SE-ED COMPUTER BOOK developer

คอมพิวเตอร์และการเขียนโปรแกรม

(Computers and Programming)

C และ Java

- เครื่องมือทางความคิด เพื่อพิชิตการเขียนโปรแกรมทุกรูปแบบ
- เทคนิคการคิดเพื่อการเขียนโปรแกรมให้ได้ดังใจ อย่างเป็นระบบ

ธนัญชัย ตรีภาค

ซีเอ็ด

คำนำ

ในปัจจุบัน รูปแบบการศึกษาเรียนรู้ด้านการเขียนโปรแกรมคอมพิวเตอร์ที่พบมากคือ การศึกษาคำสั่งของภาษาคอมพิวเตอร์ที่ใช้ในการเขียนโปรแกรม และอธิบายตามตัวอย่างโปรแกรมต่างๆ ซึ่งรูปแบบการศึกษาดังกล่าว มีแหล่งข้อมูลมากมายที่จะให้ข้อมูลชุดคำสั่งที่จำเป็น ไม่ว่าจะเป็นจากหนังสือต่างๆ ในอินเทอร์เน็ต หรือแม้กระทั่งมีการจัดเป็นคอร์สการเรียนการสอนเฉพาะภาษาใดภาษาหนึ่งโดยเฉพาะ สำหรับนักเรียนนักศึกษาที่กำลังเรียนวิชาเกี่ยวกับพื้นฐานการเขียนโปรแกรมคอมพิวเตอร์ในขณะนี้ จะพบว่าแนวทางดังกล่าวไม่ได้ทำให้ผู้เรียนเขียนโปรแกรมที่ซับซ้อนได้เลย ปัญหาที่พบมากก็คือ ผู้เรียนทราบว่า คำสั่งต่างๆ สามารถสั่งการให้คอมพิวเตอร์ทำอะไรได้บ้าง แต่ไม่สามารถเรียบเรียงจนเป็นโปรแกรมที่สมบูรณ์ตามต้องการได้ สาเหตุของปัญหานี้เกิดจากการศึกษาการเขียนโปรแกรมไม่ได้มีเฉพาะการศึกษาคำสั่งต่างๆ เท่านั้น ยังต้องมีขั้นตอนของการคิดเพื่อให้ได้กระบวนการทำงานของโปรแกรมที่ต้องการด้วย โดยหากเปรียบการเขียนโปรแกรมกับการสร้างรถยนต์คันหนึ่ง การศึกษาเกี่ยวกับคำสั่งและโครงสร้างภาษาก็เป็นเพียงแค่การศึกษาการใช้เครื่องมือต่างๆ เช่น ไขควง แท่นเจาะ อุปกรณ์พ่นสี ฯลฯ ในโรงงานผลิตเท่านั้น ซึ่งไม่ได้ช่วยให้เข้าใจถึงกระบวนการสร้างรถยนต์แต่อย่างใด อย่างนี้แล้วคงไม่สามารถสร้างรถยนต์ตามต้องการได้

หนังสือเล่มนี้ จะช่วยเติมเต็มช่องว่างที่ขาดหายไปของการศึกษาการเขียนโปรแกรมในปัจจุบัน ซึ่งก็คือ “ทักษะการคิดเพื่อเขียนโปรแกรม” โดยเนื้อหาในหนังสือจะช่วยให้ผู้อ่านทราบว่า ต้องคิดอย่างไรจึงจะสามารถเขียนโปรแกรมได้ ตัวอย่างในหนังสือจะช่วยให้ผู้อ่านเข้าใจถึงกระบวนการคิด และเห็นภาพกระบวนการต่างๆ ของการเขียนโปรแกรมอย่างชัดเจน

ผู้เขียนหวังว่า ผู้อ่านจะได้รับประโยชน์จากเนื้อหาต่างๆ ในหนังสือเล่มนี้ จนทำให้ผู้อ่านสามารถคิด วางแผน และสร้างโปรแกรมที่ต้องการได้อย่างง่ายดาย

ผู้เขียนขอขอบคุณสำนักพิมพ์ บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) ทีมบรรณาธิการ ทีมผู้ผลิต และทีมงานทุกๆ ท่านที่ได้ช่วยเหลือ ดำเนินการจนมีหนังสือเล่มนี้อยู่ในมือของผู้อ่านทุกท่าน หากผู้อ่านท่านใดมีข้อตำหนิ ติชม หรือคำแนะนำใดๆ ผู้เขียนยินดีน้อมรับเพื่อนำไปใช้ในอนาคตต่อไป

ธนัญชัย ตรีภาค
robiuz@yahoo.com

สารบัญ

1 บทนำ

สำหรับผู้ที่เริ่มศึกษาการเขียนโปรแกรม หลายๆ คนประสบปัญหาเมื่อต้องการเขียนโปรแกรมแล้วไม่ทราบว่าจะต้อง "คิด" เพื่อที่จะเขียนโปรแกรมนั้นๆ ได้อย่างไร ซึ่งปัญหาดังกล่าวอาจมีสาเหตุหลายประการด้วยกัน โดยสาเหตุแรกจะเกิดจากการที่ไม่รู้ว่า กระบวนการทำงานของคอมพิวเตอร์ทำงานอย่างไร ปัญหานี้มักจะเกิดขึ้นกับโปรแกรมเมอร์มือใหม่ ซึ่งสามารถแก้ปัญหาได้จากการศึกษาเกี่ยวกับพื้นฐานการทำงานของคอมพิวเตอร์ สาเหตุของปัญหาดังกล่าวสามารถแก้ไขได้ไม่ยากนัก และสาเหตุต่อมาคือ โปรแกรมเมอร์ไม่ทราบลำดับกระบวนการทำงานของโจทย์ที่ต้องเปลี่ยนให้กลายเป็นโปรแกรม หรือแม้กระทั่งไม่สามารถนำเสนอ "ความคิด" ของตัวเองให้กลายเป็นลำดับกระบวนการทำงานได้ ซึ่งปัญหาดังกล่าวจะเป็นปัญหาสามัญที่โปรแกรมเมอร์หลายๆ คนต้องเจอ เนื่องจากโจทย์ในการเขียนโปรแกรมมีความยากง่ายแตกต่างกัน และขึ้นอยู่กับประสบการณ์ในการเขียนโปรแกรมของโปรแกรมเมอร์คนนั้นๆ ด้วย

จากปัญหาสามัญที่เกิดขึ้นในการศึกษาการเขียนโปรแกรม ทำให้โปรแกรมเมอร์ต้องตระหนักถึงการฝึกฝนความสามารถในการคิดเพื่อการเขียนโปรแกรมให้มากขึ้น โดยโปรแกรมเมอร์ควรทราบถึงกระบวนการคิดที่ทำให้การเขียนโปรแกรมทำได้อย่างราบรื่น โดย "การคิด" นี้ถือว่าเป็นทักษะที่สามารถฝึกฝนเพื่อพัฒนาให้มีประสิทธิภาพสูงขึ้นได้ คล้ายการว่ายน้ำ หรือขี่จักรยาน แต่ต้องใช้เวลาในการฝึกฝนจนเกิดความชำนาญ เมื่อชำนาญแล้ว ทักษะดังกล่าวจะเป็นทักษะที่ติดตัวไปเรื่อยๆ ไม่หายไป เช่นเดียวกับคนที่ว่ายน้ำ หรือขี่จักรยานเป็นแล้ว ก็จะยังคงสามารถทำได้ ถึงแม้ว่าเวลาจะผ่านไปนานเท่าใดก็ตาม

ทักษะการคิด

ก่อนที่จะศึกษาเกี่ยวกับการเขียนโปรแกรมภาษานั้น โปรแกรมเมอร์ควรทราบถึงลักษณะของ "การคิด" กันก่อน เนื่องจากหลายๆ คนไม่สามารถแยกแยะข้อแตกต่างระหว่าง "การคิด" กับ "การทบทวนคำถาม" ซึ่งทำให้เกิดปัญหาตั้งแต่การเริ่มต้นเขียนโปรแกรม สำหรับการคิดเพื่อแก้ปัญหาต่างๆ เป็นทักษะอย่างหนึ่ง โดยผลลัพธ์ที่ได้จากการคิดก็คือ จะได้ข้อสังเกต ข้อมูล หรือข้อสรุป ที่เพิ่มเติมขึ้นจากเดิม โดยข้อสังเกต ข้อมูล หรือข้อสรุปที่ปรากฏขึ้นนั้น จะสามารถนำไปต่อยอด หรือใช้เป็นข้อมูลในการคิดเพื่อหาผลลัพธ์ และข้อสรุปอื่นๆ ต่อไปได้

นอกจากนี้การคิดยังสามารถขยายกรอบความคิดออกไปเรื่อยๆ จนมีรายละเอียดเพียงพอที่จะสรุปความ หรือตอบคำถามอื่นๆ ได้ ซึ่งตรงกันข้ามกับการทบทวนคำถาม ที่จะมีความคิดทบทวนคำถามซ้ำไปเรื่อยๆ แต่จะไม่มีข้อสรุป ไม่มีผลลัพธ์ใหม่ ไม่มีกระบวนการ ไม่มีข้อสังเกตใดๆ เกิดขึ้น ซึ่งทำให้ไม่สามารถตอบคำถามที่ต้องการ และไม่ต้องพูดถึงการต่อยอดความคิดให้มีการขยายกรอบความคิดออกซึ่งไม่มีทางทำได้แน่นอน

ยกตัวอย่างเช่น มีปัญหาข้อหนึ่งซึ่งมีปลอกปากกาพลาสติกอยู่ในขวดแก้ว แต่ต้องการนำปลอกปากกาดังกล่าวออกจากขวดโดยต้องไม่สัมผัสขวดแก้วเลย จะต้องทำอย่างไรบ้าง ในกรณีของคนที่พยายามคิดหากระบวนการ จะพยายามคิดวิเคราะห์ถึงคุณสมบัติต่างๆ ของขวดและปลอกปากกาพลาสติก และคิดถึงกระบวนการที่เป็นไปได้เป็นข้อๆ หลังจากนั้นจึงต่อยอดความคิดอย่างเป็นเหตุเป็นผล ซึ่งจากการดำเนินการตามกระบวนการที่คิดขึ้น ในแต่ละขั้นตอนของกระบวนการจะมีผลลัพธ์ที่เข้าใกล้ผลลัพธ์ที่โจทย์ต้องการคือ การเอาปลอกปากกาออกจากขวดมากขึ้นเรื่อยๆ จนสุดท้ายจะได้คำตอบที่ทำให้สามารถนำปลอกปากกาออกจากขวดได้ในที่สุด

แต่สำหรับคนที่ต้องการผลลัพธ์แต่ไม่มีกระบวนการคิด จะมีการทบทวนคำถามซ้ำๆ ไปเรื่อยๆ คือคิดถึงการจะทำอย่างไรจึงจะเอาปลอกปากกาออกจากขวด แต่ไม่มีการคิดถึงข้อมูลที่เป็นเหตุและปัจจัยต่างๆ ไม่มีการระบุทางเลือก ไม่มีการระบุกระบวนการที่เป็นรูปธรรมใดๆ จนสุดท้ายจะมีการจินตนาการถึงภาพปลอกปากกาลอยออกจากขวดอยู่ตลอดเวลา โดยที่ไม่มีคำตอบที่ควรจะมีเกิดขึ้นเลย

ในการคิดเพื่อการเขียนโปรแกรม เมื่อโปรแกรมเมอร์ต้องเขียนโปรแกรมใดๆ โปรแกรมเมอร์ต้องคิดถึงกระบวนการทำงานของคอมพิวเตอร์ที่ให้ผลลัพธ์ตามที่โจทย์ต้องการ โดยการคิดเพื่อเขียนโปรแกรมนี้ ผลลัพธ์ที่ได้จะเป็น "กระบวนการ" ต่างๆ ที่มีขั้นตอนโดยละเอียด ซึ่งรายละเอียดในขั้นตอนนี้จะเป็นการอธิบายกระบวนการเป็นข้อๆ หรือเป็นการอธิบายกระบวนการแบบร้อยแก้ว ซึ่งจะยังไม่มีรายละเอียดของโปรแกรมในรูปแบบของโค้ดของโปรแกรมเลย ซึ่งกระบวนการที่เป็นคำตอบของโจทย์อาจมีได้หลากหลายรูปแบบ แต่สามารถให้ผลลัพธ์การทำงานที่สามารถตอบสนองตามโจทย์ได้ หลังจากได้กระบวนการที่เป็น

คำตอบแล้ว โปรแกรมเมอร์จะแทนค่ากระบวนการต่างๆ ที่คิดขึ้นได้ให้อยู่ในรูปของโค้ดโปรแกรม เพื่อสั่งการให้คอมพิวเตอร์ทำงานตามกระบวนการที่ออกแบบไว้ ดังนั้นโจทย์คำถามทั้งหมดในการเขียนโปรแกรมของโปรแกรมเมอร์ จึงเป็นคำถามในลักษณะ “ทำอย่างไร (How)” มากกว่า “ทำอะไร (What)”

สำหรับการเขียนโปรแกรมใดๆ โปรแกรมเมอร์ต้องแยกงานออกเป็น 2 ส่วน คือ ส่วนของ **“การคิด”** เพื่อให้ได้กระบวนการทำงานที่ให้ผลลัพธ์ตามที่โจทย์ต้องการ และส่วนของ **“การเขียนโปรแกรม”** เพื่อให้ได้โค้ดของโปรแกรม ซึ่งโปรแกรมเมอร์จะเข้าสู่ขั้นตอนของการคิด เพื่อให้ได้กระบวนการทำงานที่ให้ผลลัพธ์ตามที่โจทย์ต้องการโดยสมบูรณ์ก่อน แล้วจึงเขียนโค้ดของโปรแกรม ซึ่งการคิดกระบวนการนี้ควรอยู่ในรูปของการคิดให้ได้องค์ประกอบของการทำงานในภาพรวมก่อน แล้วจึงขยายความให้มีรายละเอียดเพิ่มเติม จนเพียงพอที่จะเขียนเป็นโปรแกรมได้ หลังจากนั้นจึงเริ่มเขียนโค้ดโปรแกรมจนได้โปรแกรมที่สมบูรณ์

โค้ดเทียม (Pseudo Code)

เนื่องจากการเขียนโปรแกรมใดๆ จะเป็นการเขียนโปรแกรมของการทำงานที่เป็นรูปธรรม ซึ่งสามารถอธิบายขั้นตอนการทำงานต่างๆ ได้อย่างชัดเจนเท่านั้น โปรแกรมเมอร์จะไม่สามารถเขียนโปรแกรมของการทำงานที่เป็นนามธรรมได้ ดังนั้น การคิดกระบวนการต่างๆ ต้องคิดถึงกระบวนการที่เป็นรูปธรรมเท่านั้น โดยการชี้วัดว่า สิ่งที่คิดขึ้นมานั้นเป็นรูปธรรมหรือยัง สามารถทำได้โดยการอธิบายกระบวนการที่คิดได้ออกมาให้อยู่ในรูปแบบต่างๆ เช่น การเขียนเป็นร้อยแก้วเพื่ออธิบายการทำงานทั้งหมด การใช้โฟลว์ชาร์ตเพื่อเป็นผังแสดงการทำงาน การวาดรูปเพื่อทำให้เห็นกระบวนการทำงานตามโจทย์ต้องการ ฯลฯ แต่กระบวนการที่เหมาะสมกับการเขียนโปรแกรมมากที่สุดคือ **“การใช้โค้ดเทียม” (Pseudo Code)** ซึ่งเป็นวิธีในการอธิบายลักษณะการทำงานของโปรแกรมที่ต้องการสร้าง ซึ่งจะมีรูปแบบการเขียนลำดับการทำงานเป็นข้อๆ เรียงต่อกันโดยไม่มีการกำหนดรูปแบบไวยากรณ์ของภาษา รูปแบบคำสั่ง หรือนิยามคำสั่งใดๆ ทั้งสิ้น จึงมีข้อดีคือ ทำให้โปรแกรมเมอร์สามารถคิดกระบวนการ และเขียนโค้ดเทียมได้อย่างอิสระ สามารถเขียนการทำงานของโปรแกรมได้ง่าย โดยโค้ดเทียมสามารถเขียนอธิบายการทำงานของโปรแกรมในภาพรวม หรือลงรายละเอียดการทำงานย่อยๆ ได้ อีกทั้งเราสามารถเปลี่ยนโค้ดเทียมให้กลายเป็นโค้ดของโปรแกรมได้โดยง่าย

Jigsaw & Zoom In

ในการวาดภาพหนึ่งภาพ จิตรกรจะทำการวาดภาพโดยการร่างโครงของภาพให้เห็นองค์ประกอบต่างๆ ในภาพรวมก่อน โดยกำหนดว่า ในภาพนั้นจะมีองค์ประกอบของภาพอะไรบ้าง แต่ละส่วนจะอยู่ตรงตำแหน่งใดของภาพ แล้วจึงค่อยๆ ลงรายละเอียดขององค์ประกอบแต่ละส่วนไปเรื่อยๆ จนครบถ้วนกลายเป็นภาพที่สวยงาม ซึ่งในการวาดภาพโดยทั่วไปเราจะไม่พบการวาดภาพที่มีการลงรายละเอียดของภาพทีละจุด ตั้งแต่ขอบบนซ้ายของภาพ ไล่เรียงไปเรื่อยๆ จนถึงขอบล่างขวาของภาพ

สำหรับการเขียนโปรแกรมก็เช่นกัน โปรแกรมเมอร์จะไม่ดำเนินการเขียนโปรแกรมโดยเขียนโค้ดทีละบรรทัด ตั้งแต่บรรทัดแรกไปเรื่อยๆ จนถึงบรรทัดสุดท้าย ขั้นตอนการเขียนโปรแกรมที่ถูกต้องจะเหมือนกับขั้นตอนการวาดรูปคือ ต้องเขียนโครงร่างของโปรแกรมให้เห็นภาพรวมก่อน ซึ่งในการเขียนโครงร่างของโปรแกรมสามารถทำได้โดยการวาดรูป หรือเขียนอธิบายโดยภาษาง่ายๆ หรือเป็น "โค้ดเทียม (Pseudo Code)" หลังจากนั้นจึงเพิ่มเติมรายละเอียดขององค์ประกอบแต่ละส่วนไปเรื่อยๆ จนกว่าจะได้รายละเอียดในระดับคำสั่งพื้นฐานของคอมพิวเตอร์ ร้อยเรียงกันเป็นโปรแกรมที่มีการทำงานอย่างถูกต้องตามที่ได้ออกแบบโครงร่างไว้

สำหรับการดำเนินการเขียนโครงร่างของโปรแกรมแล้วลงรายละเอียดนี้ จะมีการนำเสนอกระบวนการคิดรูปแบบหนึ่งคือ การจัดกลุ่ม "Jigsaw" และการ "Zoom In" โดยแนวคิดแบบการจัดกลุ่ม Jigsaw จะเป็นการคิดโดยเปรียบเทียบการเขียนโปรแกรมเป็นการต่อ Jigsaw ซึ่งสิ่งที่เหมือนกันของการต่อ Jigsaw และการเขียนโปรแกรมก็คือ มีความซับซ้อนในการร้อยเรียงเหมือนกัน มีการแยกแยะองค์ประกอบเหมือนกัน ไม่สามารถทำตามลำดับจากโค้ดบรรทัดแรกหรือ Jigsaw แผ่นแรก ไปยังโค้ดบรรทัดสุดท้าย หรือ Jigsaw แผ่นสุดท้ายเหมือนกัน

ขั้นตอนของการจัดกลุ่ม Jigsaw ก็คือการเขียนโครงร่างของโปรแกรมให้เป็นภาพรวมของโปรแกรมทั้งหมด และสามารถกำหนดได้ว่า องค์ประกอบนั้นๆ อยู่ในตำแหน่งใดของโปรแกรม หลังจากนั้นจึงทำการลงรายละเอียดขององค์ประกอบต่างๆ โดยใช้ลักษณะการคิดของการ Zoom In ซึ่งจะเหมือนกับการใช้โปรแกรมประมวลผลภาพต่างๆ ที่สามารถขยายภาพในส่วนที่ผู้ใช้งานต้องการเห็นรายละเอียดในส่วนนั้นๆ ได้ หรือโปรแกรมแผนที่ต่างๆ เช่น Google Earth หรือ Google Map ที่เราสามารถหารายละเอียดของแผนที่ในระดับกว้าง และระดับรายละเอียดได้มากมายหลายระดับ ซึ่งการมองภาพในระดับกว้าง จะทำให้เราเห็นภาพรวมของพื้นที่ทั้งหมดในแผนที่และถนนหลักต่างๆ และหาก Zoom เข้าไปในระดับลึกแล้ว เราจะเห็นรายละเอียดของถนนรอง ตรอก ซอยต่างๆ อย่างชัดเจน ภายในกรอบของการ Zoom นั้น สำหรับเทคนิคการคิดแบบ Zoom In จะเป็นแนวคิดที่ทำให้โปรแกรมเมอร์ทำภายหลังจากการออกแบบโปรแกรมให้เห็นภาพรวม และองค์ประกอบของโปรแกรมทั้งหมดแล้ว โดยจะค่อยๆ เพิ่มรายละเอียดในองค์ประกอบแต่ละส่วน จนได้รายละเอียดของโปรแกรมที่ถูกต้องและสมบูรณ์ได้

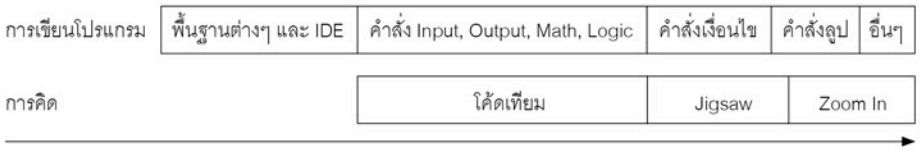
ลำดับการศึกษาการเขียนโปรแกรม

ในการศึกษาเกี่ยวกับการเขียนโปรแกรมที่ได้กล่าวไปแล้ว จะมีการศึกษาความรู้พื้นฐานเพื่อการเขียนโปรแกรม และการศึกษาด้านกระบวนการคิดเพื่อเขียนโปรแกรม ซึ่งแนวทางการศึกษาการเขียนโปรแกรมที่ดีควรจะต้องศึกษาในสองส่วนพร้อมๆ กัน ในส่วนของการศึกษาความรู้พื้นฐานเพื่อการเขียนโปรแกรมได้แก่ ความรู้พื้นฐานเกี่ยวกับคอมพิวเตอร์ เครื่องมือที่โปรแกรมเมอร์ใช้ในการเขียนโปรแกรม

รายละเอียดของโครงสร้างภาษาที่ใช้งาน และคำสั่งในโปรแกรมภาษาต่างๆ ซึ่ง ได้แก่ คำสั่งรับข้อมูล คำสั่งแสดงผล คำสั่งตรวจสอบเงื่อนไขการทำงาน คำสั่งการทำงานแบบวนลูป และคำสั่งอื่นๆ สำหรับการศึกษาด้านกระบวนการคิดเพื่อการเขียนโปรแกรมนั้น โปรแกรมเมอร์จะศึกษาเกี่ยวกับการนำเสนอความคิดของตนเองโดยใช้วิธีการต่างๆ เช่น การวาดรูป หรือการใช้โค้ดเทียม เพื่อใช้เป็นเครื่องมือในการนำเสนอความคิดในการแก้ปัญหาโจทย์ต่างๆ ทำให้ความคิดของโปรแกรมเมอร์กลายเป็นรูปธรรมโดยไม่อิงกับโครงสร้างของภาษาใดๆ หลังจากนั้นจึงศึกษาและฝึกฝนวิธีการในการแยกองค์ประกอบของโปรแกรมที่ออกแบบไว้เป็นส่วนๆ เพื่อให้เห็นภาพรวมโปรแกรม และองค์ประกอบของการทำงานทั้งหมดในโปรแกรม โดยใช้แนวคิดของการแยกกลุ่ม Jigsaw และศึกษาและฝึกฝนการเพิ่มรายละเอียดภายในโครงร่างของโปรแกรมให้มีรายละเอียดมากขึ้น โดยเพิ่มเติมรายละเอียดการทำงานขององค์ประกอบต่างๆ ที่แยกกลุ่มไว้ ให้เห็นภาพรายละเอียดกระบวนการทั้งหมดโดยใช้กระบวนการ Zoom In

เมื่อโปรแกรมเมอร์สามารถคิดแยกแยะองค์ประกอบของโปรแกรมที่จะเขียน พร้อมทั้งให้รายละเอียดของการทำงานทั้งหมดเรียบร้อยแล้ว จะสามารถเปลี่ยนความคิดที่ได้เขียนออกมาให้กลายเป็นโปรแกรมได้โดยใช้ความรู้เกี่ยวกับภาษาโปรแกรม ซึ่งการเปลี่ยนแปลงโครงร่างของโปรแกรมที่คิดวางแผนไว้ ให้กลายเป็นโปรแกรมที่สามารถสั่งให้คอมพิวเตอร์ทำงานได้ จะทำได้ไม่ยาก หากโปรแกรมเมอร์ได้เพิ่มรายละเอียดจนได้กระบวนการที่เป็นการทำงานพื้นฐานของคอมพิวเตอร์ทั้งหมดแล้ว

สำหรับลำดับในการศึกษาการเขียนโปรแกรมจะมีลักษณะการศึกษาแบบขนานคือ ศึกษาทั้งความรู้พื้นฐานเพื่อการเขียนโปรแกรม และการศึกษาด้านกระบวนการคิดเพื่อเขียนโปรแกรมไปพร้อมๆ กัน โดยในเบื้องต้น โปรแกรมเมอร์จะศึกษาพื้นฐานในการเขียนโปรแกรม ได้แก่ คอมพิวเตอร์พื้นฐาน และเครื่องมือที่ใช้ในการเขียนโปรแกรม หลังจากนั้นจึงศึกษาคำสั่งพื้นฐาน เช่น การรับค่าอินพุต การแสดงผล การคำนวณทางคณิตศาสตร์และตรรกศาสตร์ ในขั้นตอนนี้ควรศึกษาควบคู่กับการใช้โค้ดเทียม เนื่องจากมีเนื้อหาที่สอดคล้องกันอยู่มาก หลังจากนั้นจึงศึกษาคำสั่งการทำงานแบบมีเงื่อนไข และคำสั่งลูป ควบคู่กับการคิดแบบแยกกลุ่ม Jigsaw ซึ่งเนื้อหาของโจทย์ที่ใช้คำสั่งเงื่อนไข และคำสั่งลูปจะมีความซับซ้อนอยู่มาก จำเป็นต้องมีกระบวนการคิดในลักษณะการแยกองค์ประกอบ Jigsaw ร่วมด้วย จึงสามารถเขียนโปรแกรมได้โดยง่าย หลังจากนั้นจึงศึกษาการคิดเพื่อขยายความองค์ประกอบต่างๆ โดยกระบวนการของ Zoom In ควบคู่กับการประยุกต์ใช้คำสั่งเงื่อนไข และคำสั่งลูปร่วมกัน และการศึกษาคำสั่งอื่นๆ เช่น การใช้งานโครงสร้างข้อมูลต่างๆ เช่น array หรือ struct ในภาษา C และการใช้งานฟังก์ชั่น เป็นต้น เพราะการเขียนโปรแกรมที่ต้องใช้โครงสร้างข้อมูลที่เป็น array หรือ struct จะมีการคิดที่ซับซ้อนมากขึ้น โดยเฉพาะการเข้าถึงข้อมูล และใช้งานข้อมูลในโครงสร้างข้อมูลดังกล่าว มีวิธีการที่หลากหลาย ในการวางแผนเพื่อคิดกระบวนการทำงานของโปรแกรมที่มีความซับซ้อนสูงนี้ ต้องคำนึงถึงองค์ประกอบหลักๆ ก่อน แล้วจึงขยายความองค์ประกอบหลักนั้นภายหลัง



ในบทต่างๆ ต่อไปนี้ จะมีเนื้อหาเพื่อให้ผู้อ่านทำความเข้าใจเกี่ยวกับความรู้พื้นฐานเกี่ยวกับการเขียนโปรแกรม เหตุใดต้องเขียนโปรแกรมภาษา ชุดคำสั่งมาตรฐานของโปรแกรมภาษาต่างๆ ที่มีความคล้ายคลึงกัน การใช้งานโค้ดเทียม (Pseudo Code) กระบวนการคิดต่างๆ ที่ใช้ในการเขียนโปรแกรม โมเดลการคิดเพื่อแก้ปัญหาในการเขียนโปรแกรม อย่างเป็นขั้นตอน ซึ่งจะปูทางให้ผู้อ่านสามารถเข้าใจประเด็นต่างๆ ของการเขียนโปรแกรมได้อย่างเป็นลำดับขั้นตอน เรียงลำดับจากเนื้อหาที่ง่ายไปหาเนื้อหาที่ยาก โดยเนื้อหาส่วนใหญ่จะมุ่งเน้นที่กระบวนการคิดเพื่อเขียนโปรแกรมมากกว่าการให้รายละเอียดของชุดคำสั่งต่างๆ

เข้าใจคอมพิวเตอร์ก่อนเขียนโปรแกรม 2

ขั้นตอนของการออกแบบโปรแกรมให้ทำงานตรงตามที่โจทย์ต้องการนั้น โปรแกรมเมอร์ต้องหากระบวนการต่างๆ ที่สามารถร้อยเรียงให้ผลลัพธ์ของการทำงานตรงตามความต้องการของโจทย์ปัญหานั้นๆ ซึ่งกระบวนการที่โปรแกรมเมอร์คิดขึ้นมาจะต้องไม่ขัดต่อความสามารถของคอมพิวเตอร์ หากโปรแกรมเมอร์คิดถึงกระบวนการที่คอมพิวเตอร์ไม่สามารถทำงานได้ หรือเป็นกระบวนการที่เป็นข้อจำกัดของคอมพิวเตอร์ โปรแกรมเมอร์จะไม่สามารถเปลี่ยนกระบวนการที่คิดขึ้นให้กลายเป็นโปรแกรมได้เลย ดังนั้น โปรแกรมเมอร์จึงหลีกเลี่ยงไม่ได้ที่ต้องมีความรู้พื้นฐานเกี่ยวกับคอมพิวเตอร์ ต้องทราบถึงความสามารถของคอมพิวเตอร์ รวมถึงข้อกำหนดและข้อจำกัดต่างๆ ของคอมพิวเตอร์ โดยความรู้ที่ได้จะเป็นกรอบความคิดสำคัญในการออกแบบกระบวนการทำงานของโปรแกรมต่างๆ ได้

ความรู้พื้นฐานเกี่ยวกับคอมพิวเตอร์

คอมพิวเตอร์คือ อุปกรณ์อิเล็กทรอนิกส์ที่มีความสามารถในการคำนวณทั้งทางด้านคณิตศาสตร์ และทางด้านตรรกศาสตร์ สามารถทำงานตามที่สั่งการอย่างเป็นขั้นตอนได้ โดยองค์ประกอบของคอมพิวเตอร์ได้แก่

- **อุปกรณ์อินพุต** คือ อุปกรณ์ที่ผู้ใช้งานใช้ในการป้อนข้อมูลให้กับระบบคอมพิวเตอร์ เช่น คีย์บอร์ด ใช้สำหรับป้อนข้อมูลอักษรต่างๆ เมาส์ใช้สำหรับป้อนข้อมูลตำแหน่งของตัวชี้ในหน้าจอ และข้อมูลการคลิกปุ่ม เป็นต้น
- **อุปกรณ์เอาต์พุต** คือ อุปกรณ์ที่คอมพิวเตอร์ใช้ในการแสดงผลข้อมูลต่างๆ ในระบบให้กับผู้ใช้งานได้ทราบ เช่น หน้าจอคอมพิวเตอร์ที่แสดงผลลัพธ์ของการประมวลผลตามคำสั่งของคอมพิวเตอร์ และเครื่องพิมพ์ที่สามารถนำเอาข้อมูลมาแสดงผลลงในกระดาษได้

- **หน่วยประมวลผลกลาง** คือ อุปกรณ์ขนาดเล็กที่เปรียบได้กับสมองของคอมพิวเตอร์ ซึ่งใช้ในการประมวลผลทางคณิตศาสตร์ และตรรกศาสตร์
- **หน่วยเก็บข้อมูล** คือ อุปกรณ์ที่ใช้ในการเก็บข้อมูล หรือคำสั่งต่างๆ ของคอมพิวเตอร์ โดยหน่วยเก็บข้อมูลที่สำคัญ ได้แก่ หน่วยความจำที่เป็นพื้นที่หลักในการทำงานของคอมพิวเตอร์ ซึ่งจะเก็บข้อมูลที่คอมพิวเตอร์ต้องใช้งาน และคำสั่งต่างๆ ที่คอมพิวเตอร์จะดำเนินการ ฮาร์ดดิสก์เป็นพื้นที่หลักในการจัดเก็บข้อมูล ซึ่งจากเทคโนโลยีที่พัฒนามากขึ้นทำให้ฮาร์ดดิสก์มีความเร็วในการทำงาน และพื้นที่ในการจัดเก็บเพิ่มมากขึ้นเรื่อยๆ USB Drive เป็นอุปกรณ์พกพาที่ทำให้ผู้ใช้งานสามารถเก็บข้อมูล และพกพาไปใช้งานได้โดยง่าย มีขนาดเล็ก มีพื้นที่การจัดเก็บเพิ่มมากขึ้นตามเทคโนโลยีที่พัฒนาขึ้นเรื่อยๆ รวมถึงอุปกรณ์อื่นๆ เช่น CD/DVD เป็นต้น

จากองค์ประกอบต่างๆ ของคอมพิวเตอร์ ทำให้เราพอจะทราบขอบเขตความสามารถของคอมพิวเตอร์ ซึ่งจะสามารถโต้ตอบกับผู้ใช้ได้ในช่องทางของอุปกรณ์อินพุตและเอาต์พุต สามารถประมวลผลการทำงานได้โดยหน่วยประมวลผลกลาง และเก็บข้อมูลได้ในหน่วยเก็บข้อมูลเท่านั้น สำหรับคอมพิวเตอร์นั้นจะมีประโยชน์กับมนุษย์ก็ต่อเมื่อองค์ประกอบต่างๆ ในคอมพิวเตอร์สามารถทำงานสอดประสานกันได้เป็นอย่างดี และให้ผลลัพธ์การทำงานที่ถูกต้องรวดเร็ว และให้ผลลัพธ์ตรงตามที่ผู้ใช้งานต้องการ ซึ่งการที่คอมพิวเตอร์จะมีความสามารถเช่นนั้นได้ จำเป็นต้องมีการสั่งการให้คอมพิวเตอร์ทำงานตามที่ผู้ใช้งานต้องการ ซึ่งวิธีการในการสั่งการคอมพิวเตอร์ให้ทำงานนี้เองคือ **"การเขียนโปรแกรมคอมพิวเตอร์"**

จากองค์ประกอบของคอมพิวเตอร์ที่ได้กล่าวไปแล้ว เราสามารถสรุปถึงความสามารถของคอมพิวเตอร์ที่สามารถทำงานได้ 5 รูปแบบหลัก คือ

1. **การรับข้อมูลจากผู้ใช้งาน** เนื่องจากคอมพิวเตอร์เป็นอุปกรณ์เพื่อการคำนวณสำหรับคำถามต่างๆ ของมนุษย์ ในการคำนวณจึงต้องมีการรับข้อมูลจากผู้ใช้งานในหลากหลายรูปแบบ และหลากหลายช่องทาง เพื่อให้ผู้ใช้งานคอมพิวเตอร์สามารถใช้งานได้อย่างง่ายดาย การป้อนข้อมูลให้คอมพิวเตอร์ทำงานจะสามารถป้อนข้อมูลผ่านอุปกรณ์อินพุตต่างๆ เช่น คีย์บอร์ด เมาส์ กล้อง หรืออุปกรณ์รับข้อมูลอื่นๆ
2. **การแสดงผลข้อมูลแก่ผู้ใช้งาน** การแสดงผลข้อมูลให้ผู้ใช้งานเป็นการทำงานสำหรับตอบสนองต่อผู้ใช้งานในหลากหลายรูปแบบ การแสดงผลข้อมูลที่มีการใช้งานมากก็คือ หน้าจอ และสำหรับการแสดงผลในรูปแบบอื่นๆ เช่น ลำโพง (Speaker), เครื่องพิมพ์ (Printer) เป็นต้น
3. **การคำนวณ และการดำเนินการทางตรรกะ** เป็นที่รู้กันดีอยู่แล้วว่า คอมพิวเตอร์คือเครื่องคำนวณที่อเนกประสงค์ ผู้ใช้งานสามารถสั่งให้คอมพิวเตอร์ทำการคำนวณค่าต่างๆ ได้ ทั้งนี้ขึ้นอยู่กับโปรแกรมที่ทำงานในคอมพิวเตอร์นั้นๆ มีการคำนวณในลักษณะใด โดยความสามารถในการคำนวณของคอมพิวเตอร์ไม่ได้คำนวณได้มากกว่า หรือดีกว่ามนุษย์เลย แต่ความสามารถที่โดดเด่นในการคำนวณของคอมพิวเตอร์อยู่ที่การคำนวณที่ "รวดเร็ว" กว่ามนุษย์เท่านั้น

นอกจากนี้ การคำนวณเชิงตรรกะเป็นความสามารถของคอมพิวเตอร์ที่ทำให้คอมพิวเตอร์ทำงานเป็นกระบวนการเป็นขั้นตอน และทำงานอย่างมีเงื่อนไขได้ เนื่องจากคอมพิวเตอร์ทำงานโดยยึดตรรกะเป็นเกณฑ์ และไม่มีความรู้สึก หรือการคาดการณ์ใดๆ ทั้งสิ้น ทำให้ความเป็นตรรกะของคอมพิวเตอร์มีมากกว่ามนุษย์มาก จากการทำงานดังกล่าวทำให้การทำงานของคอมพิวเตอร์มีความถูกต้อง แม่นยำ และผลลัพธ์การทำงานหลายๆ อย่างไม่สามารถปฏิเสธได้ จนทำให้คนทั่วไปมองว่า คอมพิวเตอร์ฉลาดกว่ามนุษย์

4. **การควบคุมการทำงานโดยใช้เงื่อนไขการทำงาน** ในการทำงานโดยทั่วไปของระบบคอมพิวเตอร์จะทำงานอย่างเป็นลำดับขั้นตอน แต่ก็มีการทำงานบางอย่างที่จะทำงานได้ ก็ต่อเมื่อถูกต้องตามเงื่อนไขของสภาพแวดล้อมบางอย่าง เช่น ในการให้คอมพิวเตอร์รักษาความปลอดภัย การจะสั่งให้ไฟสัญญาณติดได้ก็ต่อเมื่อมีเงื่อนไขคือ มีการบุกรุก หรือมีเหตุให้ไฟสัญญาณต้องติดเท่านั้น การทำงานดังกล่าวคอมพิวเตอร์จะต้องกำหนดเงื่อนไขจากสภาวะที่ต้องการเพื่อสั่งให้ระบบทำงาน ดังนั้น ในระบบคอมพิวเตอร์จะมีความสามารถหนึ่งคือ สามารถควบคุมการทำงานให้เป็นแบบมีเงื่อนไขการทำงานได้ โดยลำดับของการทำงานในระบบคอมพิวเตอร์ที่ไม่เป็นแบบลำดับจะมีอยู่สองรูปแบบคือ “การทำงานแบบมีเงื่อนไข” ซึ่งจะทำงานก็ต่อเมื่อเงื่อนไขบางอย่างเป็นจริงและ “การทำงานแบบวนซ้ำ” โดยจะทำงานในกระบวนการบางขั้นตอนซ้ำๆ เมื่อเงื่อนไขบางอย่างเป็นจริง

5. **การเก็บข้อมูลในหน่วยเก็บข้อมูล และการดึงข้อมูลในหน่วยเก็บข้อมูลเพื่อใช้งาน** ในโลกของคณิตศาสตร์การคำนวณสูตร สมการ หรือการวิเคราะห์ข้อมูลต่างๆ จำเป็นต้องมีตัวแปร หรือค่าคงที่ที่ใช้ในการคำนวณ ในระบบคอมพิวเตอร์ที่มีการใช้งานทางด้านคณิตศาสตร์เป็นหลักจึงจำเป็นต้องมีการเก็บข้อมูลตัวแปร หรือค่าคงที่ต่างๆ ไว้สำหรับการคำนวณค่าต่างๆ ตามที่โปรแกรมกำหนดไว้ สำหรับการเก็บข้อมูลคอมพิวเตอร์จะเก็บข้อมูลในหน่วยความจำ หรือหน่วยเก็บข้อมูลอื่นๆ เช่น ฮาร์ดดิสก์ (Hard disk), CD, DVD เป็นต้น

จากองค์ประกอบของคอมพิวเตอร์ และความสามารถในการทำงานของคอมพิวเตอร์ เราจะสามารถวิเคราะห์ได้ว่า สิ่งที่คอมพิวเตอร์ทำได้ล้วนแล้วแต่เป็นสิ่งที่มนุษย์ทำได้ทั้งสิ้น ไม่ว่าจะเป็นความสามารถคิดคำนวณและประมวลผล เพียงแต่การคิดคำนวณและประมวลผลนั้นสามารถทำได้รวดเร็วกว่ามนุษย์เท่านั้น นอกจากนี้คอมพิวเตอร์ยังไม่ใช่อุปกรณ์วิเศษที่สามารถตอบคำถามได้ทุกๆ คำถาม หรือสามารถทำงานได้ทุกๆ อย่าง คอมพิวเตอร์เป็นเพียงอุปกรณ์ที่ต้องทำงานตามคำสั่งที่มนุษย์เป็นผู้สั่งให้ทำอย่างเชื่อฟัง การทำงานของคอมพิวเตอร์จึงไม่มีการทำงานที่ผิดพลาด หากจะมีการคำนวณใดๆ ที่ผิดพลาด จะเป็นการผิดพลาดจากโปรแกรมเมอร์ที่สั่งการเท่านั้น

สาเหตุที่ต้องเขียนโปรแกรมภาษาเพื่อสั่งงานคอมพิวเตอร์

จากองค์ประกอบของคอมพิวเตอร์ที่มีอุปกรณ์อิเล็กทรอนิกส์ ที่มีความซับซ้อนมาประกอบกันและทำงานร่วมกัน การสั่งการให้อุปกรณ์อิเล็กทรอนิกส์ต่างๆ ทำงานร่วมกันได้ต้องใช้สัญญาณดิจิตอลในการสั่งการ การออกแบบระบบควบคุมสัญญาณดิจิตอลทั้งระบบในคอมพิวเตอร์จะมองในรูปแบบสัญญาณ ON/OFF ซึ่งทำให้โปรแกรมที่สามารถสั่งการคอมพิวเตอร์ให้ทำงานตามที่ต้องการได้ จะอยู่ในรูปของชุดตัวเลขที่สามารถแทนสัญญาณ ON/OFF ได้ นั่นก็คือตัวเลขฐานสอง โดยเราจะเรียกชุดของตัวเลขฐานสองที่สามารถสั่งงาน และควบคุมการทำงานของคอมพิวเตอร์ได้ว่า **"รหัสเครื่อง"(Machine Code)**

ในการสั่งการคอมพิวเตอร์จำเป็นต้องมีการสร้างรหัสเครื่องที่นำสัญญาณไปควบคุมอุปกรณ์ต่างๆ ให้ทำงานสอดคล้องกับการทำงานที่ผู้ใช้งานต้องการ ซึ่งในการสร้างรหัสเครื่องโดยโปรแกรมเมอร์นั้นจะทำได้ยากมาก เนื่องจากต้องเป็นคนที่มีความเข้าใจในรหัสดังกล่าวเท่านั้น จึงจะสามารถออกแบบรหัสเครื่องได้ ซึ่งก็คือคนออกแบบคอมพิวเตอร์นั้นเพียงกลุ่มเดียว อีกทั้งการศึกษาเรียนรู้รหัสเครื่องทำได้ยาก เพราะต้องอาศัยความรู้ทางเทคนิคเชิงลึก รวมถึงความเข้าใจสถาปัตยกรรมระบบคอมพิวเตอร์นั้นๆ โดยละเอียด รวมถึงต้องมีความเชี่ยวชาญจึงจะสามารถเขียนรหัสเครื่องที่ถูกต้องได้ จากสาเหตุดังกล่าว ทำให้เกิดปัญหาที่บุคคลทั่วไปไม่สามารถเขียนโปรแกรมเพื่อสั่งการให้คอมพิวเตอร์ทำงานได้เลย

ดังนั้น เพื่อให้บุคคลทั่วไปสามารถสั่งงานคอมพิวเตอร์ได้โดยง่าย และมีความหลากหลายในการใช้งานคอมพิวเตอร์มากขึ้น จึงมีความต้องการให้มีรูปแบบของการสร้างโปรแกรมที่เอื้อต่อคนทั่วๆ ไป ที่ไม่มีความรู้ด้านคอมพิวเตอร์มากนัก ให้สามารถเขียนโปรแกรมเพื่อสั่งการคอมพิวเตอร์ให้ทำงานตามที่ต้องการได้ สำหรับทางออกของปัญหานี้ที่ได้มีการใช้งานกันเรื่อยมาจนถึงปัจจุบันคือ การใช้รูปแบบของล่ามแปลภาษา โดยเริ่มต้นจากการสร้างโค้ดของโปรแกรมที่ถูกต้องตามหลักไวยากรณ์ที่กำหนดขึ้น หรือซึ่งเรียกว่า **"โปรแกรมภาษา"** ที่บุคคลทั่วไปสามารถศึกษาได้โดยง่าย พร้อมทั้งสร้างล่ามแปลภาษาที่สามารถแปลโค้ดคำสั่งในโปรแกรมภาษานั้นๆ ให้กลายเป็นรหัสเครื่อง ซึ่งคอมพิวเตอร์สามารถเข้าใจการทำงานได้ ในกระบวนการสร้างโปรแกรมคอมพิวเตอร์โดยใช้ล่ามแปลภาษา จะเรียกโปรแกรมที่ทำหน้าที่เป็นล่ามแปลภาษาว่า **"คอมไพเลอร์"** หลังจากที่มีรูปแบบการพัฒนาซอฟต์แวร์โดยใช้คอมไพเลอร์เป็นตัวกลางในการทำงานแล้ว จึงมีการออกแบบภาษาต่างๆ ขึ้นมามากมาย เช่น ภาษา FORTRAN, ภาษา Basic, ภาษา C และภาษาอื่นๆ โดยแต่ละภาษาจะมีคอมไพเลอร์ของตัวเองที่ทำหน้าที่แปลภาษานั้นๆ ให้กลายเป็นรหัสเครื่องได้

เมื่อกระบวนการสร้างโปรแกรมคอมพิวเตอร์ได้เปลี่ยนรูปแบบเป็นการเขียนโปรแกรม โดยเริ่มต้นที่การเขียนโค้ดโปรแกรมภาษาต่างๆ แล้วใช้คอมไพเลอร์แปลให้กลายเป็นรหัสเครื่อง ทำให้กระบวนการที่จะศึกษา และสร้างโปรแกรมคอมพิวเตอร์เปลี่ยนไปเช่นกัน โดยผู้ที่ต้องการเขียนโปรแกรมสั่งการให้

คอมพิวเตอร์ทำงานตามที่ตนเองต้องการ จะเริ่มจากการศึกษาการเขียนโปรแกรมภาษาต่างๆ แทนการศึกษาสถาปัตยกรรม และการทำงานเชิงลึกของคอมพิวเตอร์ แล้วผลักภาระทั้งหมดให้เป็นหน้าที่ของคอมไพเลอร์ในการแปลภาษาต่างๆ ให้กลายเป็นรหัสเครื่อง ทำให้ในปัจจุบันการศึกษาเพื่อสั่งการให้คอมพิวเตอร์ทำงานตามที่ต้องการ จึงมาเริ่มกันที่การเขียนโปรแกรม ไม่ใช่การศึกษาสถาปัตยกรรมของคอมพิวเตอร์ ส่งผลให้โปรแกรมเมอร์ไม่จำเป็นต้องจำกัดอยู่เฉพาะบุคลากรที่เรียนในสายเทคโนโลยีสารสนเทศเท่านั้น

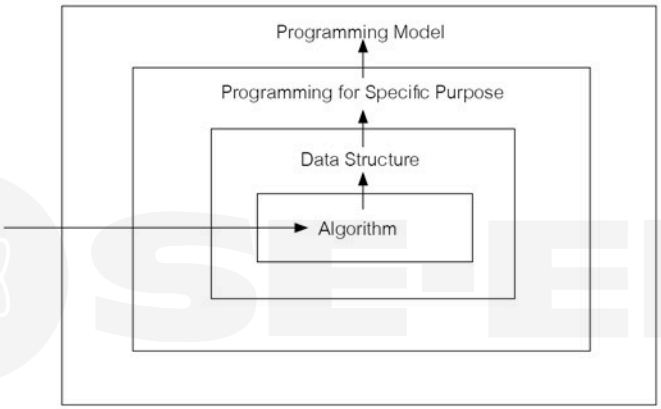
จากสาเหตุที่โปรแกรมเมอร์ต้องพึ่งพาคอมไพเลอร์เพื่อแปลจากโค้ดโปรแกรมภาษาให้กลายเป็นรหัสเครื่องนั้น ทำให้เราต้องเขียนโปรแกรมภาษาต่างๆ ให้สอดคล้องกับคอมไพเลอร์ แต่ข้อกำหนดของคอมไพเลอร์ก็ไม่ได้กำหนดลึกลงไปถึงวิธีการสร้างโค้ดของโปรแกรมภาษาต่างๆ ว่าต้องสร้างโดยวิธีใด เมื่อพิจารณาถึงความต้องการของคอมไพเลอร์ ซึ่งต้องการเฉพาะเท็กซ์ไฟล์ที่บรรจุโค้ดโปรแกรมภาษาที่ตรงตามหลักไวยากรณ์ของภาษานั้นๆ โปรแกรมเมอร์จึงสามารถสร้างโปรแกรมภาษาได้โดยใช้โปรแกรมใดๆ ที่สามารถสร้างเท็กซ์ไฟล์ได้ เช่น Notepad, WordPad, Microsoft Office, ฯลฯ ก็สามารถสร้างไฟล์โปรแกรมภาษาที่ต้องการได้แล้ว

แต่ทว่า การพัฒนาโปรแกรมอย่างเป็นระบบจะไม่ได้มีเฉพาะการสร้างไฟล์ และการคอมไพล์ให้กลายเป็นภาษาเครื่องเท่านั้น ยังจำเป็นต้องมีกระบวนการอื่นๆ เช่น การจัดเก็บข้อมูลภาษา การเรียกให้โปรแกรมทำงานตามความต้องการ การดีบั๊กเพื่อตรวจสอบการเปลี่ยนแปลงของค่าต่างๆ ในโปรแกรม สำหรับการแก้ไขความผิดพลาดต่างๆ ในโปรแกรม การตั้งค่าการแสดงผล และการดำเนินการเพื่อสร้างสภาพแวดล้อมในการพัฒนาระบบให้เหมาะสมด้วย ดังนั้น การเขียนโปรแกรมโดยใช้ Notepad, WordPad, Microsoft Office และโปรแกรมสำหรับสร้างเท็กซ์ไฟล์อื่นๆ คงไม่เหมาะสมนัก และมีการสร้างโปรแกรมสำหรับการเขียนโปรแกรมขึ้นมาโดยเฉพาะ ที่เราเรียกว่า **Integrated Development Environments (IDE)** เช่น Dev-C++, Visual Studio, C++ Builder, Turbo C++ Professional สำหรับภาษา C หรือ Eclipse, NetBeans, JBuilder และ Oracle JDeveloper สำหรับภาษา Java ซึ่งจะมีฟังก์ชั่นต่างๆ ที่จำเป็นสำหรับการพัฒนาโปรแกรมคอมพิวเตอร์ไว้ให้โปรแกรมเมอร์เรียกใช้งานได้โดยสะดวก

ลำดับการศึกษาการเขียนโปรแกรม

ในการเขียนโปรแกรม หลายๆ คนอาจเห็นว่าเป็นเรื่องใหญ่ จะเขียนโปรแกรมได้อย่างไร โดยเฉพาะโปรแกรมขนาดใหญ่ที่มีผู้ใช้งานโปรแกรมร่วมกันหลายๆ คน หรือโปรแกรมที่มีความสามารถสูงๆ ในการศึกษาเพื่อเขียนโปรแกรมจะมีลำดับการศึกษาเรียงจากง่ายไปยาก จะเริ่มจากการปูพื้นฐานการเขียนโปรแกรม แล้วจึงขยายขอบเขตความเข้าใจไปเรื่อยๆ จนศึกษาโมเดลการพัฒนาซอฟต์แวร์ขนาดใหญ่ ซึ่งลำดับการศึกษาเกี่ยวกับการเขียนโปรแกรมควรมีดังนี้ คือ

1. การศึกษาการเขียนโปรแกรมโดยเน้นด้านกระบวนการทำงานของโปรแกรม (Algorithm)
2. การศึกษาการเขียนโปรแกรมโดยเน้นที่การใช้งานโครงสร้างข้อมูล (Data Structure)
3. การศึกษาการเขียนโปรแกรมสำหรับการทำงานเฉพาะ เช่น การเขียนโปรแกรมติดต่อฐานข้อมูล เขียนโปรแกรมเพื่อทำงานผ่านเครือข่าย
4. การศึกษาเกี่ยวกับโมเดลการเขียนโปรแกรม เช่น Object Oriented Programming, การใช้งานคอมโพเนนต์ต่างๆ, การใช้งาน Unified Modeling Language, การดำเนินการเกี่ยวกับ Software Engineering



สาเหตุที่ต้องมีลำดับการศึกษาเป็นลำดับ เนื่องจากการเรียนรู้ด้านการพัฒนาซอฟต์แวร์นี้มีความเกี่ยวเนื่องกัน โดยการศึกษาด้านอัลกอริทึมจะเป็นพื้นฐานการเขียนโปรแกรมที่สำคัญที่สุด โปรแกรมเมอร์ต้องมีความเข้าใจถึงกระบวนการทำงานต่างๆ เพื่อให้สามารถสั่งการให้คอมพิวเตอร์ทำงานอย่างเป็นขั้นตอนได้อย่างมีประสิทธิภาพ เมื่อโปรแกรมเมอร์สามารถออกแบบกระบวนการทำงาน และสั่งการคอมพิวเตอร์ได้แล้ว จึงเริ่มศึกษาด้านโครงสร้างข้อมูลซึ่งมีเนื้อหาเกี่ยวกับการออกแบบโครงสร้างข้อมูลเพื่อสนับสนุนการเขียนโปรแกรมที่มีความต้องการตัวเก็บข้อมูลชนิดพิเศษ เพื่อให้สามารถคิด และเขียนโปรแกรมทำงานได้ง่ายขึ้น รวมถึงมีประสิทธิภาพการทำงานได้ดีขึ้น ซึ่งขั้นตอนนี้จำเป็นต้องอาศัยความรู้พื้นฐานเกี่ยวกับอัลกอริทึมร่วมด้วย

หลังจากนั้นจึงพัฒนาความสามารถในการเขียนโปรแกรมให้มากขึ้น โดยใช้ความรู้เกี่ยวกับอัลกอริทึม และโครงสร้างข้อมูล มาต่อยอดโดยศึกษาการเขียนโปรแกรมในงานเฉพาะด้าน เช่น การเขียนโปรแกรมเพื่อติดต่อระบบเครือข่าย หรือติดต่อกับระบบฐานข้อมูล หลังจากนั้นจึงต่อยอดความรู้อีก

คอมพิวเตอร์และการเขียนโปรแกรม
(Computers and Programming)
C และ Java

"ผม" เคยแปลกใจว่า ทำไมเขียนโปรแกรมมิ่งทั้งหลายจึงมักจะสอนเขียนโปรแกรมไม่รู้เรื่อง จนได้มาสอนพื้นฐานการเขียนโปรแกรมเอง จึงพบว่า การสอนแต่คำสั่งของภาษาคอมพิวเตอร์ไม่ได้ช่วยให้เขียนโปรแกรมได้ เช่นเดียวกับการสร้างรถยนต์ซักคัน ถ้ามัวแต่เรียนว่าไขควงใช้งานยังไง รถยนต์คันแรกคงไม่เกิดขึ้น

เนื้อหาประกอบด้วย

เกี่ยวกับผู้เขียน **ธนัญชัย ตรีภาค**

จบการศึกษาระดับปริญญาตรี สาขาวิศวกรรมคอมพิวเตอร์ จากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง และได้รับทุนพัฒนาอาจารย์ เพื่อศึกษาต่อในระดับปริญญาโท สาขาวิศวกรรมคอมพิวเตอร์ จากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปัจจุบันดำรงตำแหน่งอาจารย์ประจำที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง มีประสบการณ์ในการสอนวิชาหลักการเขียนโปรแกรมมากกว่า 5 ปี

หนังสือเล่มนี้สำหรับ
ผู้เริ่มต้น ระดับกลาง ระดับสูง
คอมพิวเตอร์/การเขียนโปรแกรม

ISBN 978-616-08-0067-4
9 786160 800674
195 บาท

www.se-ed.com