปจนครบ ซึ่งการเปรียบเทียบนี้หากพบว่าค่าในแถวลำดับ a ที่ตำแหน่งใดมีค่ามากกว่า largest ที่กำลังเก็บในรอบปัจจุบัน ก็จะเปลี่ยนค่าตัวแปร largest ไปเป็นค่านั้นแทน ทำจนครบแล้วจะพบว่าไม่มีค่าใดมากกว่าค่าของตัวแปร largest แล้วจึงสามารถสรุปได้ว่าค่า largest นั้นเป็นค่าที่มากที่สุดได้ดังแสดงใน ผังงานต่อไปนี้



รูปที่ 1.1 ผังงานแสดงการหาจำนวนที่มีค่ามากที่สุดในชุดแถวลำดับ

เมื่อนำรูปที่ผังงานที่ 1.1 มาเขียนโปรแกรมภาษาซีเพื่อหาค่ามากที่สุดที่อยู่ในชุดตัวแปร a ที่ประกอบด้วยจำนวนเต็มชุดหนึ่ง และ กำหนดตัวแปร largest เพื่อเก็บค่ามากที่สุด จะได้ดังนี้

```
#include <stdio.h>

int main() {  int a[10] = {10, 7, 2, 4, 55, 61, 27, 18, 9};

                int loop, largest;

             largest = a[0];



   for(loop = 1; loop < 10; loop++) {

        if( largest < a[loop] )

         largest = a[loop];

   }

     printf("Largest element of array is %d", largest);

      return 0;

}
```

1.2.3 รหัสเทียม (Pseudo Code)

รหัสเทียมที่ใช้เขียนขั้นตอนวิธีอาจจะอยู่ในรูปแบบคล้ายภาษาโปรแกรมต่างๆ หรือคณิตศาสตร์ อาจเริ่มด้วยคำว่า "ขั้นตอนวิธี" ตามด้วยชื่อของขั้นตอนวิธี แล้วจึงตามด้วยขั้นตอนต่างๆ แต่ละขั้นตอนเริ่มต้นด้วยวลีอยู่ในวงเล็บ [ ] วลีเหล่านี้เป็นคำอธิบายสั้นๆ ของขั้นตอนนั้น สัญลักษณ์ที่ใช้ ได้แก่ A ←B หมายถึงการแทนค่าของตัวแปร A ด้วยค่าปัจจุบันของ B สัญลักษณ์ A ← B ← C หมายถึงการแทนค่าของตัวแปร A และ B ด้วยค่าปัจจุบันของ C ผลของสัญลักษณ์

ทำให้ตัวแปร A, B และ C มีค่าเท่ากัน ส่วนสัญลักษณ์ A ↔ B หมายถึงการสลับค่ากันระหว่างค่าของตัวแปร A และ B คำสั่งที่ใช้ ได้แก่ คำสั่ง If ใช้ตรวจสอบเงื่อนไข คำสั่ง Case ใช้ในกรณีที่มีหลายทางเลือกคำสั่ง Repeat ใช้ในกรณีที่ต้องทำตามคำสั่งอย่างน้อยหนึ่งคำสั่งโดยทำซ้ำหลายๆ รอบ คำสั่ง Go to ใช้ในกรณีที่ต้องการข้ามบางขั้นตอนหรือย้อนกลับไปเริ่มต้นที่บางขั้นตอนซึ่งผ่านมาแล้ว

คำสั่ง Exitloop ใช้ในกรณีที่ต้องการออกจากการวนซ้ำ คำสั่งสุดท้ายได้แก่คำสั่ง Exit ใช้หยุดขั้นตอนวิธีคำสั่งนี้อาจจะอยู่ที่ขั้นใดๆ ก็ได้ โดยมากมักจะอยู่ท้ายของขั้นตอนวิธี สำหรับคำสั่ง Go to และ Exitloop นั้นควรหลีกเลี่ยงเพราะการใช้สองคำสั่งนี้ จะทำให้ขั้นตอนวิธีขาดโครงสร้าง ดังนั้นจะใช้สองคำสั่งนี้ต่อเมื่อจำเป็นจริงๆ

นอกจากนี้ ยังมีขั้นตอนวิธีย่อย (Subalgorithm) ซึ่งเป็นส่วนที่เป็นอิสระจากขั้นตอนวิธี ด้วยเหตุนี้จึงมีการกำหนดขั้นตอนวิธีย่อยให้แยกออกมาจากขั้นตอนวิธีหลัก (Main Algorithm) จุดประสงค์ของขั้นตอนวิธีย่อยก็เพื่อต้องการคำนวณบางอย่างภายใต้การควบคุมของขั้นตอนวิธีหลัก การคำนวณนี้อาจจะใช้ตัวแปรเสริมหรือพารามิเตอร์ (Parameter) ซึ่งส่งมาจากขั้นตอนวิธีอื่น รูปแบบการเขียนขั้นตอนวิธีย่อยเหมือนกับขั้นตอนวิธีหลัก ยกเว้นคำสั่งสุดท้ายของขั้นตอนวิธีย่อย ใช้คำสั่ง Return แทนคำสั่ง Exit และรายชื่อพารามิเตอร์จะอยู่ต่อจากชื่อขั้นตอนวิธีย่อย ข้อสังเกตประการหนึ่งคือขั้นตอนวิธีย่อยต่างๆ อาจจะช่วยเหลือซึ่งกันและกัน และขั้นตอนวิธีย่อยอาจเรียกใช้ตัวมันเอง ซึ่งเรียกว่า การเวียนเกิด หรือ รีเคอร์ชัน (Recursion) ตัวอย่างของขั้นตอนวิธีย่อยได้แก่ ฟังก์ชัน (Function) และกระบวนคำสั่ง (Procedure) ซึ่งจะได้อธิบายดังนี้

ใช้ฟังก์ชันเมื่อต้องการส่งผลลัพธ์เพียงหนึ่งค่ากลับไปยังขั้นตอนวิธีที่เรียกใช้ (Calling Algorithm) การส่งกลับทำได้โดยใช้คำสั่ง Return (Value) ในการเขียนฟังก์ชันควรเริ่มต้นด้วยคำว่า "ฟังก์ชัน" ตามด้วยชื่อและพารามิเตอร์ต่างๆ เช่น

ฟังก์ชัน NAME (PARM1,PARM2, ... , PARMN)

ส่วนกระบวนคำสั่งนั้น มีลักษณะคล้ายฟังก์ชันแต่ไม่ได้ให้ค่าออกมาอย่างเด่นชัด สำหรับกระบวนคำสั่งที่มีพารามิเตอร์ จะส่งผลลัพธ์ผ่านทางพารามิเตอร์ดังตัวอย่างของขั้นตอนวิธีที่เขียนในรูปฟังก์ชันและกระบวนคำสั่งรูปแบบรหัสเทียมดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1.2 กำหนดจำนวนเต็ม 10 จำนวน ซึ่งบรรจุค่าไว้ในแถวลำดับ เช่นกำหนดให้มีค่าในแถวลำดับ (Array) ดังนี้

Index	0	1	2	3	4	5	6	7	8	9
Value	24	13	20	53	46	15	36	72	8	29

การหาผลรวมในชุดแถวลำดับสามารถเขียนในรูปแบบขั้นตอนวิธีได้มากกว่าหนึ่งแบบดังนี้

ฟังก์ชัน SUMATION

1. [กำหนดค่าเริ่มต้น]

   SUM ⟵ 0

2. [คำนวณผลรวม]

   Repeat while <= Array_Length <= Length-1

   SUM ⟵ Array[i] + SUM

3. [ส่งผลลัพธ์]

   Return (SUM)

ตัวอย่างที่ 1.3 ต้องการคำนวณหาค่าเฉลี่ยของ VALUE1 และ VALUE2 ซึ่งเป็นเลขจำนวนจริง AVER เป็นตัวแปรซึ่งเก็บค่าผลลัพธ์เพื่อส่งไปยังขั้นตอนวิธีที่เรียกใช้

ฟังก์ชัน AVERAGE (VALUE1 , VALUE2)

1. [กำหนดค่าเริ่มต้น]

   AVER ⟵ 0

2. [คำนวณค่าเฉลี่ย]

   AVER ⟵ (VALUE1 + VALUE2) / 2.0

3. [ส่งผลลัพธ์]

   Return (AVER)

ในขั้นตอนวิธีที่เรียกใช้ฟังก์ชันนี้ อาจมีคำสั่ง E ⟵ AVERAGE(X, Y) ซึ่งทำให้ค่าเฉลี่ยอยู่ในตัวแปร E

ตัวอย่างที่ 1.4 การคำนวณหาค่าแฟกทอเรียลซึ่งสามารถทำได้หลายขั้นตอนวิธี ได้แก่ แบบเวียนเกิด หรือ การคำนวณแบบสืบเนื่องไปข้างหน้า (Forward) หรือ การคำนวณแบบสืบเนื่องย้อนหลัง (Backward) พิจารณาการคำนวณหาแฟกทอเรียล N ตามข้อกำหนด ดังนี้

$$\text{FACT (N)} = \begin{cases} 1 & \text{ถ้า } N = 0 \\ N * \text{FACT (N - 1)} & \text{ถ้า } N \neq 0 \end{cases}$$

จะเห็นว่าการกำหนด FACT (N) อยู่ในรูปของ FACT (N - 1) และทำให้ FACT (N - 1) อยู่ในรูปของ FACT (N - 2) และต่อๆ ไป จนกระทั่งถึง FACT (0) ซึ่งมีค่าเป็นหนึ่ง ซึ่งหากใช้วิธีการของโปรแกรมแบบเวียนเกิด (Recursive) ในการคำนวณหาค่าแฟกทอเรียลซึ่งเขียนเป็นขั้นตอนวิธี FACT1 ดังนี้

ฟังก์ชัน FACT1 (N)

1. [กำหนดค่าเริ่มต้น]

   Input: N

2. [เงื่อนไขเป็นจริงหรือไม่ ?]

   If N = 0

   then FACTORIAL(N)⟵1

   Go to step 4

   Else Go to step 3

3. [คำนวณ N!]

   FACTORIAL(N)⟵ N * FACTORIAL(N-1)

4. [นำค่า N และคืนค่าคำตอบ]

   Return (FACTORIAL)

โปรแกรมหาค่าแฟกทอเรียลด้วยวิธีการเวียนเกิด

```
#include<stdio.h>

 long factorial(int);

 int main(){
 int n;
            long f;

            printf("Enter an integer to find its factorial\n");

           scanf("%d", &n);

           if (n < 0)

                     printf("Factorial of negative integers isn't defined.\n");

          else      {    f = factorial(n);

                          printf("%d! = %ld\n", n, f);

           }

            return 0;

}

 long factorial(int n){

            if (n == 0)    return 1;

                  else     return(n * factorial(n-1));

}
```

ผลลัพธ์ของโปรแกรมหาค่าแฟกทอเรียลแบบเวียนเกิด

```
Enter an integer to find its factorial

6

6! = 720
```

สำหรับขั้นตอนวิธีการคำนวณหาค่าแฟกทอเรียลซึ่งใช้การควบคุมวนรอบการคำนวณแบบสืบเนื่องไปข้างหน้าตาม FACT**2** และ การคำนวณแบบสืบเนื่องย้อนหลัง FAC3 ดังนี้

ฟังก์ชัน FACT2(N)

1. [กำหนดค่าเริ่มต้น]

   PROD ⟵1

2. [กระทำซ้ำสำหรับทุก ๆ ค่าของ I]

   Repeat for I = 1, 2, ... N

   PROD ⟵ PROD * I

3. [เสร็จเรียบร้อย]

   Return PROD

ฟังก์ชัน FACT3(N)

1. [กำหนดค่าเริ่มต้น]

   PROD ⟵1

2. [กระทำซ้ำสำหรับทุก ๆ ค่าของ I]

   Repeat for I = N , N - 1, ... 2, 1

   PROD ⟵ PROD * I

3. [เสร็จเรียบร้อย]

   Return PROD

จากขั้นตอนวิธีคำนวณหาค่าแฟกทอเรียลซึ่งใช้การควบคุมวนรอบการคำนวณสามารถเขียนโปรแกรมการหาค่าแฟกทอเรียลโดยใช้การวนรอบการทำงานได้ในหลายรูปแบบเช่นโปรแกรมภาษาซีดังต่อไปนี้

```
#include <stdio.h>

int main(){

int c, n, fact = 1;

   printf("Enter a number to calculate its factorial\n");

                    scanf("%d", &n);

                    for (c = 1; c <= n; c++)

                              fact = fact * c;

                               printf("Factorial of %d = %d\n", n, fact);

                   return 0;

}
```

ผลลัพธ์ของโปรแกรมหาค่าแฟกทอเรียลโดยใช้การวนรอบการทำงานด้วยคำสั่ง loop for

```
Enter a number to calculate its factorial

6

Factorial of 6 = 720
```

โปรแกรมหาค่าแฟกทอเรียลโดยใช้การวนรอบการทำงานด้วยคำสั่ง loop for โดยเขียนเป็นฟังก์ชั่นการทำงานย่อย

```
#include <stdio.h>

long factorial(int);

int main(){

        int number;

                long fact = 1;

                 printf("Enter a number to calculate its factorial\n");

                 scanf("%d", &number);

                 printf("%d! = %ld\n", number, factorial(number));

                 return 0;

}

 long factorial(int n){

        int c;

                long result = 1;

                for (c = 1; c <= n; c++)

                        result = result * c;

                 return result;

}
```

ผลลัพธ์ของโปรแกรมหาค่าแฟกทอเรียลโดยใช้การวนรอบการทำงานด้วยคำสั่ง loop for โดยเขียนเป็นฟังก์ชั่นการทำงานย่อย

```
Enter a number to calculate its factorial

6

6! = 720
```

1.2.4 การวิเคราะห์ขั้นตอนวิธี

ในการแก้ปัญหาเดียวกันอันหนึ่งจะพบว่าอาจมีขั้นตอนวิธีหลายวิธีที่สามารถแก้ปัญหาเดียวกันนั้นได้ ในการเปรียบเทียบว่าขั้นตอนวิธีใดดีกว่ากันจะต้องพิจารณาคุณลักษณะ 3 ประการดังนี้

1) ความถูกต้องแม่นยำของคำตอบ (Accuracy of the Answer)
2) หน่วยความจำที่ใช้ (Storage Requirement)
3) เวลาที่ใช้ (Time Taken)

ขั้นตอนวิธีต้องมีความถูกต้องแม่นยำของคำตอบในการแก้ปัญหาเป็นหลัก ดังนั้นการเปรียบเทียบประสิทธิภาพของขั้นตอนวิธีจึงเน้นไปที่เวลาที่ใช้ในการประมวลผลและหน่วยความจำที่ต้องใช้ของขั้นตอนวิธีนั้นๆ สาเหตุที่ต้องทราบจำนวนของหน่วยความจำที่จะต้องใช้เนื่องจากต้องการทราบว่าขั้นตอนวิธีนั้นสามารถรองรับจำนวนข้อมูลที่ส่งเข้ามาประมวลผล (Input Data) ได้มากที่สุดเท่าใด เพื่อให้ขั้นตอนวิธีนั้นสามารถประมวลผลได้ หรือ กรณีที่ต้องประมวลผลบนเครื่องคอมพิวเตอร์ที่ใช้งานร่วมกันหลายคนในเครือข่ายจะได้ไม่กระทบกับการทำงานของคนอื่น และเพื่อเลือกคุณลักษณะของคอมพิวเตอร์ที่จะใช้ติดตั้งโปรแกรมที่พัฒนาขึ้นได้อย่างเหมาะสม เพราะถ้าหากนำไปติดตั้งที่เครื่องที่มีหน่วยความจำไม่เพียงพอโปรแกรมก็จะไม่ทำงาน