แผนบริหารการสอนประจำบทที่ 1

โปรแกรมคอมพิวเตอร์

หัวข้อประจำบท

1. ระบบคอมพิวเตอร์
2. โปรแกรมคอมพิวเตอร์
3. ภาษาคอมพิวเตอร์
4. ตัวแปลภาษาคอมพิวเตอร์
5. รหัสเทียม
6. ขั้นตอนการพัฒนาโปรแกรม
7. แผนภาพแสดงลำดับขั้นตอนการทำงาน
8. การวิเคราะห์และออกแบบโปรแกรม

วัตถุประสงค์เชิงพฤติกรรม

1. เพื่อให้ผู้เรียนสามารถอธิบายการทำงานของระบบคอมพิวเตอร์
2. เพื่อให้ผู้เรียนบอกความหมาย และการทำงานโปรแกรมคอมพิวเตอร์
3. เพื่อให้ผู้เรียนสามารถอธิบายภาษาคอมพิวเตอร์ และสามารถบอกความแตกต่างระหว่างภาษาคอมพิวเตอร์ในระดับต่าง ๆ ได้
4. เพื่อให้ผู้เรียนสามารถอธิบายตัวแปลภาษา และสามารถบอกความแตกต่างระหว่างตัวแปลภาษาแต่ละประเภทได้
5. เพื่อให้ผู้เรียนสามารถเขียนรหัสเทียมได้
6. เพื่อให้ผู้เรียนสามารถอธิบายขั้นตอนการพัฒนาโปรแกรมได้
7. เพื่อให้ผู้เรียนสามารถเขียนแผนภาพแสดงลำดับขั้นตอนการทำงานของโปรแกรม และสามารถนำไปใช้ได้อย่างถูกต้อง
8. เพื่อให้ผู้เรียนสามารถวิเคราะห์และออกแบบโปรแกรม

วิธีการสอนและกิจกรรม

1. ผู้สอนบรรยายในชั้นเรียน และโปรแกรมนำเสนอ ตามหัวข้อเนื้อหาในเอกสารประกอบการสอน
2. ผู้เรียนทำแบบฝึกหัดท้ายบท และฝึกเขียนโปรแกรมด้วยคอมพิวเตอร์

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน วิชา หลักการเขียนโปรแกรมคอมพิวเตอร์
2. โปรแกรม PowerPoint

การวัดผลและการประเมินผล

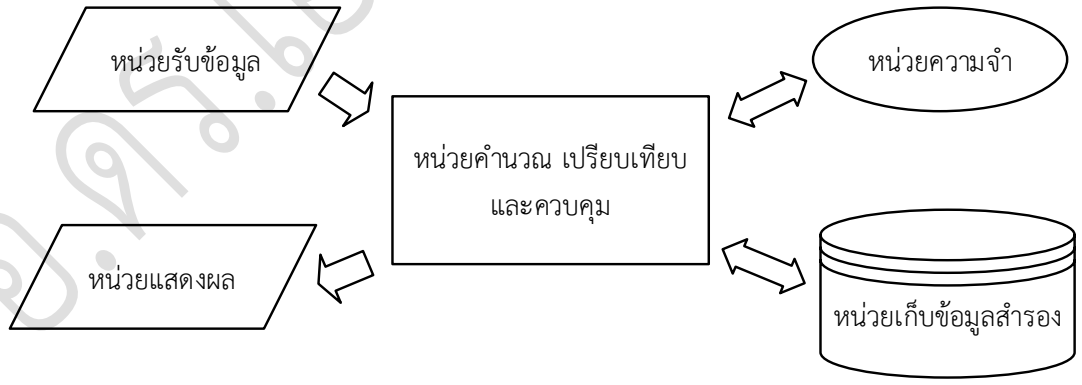
1. สังเกตจากการอภิปราย การตอบคำถาม และซักถามระหว่างเรียน
2. การปฏิบัติบนเครื่องคอมพิวเตอร์
3. การทำแบบฝึกหัดท้ายบท

บทที่ 1
โปรแกรมคอมพิวเตอร์

การทำงานของระบบคอมพิวเตอร์ประกอบไปด้วย การทำงานประสานกันระหว่างส่วนเครื่องของคอมพิวเตอร์ เรียกว่า “ฮาร์ดแวร์ (Hardware)” และส่วนชุดคำสั่ง สั่งงานให้คอมพิวเตอร์ทำงาน เรียกว่า “ซอฟต์แวร์ (Software)” ส่วนชุดคำสั่งนี้ เกิดจากการเขียนจากโปรแกรมที่เรียกว่า “ภาษาโปรแกรม (Programing Language)” (ราชบัณฑิตยสภา, 2564) ภาษาโปรแกรมคอมพิวเตอร์มีรูปแบบและการใช้งานแตกต่างกันไปตามวิวัฒนาการและจุดประสงค์ของการเขียนโปรแกรมคอมพิวเตอร์ เพื่อตอบสนองและสนับการทำงานของระบบในปัจจุบัน

ระบบคอมพิวเตอร์

คอมพิวเตอร์ (Computer) (ราชบัณฑิตยสภา, 2564) หมายถึง เครื่องกลที่สามารถรับข้อมูลผ่านอุปกรณ์ต่าง ๆ ที่เชื่อมต่อกับคอมพิวเตอร์และนำข้อมูลที่ได้ไปประมวลผลเป็นขั้นตอน ผลลัพธ์ที่ได้อยู่ในรูปแบบสัญญาณทางอิเล็กทรอนิกส์ ที่แปลงเป็นข้อมูลแสดงให้มนุษย์ได้รับรู้และเข้าใจบนเครื่องมือหรืออุปกรณ์ต่าง ๆ (Evans, 2011) ระบบของคอมพิวเตอร์ประกอบด้วยการทำงานดังต่อไปนี้



ภาพที่ 1.1 ส่วนประกอบของระบบคอมพิวเตอร์

ที่มา : Norton (2008)

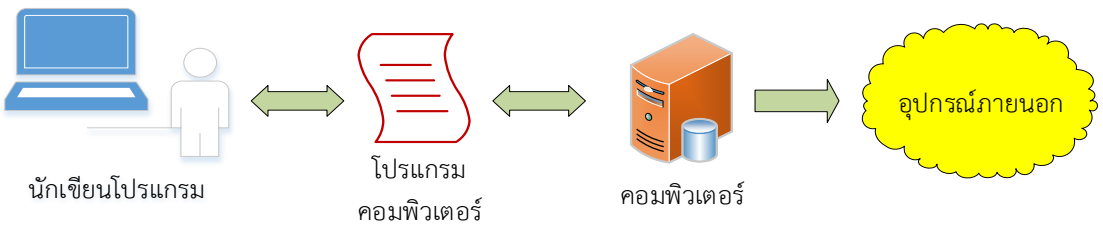
ภาพที่ 1.1 ส่วนประกอบการทำงานของระบบคอมพิวเตอร์ ประกอบไปด้วย หน่วยรับข้อมูล ทำหน้าที่ในการรับข้อมูลจากอุปกรณ์ที่รับข้อมูลหรือคำสั่ง เช่น แผงแป้นอักขระ (Keyboard) เมาส์ (Mouse) เป็นต้น ส่งข้อมูลไปยังหน่วยประมวลผลกลาง (Central Processing Unit) เพื่อทำการควบคุมการประมวลผล การประมวลผลแบ่งออกเป็น 2 ประเภท ประกอบไปด้วยการคำนวณและเปรียบเทียบ เมื่อประมวลเสร็จสิ้นข้อมูลนั้นจะถูกส่งไปยังหน่วยแสดงผล (Output Unit) แสดงผลบนอุปกรณ์ที่ทำหน้าที่แสดงผล เช่น จอแสดงผล เครื่องพิมพ์ เสียงทางลำโพง เป็นต้น ระหว่างที่มีการทำงาน ข้อมูลอาจถูกเก็บไว้ในหน่วยความจำ (Memory Unit) เพื่อใช้งานชั่วคราว เมื่อปิดเครื่องข้อมูลก็จะหายไป แต่อาจมีข้อมูลอีกประเภทที่ต้องการเก็บไว้ใช้งานในภายหลัง จะถูกเก็บไว้ในหน่วยเก็บข้อมูลสำรอง (Storage Unit) (ราชบัณฑิตยสภา, 2564)

โปรแกรมคอมพิวเตอร์

การทำงานของระบบคอมพิวเตอร์นั้น คอมพิวเตอร์จะสามารถทำงานได้จะต้องมีข้อมูลภายในระบบ ซึ่งข้อมูลเหล่านี้จะถูกควบคุมการทำงานในการรับส่งข้อมูลผ่านโปรแกรมคอมพิวเตอร์ ซึ่งมีผู้ให้ความหมายของโปรแกรมคอมพิวเตอร์ ดังต่อไปนี้

โปรแกรมคอมพิวเตอร์ (Computer Programming) หมายถึง ชนิดของตัวอักษรที่ถูกสร้างขึ้นโดยนักเขียนโปรแกรม (Programmer) (ราชบัณฑิตยสภา, 2564) จุดมุ่งหมายเพื่อให้คอมพิวเตอร์ทำงานได้ตามที่ต้องการและโปรแกรมยังเป็นข้อมูลที่อยู่ในหน่วยความจำของคอมพิวเตอร์ (Haverbeke, 2014) นอกเหนือจากนี้ โปรแกรมคอมพิวเตอร์ มีการสร้างคำสั่ง เพื่อให้เครื่องคอมพิวเตอร์ทำงานที่เป็นประโยชน์ต่อผู้ใช้งานและเนื่องจากคอมพิวเตอร์สามารถประมวลผลได้เป็นจำนวนมาก จึงมีการนำเอาไปใช้ในงานในด้านต่าง ๆ เช่น ด้านวิศวกรรม ด้านวิทยาศาสตร์และด้านงานศิลปะ เป็นต้น (Evans, 2011)

การให้ความหมายของโปรแกรมคอมพิวเตอร์ดังกล่าว โปรแกรมคอมพิวเตอร์มีลักษณะคำสั่งที่ถูกสร้างขึ้น ทำหน้าที่เพื่อควบคุมการทำงานของคอมพิวเตอร์ให้สามารถทำงานได้ตามจุดประสงค์ที่ต้องการที่ต้องการนำไปใช้เพื่อแก้ปัญหาการทำงานในรูปแบบเดิม โปรแกรมคอมพิวเตอร์สามารถอธิบายเป็นแผนภาพการทำงานของโปรแกรมคอมพิวเตอร์ ดังภาพที่ 1.2



ภาพที่ 1.2 การทำงานของโปรแกรมคอมพิวเตอร์

ภาพที่ 1.2 การทำงานของโปรแกรมคอมพิวเตอร์ เริ่มต้นจากนักเขียนโปรแกรมสร้างชุดคำสั่งในลักษณะของโปรแกรมคอมพิวเตอร์ สั่งงานคอมพิวเตอร์ให้ทำงานตามที่ต้องการ คอมพิวเตอร์จะทำการประมวลผลตามคำสั่งจากโปรแกรมคอมพิวเตอร์ที่ได้รับ ผลลัพธ์ที่ได้อาจอยู่ในลักษณะของข้อมูลการย้อนกลับมายังนักเขียนโปรแกรม ควบคุมการทำงานของคอมพิวเตอร์และแสดงผลลัพธ์ไปยังอุปกรณ์ภายนอก

ภาษาโปรแกรมคอมพิวเตอร์

ภาษาโปรแกรมคอมพิวเตอร์ (Computer Programming Language) หมายถึง การสร้างกลุ่มของกฎเกณฑ์ สัญลักษณ์ ตัวอักษร ในรูปแบบลักษณะของชุดคำสั่งใช้สั่งงานกับคอมพิวเตอร์ (โอภาส เอี่ยมสิริวงศ์ และ สมโภชน์ ชื่นเอี่ยม, 2560) หรือภาษาที่ใช้ในการออกแบบเพื่อใช้ในการอ่านและเขียนของมนุษย์ สร้างเป็นโปรแกรมขึ้น เพื่อให้คอมพิวเตอร์สามารถทำงานตามที่ต้องการได้ (Evans, 2011)

ดังนั้น ภาษาโปรแกรมคอมพิวเตอร์ มีลักษณะการสร้างกฎเกณฑ์ โดยใช้สัญลักษณ์ที่มนุษย์คิดค้นขึ้นมา เพื่อใช้สื่อสารสั่งงานระหว่างมนุษย์กับเครื่องคอมพิวเตอร์ ในปัจจุบันนอกเหนือจากคอมพิวเตอร์ ยังประกอบไปด้วยอุปกรณ์เชื่อมต่อภายนอกเครื่องคอมพิวเตอร์ วิวัฒนาการของภาษาโปรแกรมคอมพิวเตอร์ตั้งแต่ยุคเริ่มแรกจนถึงปัจจุบัน มีรายละเอียดดังต่อไปนี้

1. ภาษาระดับต่ำ

ภาษาระดับต่ำ (Programming Low Level Language) มีลักษณะการใช้คำสั่งเป็นชุดของเลขฐานสองหรือใช้สัญลักษณ์เป็นภาษาอังกฤษร่วมกับเลขฐานอื่น ๆ เพื่อใช้ในการสั่งงานคอมพิวเตอร์ (Oram and Naik, 2010) ภาษาในยุคนี้มีภาพรวมของภาษามีความยากในการเขียนชุดคำสั่ง บุคคลที่สามารถเขียนคำสั่งควบคุมการทำงานของเครื่องคอมพิวเตอร์ ต้องมีผู้มีความรู้ความชำนาญในด้านของอุปกรณ์อิเล็กทรอนิกส์เป็นพื้นฐาน จึงมีการใช้งานเฉพาะกลุ่มเท่านั้น ภาษาระดับต่ำแบ่งออกเป็น 2 ประเภท ดังนี้

1.1 ภาษาเครื่อง

ภาษาเครื่อง (Machine Language) หมายถึง เครื่องคอมพิวเตอร์นั้นจะถูกขับเคลื่อนได้โดยโปรแกรมที่เรียกว่า “รหัสเครื่อง (Machine Code)” โปรแกรมรหัสเครื่องนี้เกิดจากการเรียงกันของคำสั่ง แต่ละคำสั่งประกอบไปด้วยบิต (เลข 0 กับ 1) ที่ซึ่งจะถูกแปลความหมายโดยเครื่องและปฏิบัติตามขอบเขตของคำสั่งนั้น รหัสเครื่องเป็นคำสั่งยุคแรกของการสั่งงานคอมพิวเตอร์ (Watt and Brown, 2000)

รหัสเครื่องสามารถใช้งานได้ง่ายกับเครื่องคอมพิวเตอร์ เนื่องจากเมื่อมีการการนำคำสั่งเข้าไปในเครื่อง เครื่องจะสามารถทำงานได้ทันที เนื่องจากเครื่องคอมพิวเตอร์นั้น สามารถเข้าใจรหัสเครื่องเมื่อมีการโปรแกรมเข้าไปในระบบ ดังนั้นสามารถกล่าวได้ว่า ภาษาคอมพิวเตอร์และภาษาเครื่องเป็นภาษาเดียวกัน

จากลักษณะดังกล่าว คอมพิวเตอร์จะติดต่อสื่อสารหรือพูดคุยในลักษณะของรหัสเครื่องที่เป็นเลขฐานสอง ทำให้ยากต่อการอ่าน การเขียนและการแก้ไข นักเขียนโปรแกรมคอมพิวเตอร์ต้องอ้างอิงตำแหน่งที่อยู่ในหน่วยความจำจริง เพื่อใช้ในการเก็บข้อมูลหรือคำสั่งและการสร้างคำสั่งจะต้องอยู่ในรูปของบิตเรียงกัน ทำให้นักเขียนโปรแกรมคอมพิวเตอร์ยากต่อการใช้งานคอมพิวเตอร์

1.2 ภาษาแอสเซมบลี

ภาษาแอสเซมบลี หรือ ภาษาสัญลักษณ์ (Assembly/Symbolic Language) หมายถึง ภาษาระดับต่ำ ที่เรียกว่า “นิวมอนิกโค้ด” (Mnemonic code) ถูกนำไปใช้แทนภาษาเครื่อง เพื่อให้ง่ายต่อการเขียนโปรแกรม (Liang, 2015) ภาษาแอสเซมบลีถูกพัฒนาขึ้นมาใหม่ต่อจากภาษาเครื่อง สืบเนื่องมาจากความยากในการเขียนชุดคำสั่งงานด้วยเลขฐานสอง จึงมีผู้คิดค้นพัฒนาเป็นรูปแบบของภาษาสัญลักษณ์แทนคำสั่งเลขฐานสองมาแทน เพื่อให้การสร้างโปรแกรมคอมพิวเตอร์ให้มีความง่ายต่อการใช้งาน

2. ภาษาระดับสูง

ภาษาระดับสูง (Programming High Level Language) หมายถึง ภาษาโปรแกรมคอมพิวเตอร์ที่ถูกพัฒนาขึ้น คำสั่งคล้ายกับภาษาอังกฤษ ง่ายต่อการเรียนรู้และการเขียนโปรแกรม (Liang, 2015) ภาษาระดับนี้ถูกพัฒนาต่อเนื่องมาจากภาษาระดับต่ำ ลักษณะภาษาระดับสูงมีการกำหนดสัญลักษณ์การใช้งานคำสั่งเป็นรูปแบบของอักษรภาษาอังกฤษ ที่ใกล้เคียงกับข้อความที่ใช้ในการสื่อสารกันของมนุษย์ จึงทำให้ผู้เขียนโปรแกรมสามารถเขียนคำสั่ง สั่งงานควบคุมการทำงานคอมพิวเตอร์ได้ง่ายขึ้น

ภาษาระดับสูง ได้มีผู้คิดค้นพัฒนาขึ้นมาใช้งานมากมายหลายภาษา เนื่องจากความแตกต่างกันของนำภาษาโปรแกรมไปพัฒนาให้คอมพิวเตอร์ทำงานตามวัตถุประสงค์ที่ต้องการ เช่น ภาษาฟอร์แทน (FORTRAN : FORmular TRANslator) นำไปใช้ในด้านการคำนวณทางคณิตศาสตร์และแก้ปัญหางานทางวิทยาศาสตร์ พัฒนาขึ้นมาโดยบริษัทไอบีเอ็ม ภาษาเบสิก (BASIC : Beginner’s All-purpose Symbolic Instruction Code) ใช้ในการเรียนการสอน ลักษณะของภาษาที่ถูกออกแบบให้ง่ายต่อการศึกษาเรียนรู้ ใช้งานได้อย่างกว้างขวาง ทั้งงานทางธุรกิจและงานอื่น ๆ พัฒนาขึ้นมาใช้งาน โดยสถาบันมาตรฐานแห่งชาติของสหรัฐอเมริกา (American National Standards Institute : ANSI) ภาษาโคบอล (COBOL : Common Business Oriented Language) ใช้ในงานด้านธุรกิจเป็นหลัก ภาษาปาสคาล (Pascal) ตั้งชื่อเป็นเกียรติกับ Blaise Pascal พัฒนาขึ้น

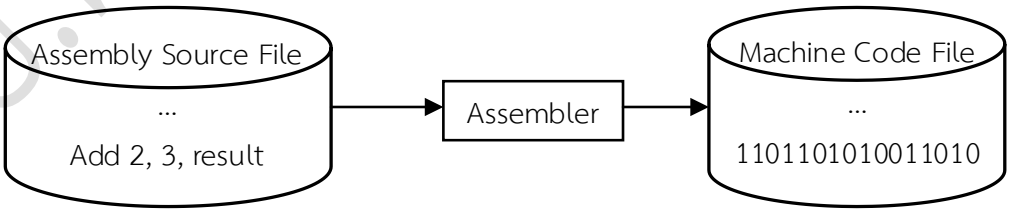
มาใช้ในงานด้านการคำนวณทั่วไป งานทางวิทยาศาสตร์ ธุรกิจและวิศวกรรม พัฒนาโดย เดนนิส ริชชี่ (Dennis Ritchie) ภาษาวิชวลเบสิค (Visual Basic) ใช้ลักษณะของการใช้โปรแกรมรูปภาพร่วมกับการเขียนโปรแกรม ทำให้ผู้พัฒนาโปรแกรม สามารถสร้างโปรแกรมประยุกต์ได้ง่ายและรวดเร็วขึ้น ภาษาซีพลัสพลัส (C++) ใช้ในการพัฒนาโปรแกรมระบบและภาษาจาวา (Java) พัฒนาขึ้นด้วยบริษัท Sun Microsystems ใช้งานอย่างกว้างขวางในปัจจุบัน ใช้ในการสร้างกรอบการใช้งานระบบเครือข่าย ซึ่งมีเครื่องมือสนับการทำงานจำนวนมาก เป็นต้น (Suchato, 2011)

ตัวแปลภาษาโปรแกรม

การเขียนโปรแกรมในปัจจุบัน ภาษาโปรแกรมคอมพิวเตอร์ที่ถูกสั่งงานมีลักษณะใกล้เคียงกับภาษามนุษย์ แต่เครื่องคอมพิวเตอร์ไม่สามารถเข้าใจภาษานี้ได้ จึงจำเป็นต้องมีตัวกลางที่จะทำหน้าที่ในการแปลภาษาโปรแกรมที่ไม่ใช่ภาษาเครื่อง ให้เป็นภาษาเครื่องที่คอมพิวเตอร์เข้าใจ เรียกว่า “ตัวแปลภาษาโปรแกรม (Translator Program)” ทำหน้าที่ในการแปลงโปรแกรมที่สร้างขึ้นให้เป็นเลขฐานสองหรือภาษาเครื่อง เพื่อให้คอมพิวเตอร์สามารถเข้าใจคำสั่งนั่นเอง ลักษณะของตัวแปลภาษามีรายละเอียดการทำงานดังต่อไปนี้

1. แอสเซมเบลอร์

แอสเซมเบลอร์ (Assembler) หมายถึง ตัวแปลภาษาที่ทำหน้าที่ในการแปลภาษาแอสเซมบลีให้เป็นภาษาเครื่อง (Jorgensen, 2016) โดยภาษาแอสเซมบลีจะให้ตัวแปลภาษาแอสเซมเบลอร์ ทำการแปลจากไฟล์ต้นฉบับแอสเซมบลี (Assembly Source File) ให้อยู่ในรูปของแฟ้มรหัสเครื่อง (Machine Code File) และนำไปสั่งคอมพิวเตอร์ให้ทำงานตามที่โปรแกรมได้สร้างขึ้นตามวัตถุประสงค์ที่ได้กำหนด ตัวแปลภาษาแอสเซมเบลอร์จะมีลักษณะการทำงานที่แตกต่างกัน ขึ้นอยู่กับสถาปัตยกรรมของคอมพิวเตอร์ที่นำภาษาแอสเซมบลีไปใช้งาน ดังภาพที่ 1.3



ภาพที่ 1.3 การทำงานของตัวแปลภาษาแอสเซมเบลอร์

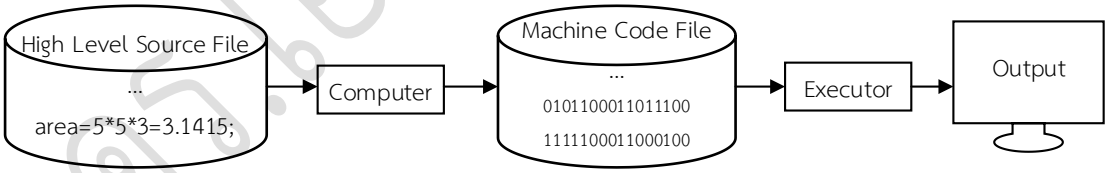
ที่มา : Liang (2015)

การสร้างโปรแกรมคอมพิวเตอร์ภาษาแอสเซมบลีค่อนข้างมีความยุ่งยากในการใช้งาน เนื่องจากการเขียนโปรแกรมเป็นรูปแบบของภาษาสัญลักษณ์ร่วมกับเลขฐาน ยังมีรูปแบบที่ไม่ใช่ภาษาที่ใกล้เคียงกับภาษาของมนุษย์ แต่จะง่ายกว่าเมื่อเปรียบเทียบกับภาษาเครื่อง ข้อดีของภาษาแอสเซมบลี คือ การทำงานของโปรแกรมเร็วและขนาดของตัวโปรแกรมมีขนาดเนื้อที่น้อยกว่าโปรแกรมที่สร้างจากภาษาระดับสูง จึงนิยมใช้ภาษานี้เมื่อต้องการประหยัดเวลาทำงานของเครื่องคอมพิวเตอร์และเพิ่มประสิทธิภาพของโปรแกรม

2. คอมไพเลอร์

คอมไพเลอร์ (Complier) หรือ ตัวแปลโปรแกรม (ราชบัณฑิตยสภา, 2564) หมายถึงโปรแกรมคอมพิวเตอร์ที่ถูกสร้างด้วยภาษาโปรแกรม ทำหน้าที่ในการแปลโปรแกรมที่ถูกสร้างขึ้นในภาษาระดับสูง ซึ่งใช้งานง่ายสำหรับมนุษย์ ให้เป็นภาษาเครื่องที่ซึ่งคอมพิวเตอร์ประมวลผลได้ (Evans, 2011)

ตัวแปลภาษาคอมไพเลอร์ทำหน้าที่แปล แฟ้มต้นฉบับภาษาระดับสูง (High Level Source File) ให้เป็นแฟ้มรหัสเครื่อง ซึ่งกระบวนการทำงานนั้น จะทำการแปลรหัสทีละบรรทัด หากพบข้อผิดพลาดของคำสั่ง ในการใช้รูปแบบคำสั่งผิดกฎเกณฑ์ที่กำหนดไว้ของแต่ละภาษา จะแสดงข้อความแจ้งข้อผิดพลาดที่เกิดขึ้นและหยุดการแปล ผู้เขียนโปรแกรมต้องแก้ไขข้อผิดพลาดนั้นและทำการแปลใหม่จนกระทั่งไม่มีข้อผิดพลาดของโปรแกรม จึงได้ไฟล์รหัสเครื่องและนำไฟล์ที่ได้ไปสั่งงานเครื่องคอมพิวเตอร์ผ่านตัวกระทำการ (Executor) ซึ่งจะทำหน้าที่ในการประมวลผล ให้ได้ผลลัพธ์ตามโปรแกรมที่สร้างขึ้น ดังภาพที่ 1.4



ภาพที่ 1.4 การทำงานของตัวแปลภาษาคอมไพเลอร์

ที่มา : Liang (2015)

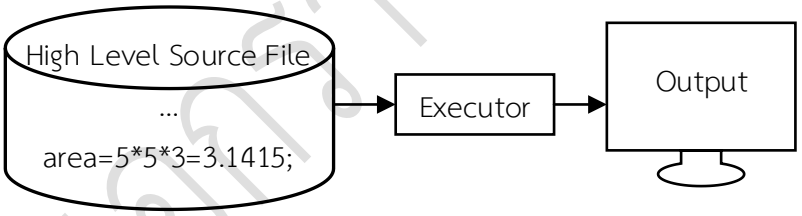
การทำงานของลักษณะตัวแปลภาษาแบบคอมไพเลอร์ จากภาพที่ 1.4 จะเห็นได้ว่า ข้อดีของตัวแปลภาษาคอมไพเลอร์ คือ แฟ้มรหัสต้นฉบับเมื่อผ่านการคอมไพล์ จะได้ไฟล์ที่เป็นรหัสเครื่อง เมื่อมีการใช้งานโปรแกรมนี้อีกครั้ง สามารถเรียกไฟล์ที่เป็นรหัสเครื่องไปสั่งงานคอมพิวเตอร์ได้ทันที โดยไม่ต้องผ่านกระบวนการคอมไพล์ใหม่อีกครั้ง ส่วนข้อเสียของคอมไพเลอร์ คือ ผลลัพธ์โปรแกรมที่

ได้มีความล่าช้า เนื่องจากจะต้องมีการตรวจสอบความถูกต้องของโปรแกรมทุกขั้นตอน จะต้องไม่เกิดความผิดพลาดเกิดขึ้น โปรแกรมจึงจะสามารถนำไปใช้งานได้ โปรแกรมที่มีขนาดใหญ่อาจจะต้องใช้เวลาในการประมวลผล เพื่อให้ได้ผลลัพธ์ที่ต้องการ

3. อินเทอร์พรีเตอร์

อินเทอร์พรีเตอร์ (Interpreter) หรือ ตัวแปลคำสั่ง (ราชบัณฑิตยสภา, 2564) หมายถึง โปรแกรมที่เป็นเครื่องมือในการแปลภาษาระดับสูงให้เป็นภาษาเครื่อง หลักการทำงานของตัวแปลภาษาอินเทอร์พรีเตอร์ จะทำการแปลภาษาทีละบรรทัด หากบรรทัดนั้นถูกต้องตามหลักของภาษาโปรแกรมหรือผ่านการประมวลผลชุดคำสั่ง จะนำไปแสดงผลลัพธ์ที่ได้ แต่ถ้าหากบรรทัดใดมีข้อผิดพลาด จะหยุดการทำงานบรรทัดนั้นพร้อมทั้งแสดงข้อผิดพลาดที่เกิดขึ้นและข้ามกระโดด ไปทำงานในบรรทัดถัดไป การทำงานลักษณะนี้จะเห็นผลลัพธ์ที่ชุดคำสั่งถูกต้องและส่วนที่ผิดพลาดพร้อมกัน (Evans, 2011)

ตัวแปลภาษาแบบอินเทอร์พรีเตอร์ โดยโปรแกรมภาษาคอมพิวเตอร์ต่าง ๆ ที่ใช้ตัวแปลภาษาแบบอินเทอร์พรีเตอร์ จะมีไลบลารีเก็บคำสั่งของโปรแกรมนั้นไว้ เมื่อมีการแปลคำสั่ง จะทำการตรวจสอบคำสั่งกับไลบลารี เพื่อตรวจสอบความถูกต้องของคำสั่ง ดังภาพที่ 1.5



ภาพที่ 1.5 การทำงานของตัวแปลภาษาอินเทอร์พรีเตอร์
ที่มา : Liang (2015)

การทำงานของลักษณะตัวแปลภาษาแบบอินเทอร์พรีเตอร์ จากภาพที่ 1.5 จะเห็นได้ว่า ข้อดีของตัวแปลภาษาอินเทอร์พรีเตอร์ คือ การแปลภาษาคอมพิวเตอร์นั้น เมื่อผ่านการแปลแต่ละบรรทัด สามารถแสดงผลการทำงานได้ทันทีในส่วนของชุดคำสั่งที่ถูกต้อง โดยไม่ต้องรอตรวจสอบชุดคำสั่งทั้งหมดในโปรแกรมและแสดงผลลัพธ์ ลักษณะการทำงานของตัวแปลอินเทอร์พรีเตอร์ในปัจจุบันคือกลุ่มภาษาโปรแกรมที่ทำงานบนเว็บ ส่วนข้อเสียของอินเตอร์พรีเตอร์ คือ แฟ้มรหัสต้นฉบับเมื่อผ่านการประมวลผลจะไม่มีแฟ้มรหัสเครื่อง ดังนั้น เมื่อมีการใช้งานโปรแกรมอีกครั้ง จะต้องดำเนินการประมวลผลไฟล์ต้นฉบับใหม่ทุกครั้ง โดยเริ่มอ่านคำสั่งจากจุดเริ่มของโปรแกรมไปจนสิ้นสุด หากโปรแกรมมีความซับซ้อนและปริมาณเป็นจำนวนมาก ทำให้การประมวลผลได้ล่าช้า

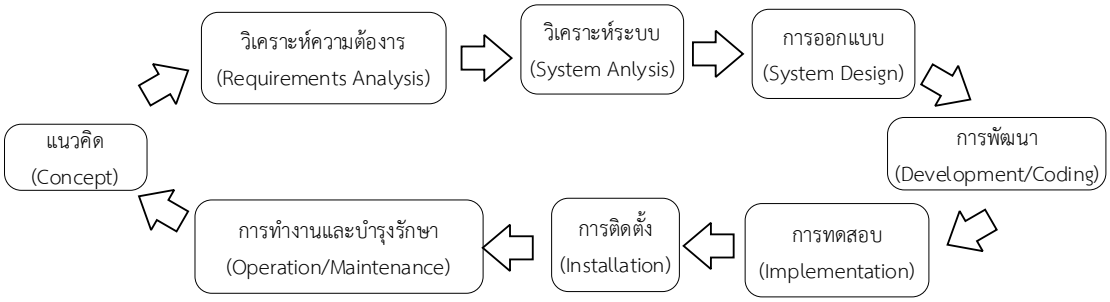
ขั้นตอนการพัฒนาโปรแกรม

การทำงานของระบบคอมพิวเตอร์เบื้องต้น จะประกอบไปด้วยฮาร์ดแวร์และซอฟต์แวร์ ความหมายของคำว่าซอฟต์แวร์และโปรแกรม จะมีลักษณะที่สั่งงานให้คอมพิวเตอร์ให้ทำงานเหมือนกัน แต่จะมีความแตกต่างกัน คือ โปรแกรมจะมีลักษณะของการทำงานที่มีขนาดเล็ก ซึ่งมีหน้าที่เฉพาะส่วนการทำงานอย่างใดอย่างหนึ่ง ซึ่งโดยทั่วไปโปรแกรมจะอยู่ภายใต้การทำงานของซอฟต์แวร์ ซึ่งซอฟต์แวร์จะกำหนดขอบเขตการทำงาน (Thakur, 2021) ลักษณะของโปรแกรมและซอฟต์แวร์มีรายละเอียดดังตัวอย่างต่อไปนี้

ภาพที่ 1.6 ลักษณะตัวอย่างของโปรแกรมและซอฟต์แวร์

ภาพที่ 1.6 ลักษณะความแตกต่างระหว่างโปรแกรมและซอฟต์แวร์ โดยตัวอย่างซอฟต์แวร์การขายสินค้าออนไลน์ ซึ่งซอฟต์แวร์ประกอบด้วยส่วนการทำงานที่แยกกัน ได้แก่ ลงทะเบียน เลือกสินค้า ชำระเงิน สถานะการส่ง จัดการลูกค้า จัดการสินค้า รายงานการสั่งซื้อและรายงานการส่งสินค้า ลักษณะดังกล่าวมีขอบเขตของความหมายแตกต่างกันระหว่างโปรแกรมและซอฟต์แวร์ ซึ่งนอกเหนือจากนี้การทำงานของโปรแกรมจะไม่ซับซ้อน เช่น การหาค่าดัชนีมวลกาย การคำนวณงวดรถยนต์ การคำนวณภาษี การคำนวณเงินซื้อสินค้าหลังหักส่วนลด การคำนวณเกรด เป็นต้น

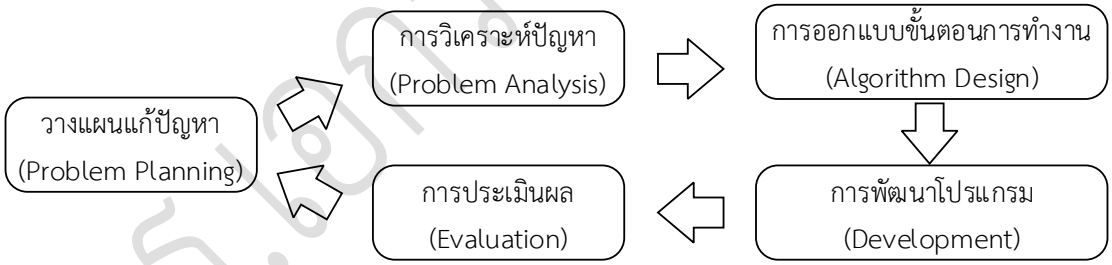
วงจรการพัฒนาซอฟต์แวร์ (Software Development Life Cycle) หมายถึง ช่วงของระยะเวลาของการพัฒนาซอฟต์แวร์ตั้งแต่เริ่มต้นจนสิ้นสุด ซอฟต์แวร์ที่ได้จะต้องผ่านการพัฒนาแต่ละระยะ สิ่งที่มีความสำคัญของกระบวนการตั้งแต่ การกำหนดคุณลักษณะและหน้าที่การทำงานของแต่ละส่วน การวิเคราะห์ การออกแบบ การประเมินผล การนำซอฟต์แวร์ไปใช้ในการทำงานพร้อมกับปรับปรุงให้ถูกต้องควบคู่กันไปและสุดท้ายเมื่อซอฟต์แวร์มีประโยชน์น้อยลงหรือล้าสมัย จะต้องผ่านการพิจารณาถึงสิ่งที่ปัญหาใหม่ผ่านกระบวนการเดิมอีกครั้ง (Leon, 2015) ซึ่งวงจากการพัฒนาซอฟต์แวร์แต่ระยะ มีรายละเอียดดังต่อไปนี้



ภาพที่ 1.7 วงจรการพัฒนาซอฟต์แวร์

ที่มา : Leon (2015)

การได้มาของซอฟต์แวร์ที่มีประสิทธิภาพ จะต้องมีกระบวนการหรือขั้นตอนในการซอฟต์แวร์ ซึ่งการพัฒนาโปรแกรมจะให้มีประสิทธิภาพ ก็จะต้องมีขั้นตอนของการพัฒนาโปรแกรมเช่นกัน กระบวนการของการพัฒนาโปรแกรมนี้เรียกว่า “วงจรการพัฒนาโปรแกรม หรือ PDLC (Program Development Life Cycle)” คือ กระบวนการนำไปสู่การออกแบบภาพรวมและการสร้างกระบวนการสำหรับพัฒนาโปรแกรม โปรแกรมที่มีประสิทธิภาพจะต้องผ่านการวางแผนก่อนถึงจะนำไปสู่การเขียนเป็นโปรแกรม (Prettyman, 2016) มีรายละเอียดดังต่อไปนี้



ภาพที่ 1.8 วงจรการพัฒนาโปรแกรม

ที่มา : Prettyman (2016)

วงจรการพัฒนาโปรแกรมจะประกอบด้วย 5 ขั้นตอน เริ่มต้นจากกระบวนการวางแผน การวิเคราะห์ การออกแบบ การพัฒนา การประเมินผล และเมื่อมีข้อบกพร่องหรือปัญหาก็จะย้อนกลับไปสู่กระบวนการของการวางแผนแก้ปัญหาอีกครั้ง ซึ่งภาพรวมวงจรการพัฒนาโปรแกรมจะเห็นได้ว่ามีลักษณะใกล้เคียงกันกับวงจรการพัฒนาซอฟต์แวร์ ถือได้ว่าวงจรการพัฒนาโปรแกรมนั้นเป็นส่วนหนึ่งของวงจรการพัฒนาซอฟต์แวร์แต่มีความซับซ้อนน้อยกว่า การทำงานจะเป็นลักษณะของวงจรของการพัฒนาโปรแกรมแต่ละขั้นตอนมีรายละเอียดดังต่อไปนี้

1. การวางแผนแก้ไขปัญหา

การวางแผน (Problem Planning) หมายถึง การกำหนดนิยามและขอบเขตของปัญหาที่จะนำมาพัฒนาโปรแกรม ในขั้นตอนนี้ต้องมีความเข้าใจในปัญหา สิ่งที่ต้องการและผลลัพธ์ที่แก้ปัญหาได้ การวางแผนแก้ปัญหามีรายละเอียดดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1.1 โจทย์ปัญหา : ผู้สอนมีปัญหาของการหาผลการเรียนของผู้เรียน ซึ่งจะคำนวณคะแนนรวม ซึ่งได้มาจากคะแนนเก็บ คะแนนกลางภาคและคะแนนปลายภาค นำคะแนนที่ได้มาเปรียบเทียบหาผลการเรียนที่ได้ของนักศึกษา โดยวิธีการเดิมผู้สอนจะต้องคำนวณและหาผลการเรียนด้วยตัวผู้สอนเอง ทำให้เกิดความล้าช้าในการทำงานและมีความผิดพลาดจากการคำนวณ หาผลการเรียนจากตัวผู้สอนเอง

การวางแผน : พัฒนาโปรแกรมให้หาผลการเรียนที่ได้ โดยกรอกข้อมูลเฉพาะคะแนนเก็บ คะแนนกลางภาค คะแนนปลายภาค และให้โปรแกรมทำการควณนวณ แสดงผลลัพธ์ของผล การเรียนที่ได้ โดยมีเงื่อนไขดังนี้คือ

ถ้าคะแนนรวมมากว่าหรือเท่ากับ 80 จะได้ผลการเรียน A
ถ้าคะแนนรวมมากกว่าหรือเท่ากับ 70 แต่ไม่ถึง 80 ได้ผลการเรียน B
ถ้าคะแนนรวมมากกว่าหรือเท่ากับ 60 แต่ไม่ถึง 70 ได้ผลการเรียน C
ถ้าคะแนนรวมมากกว่าหรือเท่ากับ 50 แต่ไม่ถึง 60 ได้ผลการเรียน D
ถ้าคะแนนรวมน้อยกว่า 50 ได้เกรด E

จากตัวอย่างข้างต้น เริ่มต้นจากโจทย์ปัญหาที่เกิดขึ้นและมีความต้องการในการที่จะแก้ไขปัญหา โดยการวางแผนจะบอกรายละเอียดขั้นตอนการทำงาน ข้อมูล เงื่อนไขหรือกระบวนการที่ จะได้มาซึ่งผลลัพธ์แก้ไขโจทย์ปัญหาให้ชัดเจน ก่อนนำไปสู่กระบวนการต่อไป ถ้ามีความถูกต้องชัดเจนในขั้นต้น ผลลัพธ์ที่ได้ของโปรแกรมก็จะมีประสิทธิภาพตรงกับความต้องการในการแก้โจทย์ปัญหา

2. การวิเคราะห์ปัญหา

การวิเคราะห์ปัญหา (Problem Analysis) หมายถึง ขั้นตอนของการกำหนดลักษณะของข้อมูลและกระบวนการที่อยู่ในปัญหา โดยการวิเคราะห์ทั่วไปจะเริ่มจากการวิเคราะห์ข้อมูลส่วนนำออก (Output) มายังส่วนประมวลผล (Process) และข้อมูลส่วนนำเข้า (Input) ซึ่งจากตัวอย่างที่

2.1 นำมาวิเคราะห์ปัญหา ซึ่งรายละเอียดดังต่อไปนี้

1. ข้อมูลนำออก

   ผลการเรียนที่ได้

2. ประมวลผล

   2.1 ผลการเรียนที่ได้

   ถ้าคะแนนรวมมากว่าหรือเท่ากับ 80 จะได้ผลการเรียน A

ถ้าคะแนนรวมมากกว่าหรือเท่ากับ 70 แต่ไม่ถึง 80 ได้ผลการเรียน B
ถ้าคะแนนรวมมากกว่าหรือเท่ากับ 60 แต่ไม่ถึง 70 ได้ผลการเรียน C
ถ้าคะแนนรวมมากกว่าหรือเท่ากับ 50 แต่ไม่ถึง 60 ได้ผลการเรียน D
ถ้าคะแนนรวมน้อยกว่า 50 ได้ผลการเรียน E

2.2 คะแนนรวม = คะแนนเก็บ + คะแนนกลางภาค + คะแนนปลายภาค

3. ข้อมูลนำเข้า

คะแนนเก็บ คะแนนกลางภาค และคะแนนปลายภาค

การวิเคราะห์ตัวอย่างของปัญหา จะเริ่มต้นจากการวิเคราะห์ข้อมูลนำออกหรือสิ่งที่เป็นผลลัพธ์ที่ต้องการนั่นก็คือผลการเรียน ดำเนินการวิเคราะห์ที่มาของผลการเรียนที่ได้ ซึ่งได้จากการประมวลผลขั้นที่ 1 คือ การเปรียบเทียบผลการเรียนที่ได้ โดยนำคะแนนรวมไปทำการเปรียบเทียบ ซึ่งมีเงื่อนไขของการประมวลผลขั้นที่ 2 คือ คำนวณคะแนนรวม โดยได้มาจาก คะแนนเก็บ คะแนนกลางภาคและคะแนนปลายภาค ซึ่งคะแนนทั้ง 3 ส่วน ไม่มีเงื่อนไขของการประมวลต่อ จึงถือว่าคะแนนทั้ง 3 ส่วนอยู่ในส่วนของข้อมูลนำเข้า

การวิเคราะห์ปัญหาพบว่าจะต้องวิเคราะห์ทีละขั้นตอน โดยเริ่มจากข้อมูลนำออก วิเคราะห์ที่มาของข้อมูลได้มาจากการประมวลอะไรและวิเคราะห์การประมวลต่อ จะต้องประกอบไปด้วยข้อมูลอะไรและข้อมูลที่ใช้ในการประมวลผลได้มาจากที่ใด ซึ่งข้อมูลที่นำมาประมวลผลอาจเกิดจากการประมวลอีกขั้นตอนก็ได้ แต่ถ้าไม่ได้เกิดจากการประมวลผลใด ข้อมูลนั้นจะถือว่าเป็นข้อมูลนำเข้า

3. การออกแบบขั้นตอนวิธี

ขั้นตอนวิธี หรือ อัลกอริทึม (Algorithm) (ราชบัณฑิตยสภา, 2564) หมายถึง ขั้นตอนการทำงาน เพื่อนำเสนอในการแก้ปัญหา ซึ่งเป็นการอธิบายลักษณะการทำงานโดยการใช้ภาษามนุษย์ (Karumanchi, 2014) หรือกลุ่มของขั้นตอน กฎเกณฑ์ที่นำพาไปสู่การไขปัญหา (โอภาส เอี่ยมสิริวงศ์, 2559)

ดังนั้น การออกแบบขั้นตอนวิธี (Algorithm Design) เป็นขั้นตอนการอธิบายการวางแผนแก้ปัญหาโจทย์ปัญหาทางคอมพิวเตอร์อย่างเป็นขั้นเป็นตอน ซึ่งการอธิบายขั้นตอนมีหลายวิธีที่ใช้วิธีการเขียนอธิบายลำดับขั้นตอนการทำงานตั้งแต่เริ่มต้นจนสิ้นสุด เช่น การเขียนอัลกอริทึม การเขียนซูโดโค้ด (Pseudo code) หรือรหัสเทียมและการเขียนผังโปรแกรม ซึ่งเป็นการอธิบายการทำงานในลักษณะของแผนภาพการทำงาน เป็นต้น ทั้งนี้แต่ละวิธีต่างมีจุดประสงค์ ในการนำเสนอไม่เหมือนกัน แต่ทุกวิธีต่างมีจุดประสงค์ในการแสดงลำดับขั้นตอนกระบวนการแก้ปัญหางาน เพื่อให้ได้ผลลัพธ์ตามความต้องการ ก่อนนำไปสู่ขั้นตอนของการเขียนโปรแกรม

การออกแบบขั้นตอนวิธีนั้น จะนำข้อมูลที่ได้จากการวิเคราะห์มาดำเนินการสร้างขั้นตอนวิธี ซึ่งจะมองภาพตรงข้ามกันกับการวิเคราะห์ โดยจะมองจากข้อมูลนำเข้า การประมวลผลและข้อมูล

นำออก นำมาเขียนเป็นขั้นตอนวิธี การกำหนดข้อมูลและนำเสนอในลักษณะของแผนภาพ ดังตัวอย่างการเขียนอัลกอริทึมจากตัวอย่างที่ 1.1 ดังนี้

1. อัลกอริทึม
    1.1 นำข้อมูลเข้า คะแนนเก็บ คะแนนกลางภาค และคะแนนปลายภาค
    1.2 คำนวณคะแนนรวม
        คะแนนรวม = คะแนนเก็บ + คะแนนกลางภาค + คะแนนปลายภาค
    1.3 เปรียบเทียบผลการเรียนที่ได้
        ถ้าคะแนนรวมมากว่าหรือเท่ากับ 80 จะได้ผลการเรียน A
        ถ้าคะแนนรวมมากกว่าหรือเท่ากับ 70 แต่ไม่ถึง 80 ได้ผลการเรียน B
        ถ้าคะแนนรวมมากกว่าหรือเท่ากับ 60 แต่ไม่ถึง 70 ได้ผลการเรียน C
        ถ้าคะแนนรวมมากกว่าหรือเท่ากับ 50 แต่ไม่ถึง 60 ได้ผลการเรียน D
        ถ้าคะแนนรวมน้อยกว่า 50 ได้ผลการเรียน E
    1.4 แสดงผลการเรียนที่ได้
    1.5 จบการทำงาน
2. กำหนดข้อมูล
    2.1 ข้อมูลนำเข้า
        คะแนนเก็บ เก็บข้อมูลตัวเลขจำนวนเต็ม
        คะแนนกลางภาค เก็บข้อมูลตัวเลขจำนวนเต็ม
        คะแนนปลายภาค เก็บข้อมูลตัวเลขจำนวนเต็ม
    2.2 ข้อมูลนำออก
        คะแนนรวม เก็บข้อมูลตัวเลขจำนวนเต็ม
        ผลการเรียนที่ได้ เก็บข้อมูลตัวอักษร

จากอัลกอริทึมดังกล่าว มีรายละเอียดขั้นตอนวิธีการทำงาน เพื่อแก้ปัญหาที่ได้กำหนดไว้ ซึ่งสามารถนำไปใช้ในการเขียนโปรแกรมได้ เนื่องจากได้กล่าวถึงขั้นตอนและข้อมูลที่จะใช้ทำงานของโปรแกรม ข้อดีของอัลกอริทึมก็คือ จะมีรายละเอียดที่สามารถเข้าใจได้ชัดเจน แต่ข้อเสียจะมองโครงสร้างการทำงานภาพรวมยาก เนื่องจากจะต้องอ่านทำความเข้าใจทีละขั้นตอน ดังนั้นเพื่อให้สามารถเข้าใจได้รวดเร็ว จึงนิยมนำไปเขียนเป็นแผนภาพการทำงาน

4. การพัฒนาโปรแกรม

การพัฒนาโปรแกรม (Development) หมายถึง การเขียนโปรแกรม (Program Coding) สั่งงานคอมพิวเตอร์ โดยต้องใช้คำสั่งของภาษาคอมพิวเตอร์ ซึ่งปัจจุบันมีให้เลือกในการพัฒนา

มากมาย แต่จะเลือกภาษาโปรแกรมคอมพิวเตอร์ใดมาเขียนโปรแกรม ต้องพิจารณารายละเอียดดังต่อไปนี้

4.1 ประสิทธิภาพของภาษาโปรแกรม

ผลที่ได้จากการออกแบบขั้นตอนวิธี สามารถนำไปเขียนด้วยภาษาใดก็ได้ ซึ่งการเลือกใช้ภาษาคอมพิวเตอร์ที่เหมาะสมกับระบบงานมาพัฒนาโปรแกรม ต้องพิจารณาความเหมาะสม ซึ่งภาษาโปรแกรมคอมพิวเตอร์มีมากมาย ซึ่งแต่ละภาษาคอมพิวเตอร์มีจุดมุ่งหมายในการพัฒนาขึ้นมาใช้งานแตกต่างกัน ดังนั้นจึงต้องพิจารณาประสิทธิภาพและสภาพแวดล้อมในการทำงานของแต่ละภาษาโปรแกรม เพื่อที่จะได้โปรแกรมที่สร้างขึ้นมีประสิทธิภาพสูงสุด แต่การพิจารณาอาจต้องตรวจสอบค่าใช้จ่ายเรื่องลิขสิทธิ์ที่จะซื้อภาษาโปรแกรมคอมพิวเตอร์มาพัฒนา เพื่อความเหมาะสมในการนำมาใช้งาน

4.2 ทรัพยากร

การเขียนโปรแกรมคอมพิวเตอร์ เพื่อความสะดวกในการจัดการและลดค่าใช้จ่ายในองค์กร ต้องพิจารณาบุคลากรภายในที่มีความรู้ความสามารถในการเขียนโปรแกรมคอมพิวเตอร์นำมาพัฒนาโปรแกรม ซึ่งต้องคำนึงถึงความรู้ความสามารถของผู้เขียนโปรแกรมภายในองค์กรนั้นสามารถใช้ภาษาโปรแกรมคอมพิวเตอร์ภาษาใดได้

5. การประเมินผลโปรแกรม

การประเมินผลโปรแกรม (Evaluation) คือ การทดสอบและตรวจสอบข้อผิดพลาด การทำงานของโปรแกรม (Program Testing & Debugging) ประกอบไปด้วย การทำงานของโปรแกรมไม่เกิดการหยุดระหว่างการทำงาน ผลลัพธ์และการประมวลผลที่ได้มีความถูกต้อง สะดวกและง่ายต่อการใช้งาน ซึ่งการทดสอบโปรแกรมจะประกอบไปด้วยการประเมินผล 2 กลุ่ม ดังต่อไปนี้

5.1 การทดสอบโดยผู้พัฒนาระบบงาน

การทดสอบโดยผู้พัฒนาระบบ คือ การทดสอบก่อนนำไปใช้งานจริง โดยใช้ข้อมูลสมมติบันทึกเข้าไปในโปรแกรมและตรวจสอบผลลัพธ์จากการประมวลผล พิจารณาความถูกต้องผลลัพธ์เป็นไปตามที่ได้วิเคราะห์หรือไม่ หากยังมีข้อผิดพลาด ต้องดำเนินการแก้ไขปรับปรุงโปรแกรมให้ถูกต้อง ซึ่งการทดสอบแบบนี้ จะมุ่งเน้นในการทดสอบการประมวลผลที่ถูกต้องเป็นหลัก เช่น การคำนวณ กระบวนการทำงาน เป็นต้น จากนั้นเมื่อทดสอบการทำงานถูกต้องทั้งหมด จึงนำสู่การทดสอบกระบวนการถัดไป

5.2 การทดสอบโดยผู้ใช้ระบบจริง

การทดสอบโดยผู้ใช้งานจริง หมายถึง การทดสอบในขั้นนี้ ลักษณะและปริมาณข้อมูลที่ใช้ในการทดสอบใกล้เคียงกับความเป็นจริง ซึ่งการทดสอบในขั้นตอนนี้ จะพิจารณากระบวนการของการใช้โปรแกรมเป็นหลัก เช่น ความซับซ้อน ความยาก รูปแบบรายงาน เป็นต้น ซึ่งอาจมีบางส่วนที่มี

ข้อผิดพลาดหลุดลอดจาการทดสอบโดยผู้พัฒนาระบบ จึงต้องปรับปรุง แก้ไขให้โปรแกรมมีการใช้งานที่ดีขึ้นและตรงตามวัตถุประสงค์ที่ตั้งเอาไว้

รหัสเทียม

รหัสเทียม หรือ ซูโดโค้ด (Pseudo code) (ราชบัณฑิตยสภา, 2564) คือ การจำลองความคิดเป็นข้อความหรือคำบรรยาย อธิบายขั้นตอนการทำงานของโปรแกรมโดยใช้ถ้อยคำผสมระหว่างภาษาอังกฤษและภาษาการเขียนโปรแกรมแบบโครงสร้าง หรืออาจใช้ภาษาไทยก็ได้แต่ควรเขียนเป็นภาษาอังกฤษ โดยให้ผู้เขียนโปรแกรมสามารถพัฒนาขั้นตอนต่าง ๆ ให้กับโปรแกรมได้ง่ายขึ้น แต่ส่วนใหญ่แล้วคำที่ใช้มักเป็นคำเฉพาะ (Reseved Word) ที่มีอยู่ในภาษาการเขียนโปรแกรมและมักจะเขียนด้วยตัวอักษรตัวใหญ่ ซูโดโค้ดที่ดีจะต้องมีความชัดเจน สั้น และได้ใจความ ข้อมูลต่างๆ ที่ใช้จะถูกเขียนอยู่ในรูปแบบของตัวแปร รหัสเทียมนี้บางครั้งจะเรียกว่า อัลกอริทึม รูปแบบทั่วไปรหัสเทียมจะเป็นการเขียนอัลกอริทึมโดยใช้ประโยคภาษาอังกฤษที่สื่อความหมายง่าย ๆ สามารถอ่านแล้วเข้าใจได้โดยทันที

1. กฎเกณฑ์การเขียนรหัสเทียม

การเขียนรหัสเทียมอธิบายการทำงาน มีหลักการและเงื่อนไขในการเขียนรหัสเทียมดังต่อไปนี้

1.1 ถ้อยคำหรือประโยคคำสั่ง (Statement) ให้เขียนอยู่ในรูปแบบของภาษาอังกฤษอย่างง่าย

1.2 ในหนึ่งบรรทัด ให้เขียนประโยคคำสั่งเพียงคำสั่งเดียว

1.3 ควรใช้ย่อหน้าให้เป็นประโยชน์ เพื่อแยกคำเฉพาะ (Keywords) ได้ชัดเจน รวมถึงจัดโครงสร้างการควบคุมให้เป็นสัดส่วน ซึ่งการกระทำดังกล่าวจะทำให้อ่านง่าย

1.4 แต่ละประโยคคำสั่งให้เขียนลำดับจากบนลงล่าง โดยมีทางเข้าเพียงทางเดียวและมีทางออกทางเดียวเท่านั้น

1.5 กลุ่มของประโยคคำสั่งต่างๆ อาจจัดรวมกลุ่มเข้าด้วยกันในรูปแบบของโมดูล แต่ต้องกำหนดชื่อโมดูลเหล่านั้นด้วย เพื่อให้สามารถเรียกใช้งานโมดูลนั้นได้

รูปแบบ

```
Algorithm <ชื่อของอัลกอริทึม>
1.....
2.....
.....
END
```

2. วิธีการเขียนอธิบายรหัสเทียม

การเขียนอธิบายรหัสเทียม มีรูปแบบการใช้คำในการเขียนอธิบายรหัสเทียม รายละเอียดดังต่อไปนี้

2.1 การรับข้อมูลเข้า และการแสดงผลข้อมูล

2.1.1 การรับข้อมูลเข้า มักจะนิยมใช้คำว่า READ หรือ INPUT ตามด้วยชื่อตัวแปรที่ต้องการเก็บข้อมูล ถ้าหากมีหลายตัวใช้ (" ; ") กั้น

2.1.2 การแสดงผล ใช้คำว่า OUTPUT หรือ PRINT

2.2 การคำนวณ ใช้คำว่า Compute ตามด้วยชื่อตัวแปรที่ต้องการเก็บค่าจากการคำนวณ ตามด้วย เครื่องหมายเท่ากับ และนิพจน์ของการคำนวณ เช่น Compute area = a*b;

2.3 การตัดสินใจ และทดสอบทางเลือก นิยมใช้คำว่า IF หรือ IF-THEN-ELSE และ ENDIF

```
IF a>0 THEN
PRINT POSITION
ELSE PRINT NEGATIVE
ENDIF
```

สำหรับในกรณีที่มีมากกว่า 2 ทาง จะใช้คำว่า CASE และ ENDCASE เช่น

```
CASE num OF
1 : PRINT ONE
2 : PRINT TWO
3 : PRINT THREE
ENDCASE
```

2.4 การกระทำแบบวนซ้ำ (Loop) การทำซ้ำจนกว่าเงื่อนไขที่กำหนดจะเป็นจริงจึงหยุดทำ (REPEAT UNTIL)

```
REPEAT
Statement 1
Statement 2
----------
UNTIL (CONDITION)
```

มีการตรวจสอบเงื่อนไขก่อนเข้าทำ ถ้าเงื่อนไขเป็นจริง จึงจะเข้าทำคำสั่งภายใน (DO...WHILE)

```
DO (CONDITION)
WHILE
Statement 1
```

Statement 2

ENDDO

3. ตัวอย่างการเขียนรหัสเทียม

จากกฎเกณฑ์และการเขียนรหัสเทียม สามารถนำใช้ในการอธิบายขั้นตอนการทำงาน ดังตัวอย่างต่อไปนี้

3.1 การเขียนซูโดโค้ดสำหรับให้คอมพิวเตอร์รับค่า 2 ค่าจากผู้ใช้ นำมาบวกกัน อาจเขียนได้ดังนี้

Algorithm การบวกเลข 2 จำนวน	Algorithm Add 2 Number
1. เริ่มต้น	1. START
2. รับค่าจำนวนที่ 1	2. READ X, INPUT X
3. รับค่าจำนวนที่ 2	3. READ Y, INPUT Y
4. นำตัวเลขมาบวกกัน	4. COMPUTE SUM = X+Y
ผลรวม =จำนวนที่ 1+ จำนวนที่ 2	5. PRINT SUM, OUTPUT SUM
5. แสดงผลลัพธ์ทางจอภาพ	6. STOP
6. จบ	

3.2 การเขียนซูโดโค้ดสำหรับคำนวณหาพื้นที่สามเหลี่ยมโดยรับค่าความกว้างของฐานและความสูงจากผู้ใช้ และนำมาคำนวณหาพื้นที่และแสดงผลออกทางจอภาพ อาจเขียนได้ดังนี้

Algorithm การหาพื้นที่สามเหลี่ยม	Algorithm The Triangle area
1. เริ่มต้น	1. START
2. รับค่าความกว้างฐาน	2. INPUT W
3. รับค่าความสูง	3. INPUT H
4. นำค่าที่รับมาคำนวณหาพื้นที่	4. COMPUTE TRIANGLE = $\frac{1}{2}$xWxH
พื้นที่สามเหลี่ยม= $\frac{1}{2}$ x ฐาน x สูง	5. PRINT TRIANGLE
5. แสดงผลลัพธ์ทางจอภาพ	6. STOP
6. จบ	

3.3 การเขียนซูโดโค้ดสำหรับให้คอมพิวเตอร์หาค่าเฉลี่ยจากข้อมูลที่รับเข้าทางแป้นพิมพ์อาจเขียนได้ ดังนี้

Algorithm การหาค่าเฉลี่ย	Algorithm Avarage_Sum
1. ตัวนับ = 0	1. COUNT = 0
2. ผลรวม = 0	2. SUM =0
3. รับค่าทางแป้นพิมพ์เก็บไว้ใน (ข้อมูล)	3. INPUT (VALUE)
4. ถ้าข้อมูลมากกว่า 0 เพิ่มค่าตัวนับขึ้นหนึ่งค่า ผลรวม = ผลรวม + ค่าข้อมูล ย้อนกลับไปทำขั้นตอนที่ 3 ถ้าไม่มากกว่าไปทำขั้นตอนที่ 5	4. IF VALUE > 0 THEN 1. COUNT = COUNT+1 2. SUM = SUM+ VALUE 3. GOTO 3 4. ELSE GOTO 5
5. ค่าเฉลี่ย = ผลรวมหารด้วยตัวนับ	5. AVARAGE = SUM/COUNT
6. แสดงค่าเฉลี่ยทางจอภาพ โดยมีทศนิยมสองตำแหน่ง	6. OUTPUT (AVARAGE)
7. จบ	7. END

จะเห็นว่าขั้นตอนการหาค่าเฉลี่ยได้เขียนไว้อย่างเข้าใจ สามารถทราบได้ว่าในการทำงานต่างๆ จะต้องใช้ตัวแปรใดบ้าง แต่ละขั้นตอนมีการประมวลผลอย่างไร แต่โดยทั่วไปแล้วซูโดโค้ดจะถูกเขียนด้วยภาษาอังกฤษ ดังต่อไปนี้

3.4 การเขียนซูโดโค้ดเพื่อคำนวณผลการเรียน การคิดผลการสอบของนักเรียนจากคะแนน โดยกำหนดให้ ถ้าคะแนนมากกว่าหรือเท่ากับ 80 ได้เกรด A ถ้าคะแนนมากกว่าหรือเท่ากับ 70 ได้เกรด B ถ้าคะแนนมากกว่าหรือเท่ากับ 60 ได้เกรด C ถ้าคะแนนมากกว่าหรือเท่ากับ 50 ได้เกรด D และ ถ้าคะแนนน้อยกว่า 50 ได้เกรด F สามารถเขียนรหัสเทียมได้ดังนี้

Algorithm Grade

1. SCORE = 0
2. INPUT (SCORE)
3. IF SCORE >= 80

   GRADE = “A”

   ELSE IF SCORE >= 70

   GRADE = “B”

   ELSE IF SCORE >= 60

   GRADE = “C”

ELSE IF SCORE >= 50

GRADE = “D”

ELSE

GRADE = “F”

END IF

4. OUTPUT (GRADE)
5. END

3.5 การเขียนซูโดโค้ดเพื่อคำนวณผลการเรียน เพื่อรับค่าตัวเลข และทำการบวกค่าที่รับเข้ามาแบบวนซ้ำโดยให้ออกจากโปรแกรม เมื่อผลลัพธ์มีค่ามากกว่า 1,000 และแสดงผลของผลลัพธ์ที่ได้ สามารถเขียนรหัสเทียมได้ดังนี้

Algorithm Loop Add Number

1. TOTAL = 0
2. WHILE TOTAL <= 1000
3. INPUT NUMBER
4. TOTAL = TOTAL + NUMBER
5. ENDWHILE
6. PRINT TOTAL
7. END

แผนภาพแสดงลำดับขั้นตอนการทำงาน

แผนภาพแสดงลำดับขั้นตอนการทำงาน (Flowchart) หมายถึง เทคนิควิธีการแสดงขั้นตอนการทำงานของโปรแกรม โดยใช้สัญลักษณ์ภาพในการสื่อความหมาย อธิบายขั้นตอนการทำงานของระบบหรือโปรแกรม แก้ไขปัญหาให้ผู้ศึกษาสามารถทำความเข้าใจได้ง่ายขึ้น สื่อความหมายในทิศทางเดียวกัน (Farrell, 2011) โดยสัญลักษณ์ภาพที่ใช้นั้น ถูกกำหนดขึ้นตามมาตรฐานของหน่วยงานที่ได้การรับรองการใช้งานระดับนานาชาติ ANSI (American National Standards Institute) และ ISO (International Standard Organization) โดยมีรายละเอียดการนำไปใช้ดังต่อไปนี้

1. ประเภทของแผนภาพแสดงลำดับขั้นตอนการทำงาน

แผนภาพแสดงลำดับขั้นตอนการทำงานแบ่งตามการนำไปใช้ ซึ่งมีรูปแบบการนำเสนอ เพื่ออธิบายการทำงานแบ่งออกเป็น 2 ประเภทดังนี้

1.1 ผังระบบ

ผังระบบ (System Flowchart) หมายถึง แผนภาพที่ใช้ในการแสดงขั้นตอนการทำงานของระบบทั้งระบบ แสดงถึงความสัมพันธ์ของส่วนที่เกี่ยวข้องกับระบบงานทั้งหมด เช่น บุคคลภายในระบบ สิ่งของหรือเครื่องมือที่เกี่ยวข้อง หน่วยงานที่เกี่ยวข้อง เป็นต้น ซึ่งลักษณะของผังระบบจะมองภาพการทำงานขนาดใหญ่และมองการทำงานแต่ละส่วนมาประกอบกัน เพื่ออธิบายโครงสร้างการทำงาน

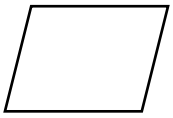
1.2 ผังโปรแกรม

ผังโปรแกรม (Program Flowchart) หมายถึง แผนภาพแสดงขั้นตอนการทำงานของของโปรแกรม ซึ่งแต่ละส่วนของโปรแกรมมาทำงานร่วมกันเป็นลักษณะการทำงานของระบบ ดังนั้นผังโปรแกรมแยกออกมากจากผังระบบ ซึ่งหน่วยการทำงานที่เกี่ยวข้องกับโปรแกรม อธิบายถึง การป้อนค่าข้อมูล การประมวลผลข้อมูล และการแสดงข้อมูล เป็นต้น

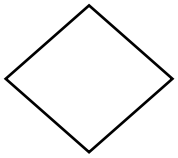
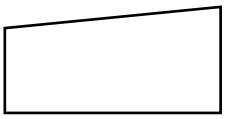
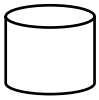
2. สัญลักษณ์แผนภาพแสดงลำดับขั้นตอนการทำงาน

สัญลักษณ์แผนภาพแสดงลำดับขั้นตอนการทำงาน ได้ถูกกำหนดสัญลักษณ์มาตรฐานของหน่วยงาน ANSI และ ISO เพื่อใช้ในการนำเสนอลักษณะขั้นตอนการทำงาน การนำเสนอด้วยแผนภาพแสดงขั้นตอนการทำงาน จะให้รายละเอียดและความเข้าใจในทิศทางเดียวกัน ในรูปแบบของสัญลักษณ์มาตรฐาน (Wikipedia, 2021) โดยมีสัญลักษณ์ที่ใช้ มีรายละเอียดดังต่อไปนี้

ตารางที่ 1.1 สัญลักษณ์แผนภาพแสดงลำดับขั้นตอนการทำงาน

ลำดับที่	สัญลักษณ์	ชื่อ (Symbol)	ความหมาย
1		Terminal	สัญลักษณ์ที่ใช้ในการอธิบายถึงจุดเริ่มต้นของการทำงานหรือจุดสิ้นสุดของการทำงาน
2		Processing	สัญลักษณ์ที่ใช้ในการประมวลผล เช่น การประมวลผล การเปลี่ยนค่า การบันทึกข้อมูล กำหนดรูปแบบหรือตำแหน่งที่อยู่ของข้อมูล
3		Input/Output	สัญลักษณ์ที่เกี่ยวข้องกับการนำข้อมูล 2 ลักษณะ ได้แก่ การแสดงการรับข้อมูลเข้าและการแสดงผลลัพธ์ของข้อมูล

ตารางที่ 1.1 สัญลักษณ์แผนภาพแสดงลำดับขั้นตอนการทำงาน (ต่อ)

ลำดับที่	สัญลักษณ์	ชื่อ (Symbol)	ความหมาย
4		Decision	สัญลักษณ์ที่ใช้ในการเปรียบเทียบข้อมูลที่รับเข้าและตัดสินใจ ในการเลือกเส้นทางของการทำงาน ซึ่งทางเลือกในการทำงานจะประกอบไปด้วย 2 เส้นทาง เช่น จริงหรือเท็จ ใช่หรือไม่ใช่ เป็นต้น
5		Manual	สัญลักษณ์ที่แสดงการรับข้อมูลเหมือนกับสัญลักษณ์ Input แต่จะระบุข้อมูลที่รับเข้ามาผ่านทางคีย์บอร์ด
6		Display	สัญลักษณ์ที่ใช้ในการแสดงการแสดงผลข้อมูลเหมือนกับสัญลักษณ์ Output แต่จะเป็นการระบุอุปกรณ์ที่ใช้ในการแสดงผลเป็นจอคอมพิวเตอร์
7		Database	สัญลักษณ์การเก็บข้อมูล แบบแฟ้มข้อมูลหรือฐานข้อมูล
8		Connect	สัญลักษณ์ที่ใช้แสดงจุดที่มีเส้นทางการทำงาน 2 เส้นทางขึ้นไปมาบรรจบกัน
9		Off- page Connector	สัญลักษณ์ที่ใช้ในการแสดงจุดเชื่อมต่อระหว่างหน้า หากมีลำดับขั้นตอนการทำงานถ้ามีการข้ามไปอีกหน้า จะต้องใช้สัญลักษณ์นี้เป็นจุดเชื่อมต่อระหว่างหน้า
10		Flow Line	สัญลักษณ์ที่ใช้ในการแสดงทิศทางการทำงาน ซึ่งปลายเส้นจะมีหัวลูกศรเป็นการบอกทิศทาง ซึ่งจะมีได้เพียงหัวลูกศรเดียวบอกทิศทางการทำงานเท่านั้น

ที่มา : Wikipedia (2021)

สัญลักษณ์แผนภาพแสดงการทำงานแต่ละประเภท การนำไปใช้ในการอธิบายขั้นตอนการทำงาน จะต้องใช้สัญลักษณ์ให้ถูกต้องตามความหมาย ซึ่งถ้าหากใช้สัญลักษณ์ผิด อาจจะทำให้เกิดความเข้าหรือนำไปใช้ในการเขียนโปรแกรมผิดได้

3. หลักการเขียนแผนภาพแสดงลำดับขั้นตอนการทำงาน

การเขียนแผนภาพแสดงลำดับขั้นตอนการทำงาน เพื่อให้การสื่อสารระหว่างผู้นำเสนอและผู้รับสามารถเข้าใจตรงกัน จึงมีหลักการและเงื่อนไขการนำเสนอ ตามลำดับขั้นตอนการทำงานดังต่อไปนี้

3.1 ทิศทางการทำงานการกำหนดทิศทางขั้นตอนการทำงาน จะต้องเริ่มต้นจากกระบวนการแรกที่อยู่ด้านบนสุด เรียงลำดับลงมาด้านล่างและจากซ้ายมาขวา

3.2 การกำหนดข้อมูลหรือกระบวนการที่เขียนลงแผนภาพแสดงลำดับขั้นตอนการทำงานจะต้องเป็นข้อมูลหรือกระบวนการที่ได้จากการวิเคราะห์

3.3 ลูกศรบอกทิศทางของการทำงาน จะใช้หัวลูกศรที่อยู่ปลายทางเท่านั้น ห้ามมีลูกศรที่ไม่มีจุดเชื่อมต่อกับสัญลักษณ์ปลายทาง

3.4 เส้นทางการทำงานลีกเลี่ยงการทับเส้นกัน เนื่องจากจะทำให้การแปลความหมายของแผนภาพแสดงลำดับขั้นตอนการทำงานผิด แต่บางกรณีอาจมีความจำเป็น ดังนั้นจึงสามารถใช้โปรแกรมบางชนิดที่ออกมาให้ลักษณะของจุดตัดระหว่างเส้นทางในลักษณะของสะพานข้าม ซึ่งจะไม่ทำให้ไม่เกิดความเข้าใจผิดขั้นอตอนการทำงาน

3.5 สัญลักษณ์ที่นำมาใช้ในแต่ละขั้นตอนการทำงาน จะต้องสื่อให้ตรงกับความหมายที่จะนำเสนอหรืออธิบาย ถ้านำมาใช้ผิด จะทำการสื่อความหมายหรือเกิดความเข้าใจผิด เมื่อนำไปเขียนโปรแกรมอาจเกิดการทำงานไม่ถูกต้องของโปรแกรมได้

3.6 นำข้อมูลหรือการประมวลผลที่ได้จากการวิเคราะห์ มาเขียนลงในแผนภาพแสดงลำดับขั้นตอนการทำงานได้ ทำให้สื่อความหมายของกระบวนการทำงานได้ถูกต้องและรวดเร็ว

4. ประโยชน์ของแผนภาพแสดงลำดับขั้นตอนการทำงาน

แผนภาพแสดงลำดับขั้นตอนการทำงาน มีการนำมาใช้ตั้งแต่อดีตจนถึงปัจจุบัน ยังนิยมนำมาเพื่ออธิบายขั้นตอนการทำงานของระบบหรือโปรแกรม ซึ่งภาพรวมสามารถสรุปประโยชน์ของการนำมาใช้มีดังต่อไปนี้

4.1 สามารถมองภาพรวมและทำความเข้าใจในขั้นตอนการทำงานได้อย่างรวดเร็ว ชัดเจนมากยิ่งขึ้น

4.2 รูปแบบการนำเสนอผังโปรแกรมเป็นสากล บุคคลอื่นสามารถเข้าใจในหลักการทำงานที่ถูกต้องเช่นเดียวกัน โดยบุคคลที่นำผังโปรแกรมไปใช้เขียนโปรแกรม สามารถเขียนได้ทันทีและสามารถนำไปเขียนโปรแกรมได้ทุกภาษา

4.3 การตรวจสอบข้อผิดพลาดของการทำงานด้วยแผนภาพแสดงลำดับขั้นตอนการทำงานจะมองภาพรวมและตรวจสอบความผิดพลาดการทำงานได้อย่างชัดเจน ดังนั้นจึงสามารถตรวจสอบความผิดพลาดได้รวดเร็ว สามารถแก้ไข ปรับปรุงหรือพัฒนาขั้นตอนการทำงานของโปรแกรมใหม่ได้รวดเร็วขึ้น

การวิเคราะห์และออกแบบโปรแกรม

ขั้นตอนการพัฒนาโปรแกรม การวิเคราะห์และออกแบบโปรแกรมเป็นกระบวนการที่สำคัญ ซึ่งหากกระบวนการนี้ไม่ได้จัดทำหรือการจัดทำไม่ถูกต้อง จะส่งผลให้ไม่สามารถนำไปเขียนโปรแกรมได้หรือเมื่อนำไปเขียนโปรแกรมแล้ว จะเกิดการทำงานผิดพลาดหรือประสิทธิภาพที่ได้จากการทำงานของโปรแกรมจะต่ำ จากหลักการของขั้นตอนการพัฒนาโปรแกรม สามารถสรุปกระบวนการของการวิเคราะห์และออกแบบ โดยการทำงานของวิเคราะห์ มีขั้นตอน ได้แก่ ข้อมูลนำออก การประมวลผล และข้อมูลนำเข้า ส่วนของการออกแบบ ประกอบด้วยขั้นตอนการออกแบบอัลกอริทึม การกำหนดข้อมูลและการเขียนผังโปรแกรม ซึ่งมีรายละเอียดการทำงานของแต่ละโครงสร้างโปรแกรมดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1.2 คำนวณหาพื้นที่ของวงกลม ซึ่งได้มาจากสูตร ค่า pi*(รัศมี*รัศมี)

ขั้นตอนที่ 1 วิเคราะห์

ข้อมูลนำออก : พื้นที่

ประมวลผล : พื้นที่ = pi * (รัศมี*รัศมี)

(ค่า pi = 22/7)

ข้อมูลนำเข้า : รัศมี

ขั้นตอนที่ 2 ออกแบบ

อัลกอริทึม

1.1) รับข้อมูลรัศมี

1.2) คำนวณพื้นที่

พื้นที่ = pi * (รัศมี*รัศมี)

1.3) แสดงพื้นที่

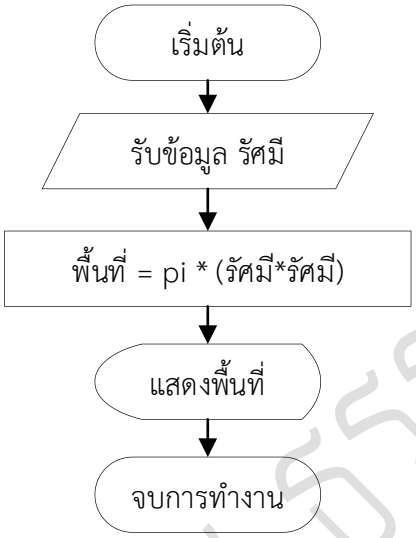
1.4) จบการทำงาน

กำหนดข้อมูล

2.1) พื้นที่ เก็บข้อมูลตัวเลขทศนิยม

2.2) รัศมี เก็บข้อมูลตัวเลขทศนิยม

ผังโปรแกรม



ภาพที่ 1.9 ผังโปรแกรมแบบเรียงลำดับ การหาพื้นที่วงกลม

ตัวอย่างที่ 1.3 คำนวณเงินรับที่ได้ ซึ่งคำนวณจากเงินเดือนหักเงินประกันสังคมจากเงินเดือน 5 เปอร์เซ็นต์รวมกับเงินล่วงเวลา

ขั้นตอนที่ 1 วิเคราะห์

ข้อมูลนำออก : เงินรับ

ประมวลผล : เงินรับ = เงินเดือน – เงินสะสมสวัสดิการ + เงินล่วงเวลา

เงินสะสมสวัสดิการ = (เงินเดือน * 5) / 100

ข้อมูลนำเข้า : เงินเดือน, เงินล่วงเวลา

ขั้นตอนที่ 2 ออกแบบ

1) อัลกอริทึม

1.1) รับข้อมูลเงินเดือน, เงินล่วงเวลา

1.2) คำนวณเงินประกันสังคม เงินสะสมสวัสดิการ=(เงินเดือน * 5)/100

1.3) คำนวณ เงินรับ= เงินเดือน–เงินสะสมสวัสดิการ+เงินล่วงเวลา

1.4) แสดงเงินรับ

1.5) จบการทำงาน

2) กำหนดข้อมูล

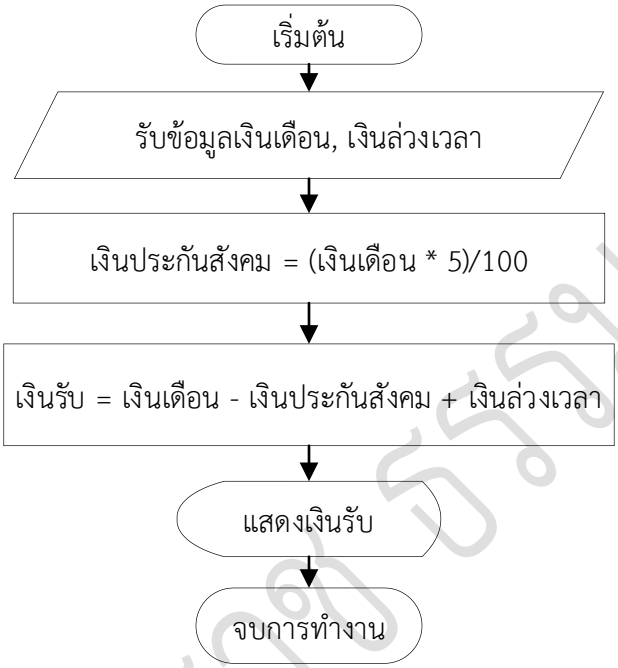
2.1) เงินรับ เก็บข้อมูลตัวเลขทศนิยม

2.2) เงินประกันสังคม เก็บข้อมูลตัวเลขทศนิยม

2.3) เงินเดือน เก็บข้อมูลตัวเลขทศนิยม

2.4) เงินค่าล่วงเวลา เก็บข้อมูลตัวเลขทศนิยม

3) ผังงาน



ภาพที่ 1.10 ผังโปรแกรมแบบเรียงลำดับ การคำนวณเงินรับหลังหักประกันสังคม

ตัวอย่างที่ 1.4 คำนวณเงินรับที่ได้ โดยถ้าเงินเดือนตั้งแต่ 15,000 บาท หักค่าประกัน 750 บาท ถ้าเงินเดือน 1,650 บาท แต่ไม่ถึง 15,000 บาท หักประกันสังคม 5 % แต่ถ้าต่ำกว่า 1,650 บาท ไม่ต้องหัก

ขั้นตอนที่ 1 วิเคราะห์

ข้อมูลนำออก : เงินประกันสังคม

ประมวลผล : เงินเดือน >= 15,000

เงินประกันสังคม = เงินเดือน – 750

เงินเดือน >= 1,650 < 15,000

เงินประกันสังคม = เงินเดือน (เงินเดือน *5/100)

ถ้าเงินเดือน < 1,650

เงินประกันสังคม = 0

ข้อมูลนำเข้า : เงินเดือน

ขั้นตอนที่ 2 ออกแบบ

1) อัลกอริทึม

1.1) รับข้อมูลเงินเดือน

1.2) คำนวณเงินประกันสังคม

ถ้าเงินเดือนมากกว่าหรือ 15,000

เงินประกันสังคม = เงินเดือน – 750

ถ้าเงินเดือน 1,650 แต่ไม่ถึง 15,000

เงินประกันสังคม = เงินเดือน (เงินเดือน *5/100)

ถ้าเงินเดือน ต่ำกว่า 1,650

เงินประกันสังคม = 0

1.3) แสดงเงินประกันสังคม

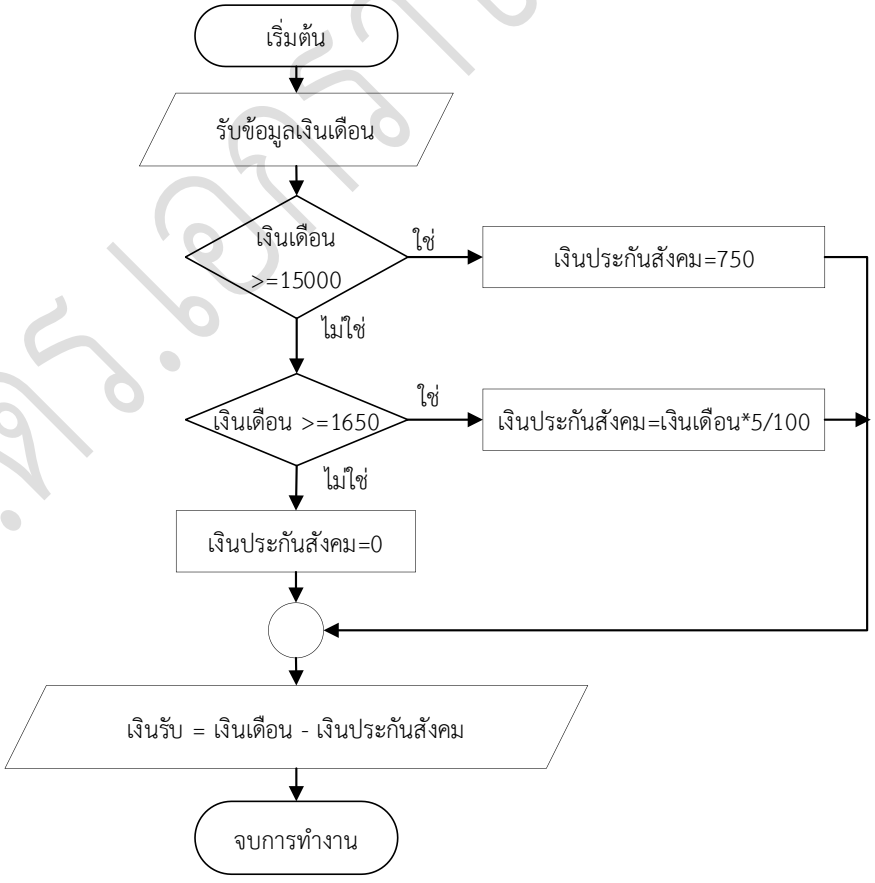
1.4) จบการทำงาน

2) กำหนดข้อมูล

2.1) เงินเดือน เก็บข้อมูลตัวเลขทศนิยม

2.2) เงินประกันสังคม เก็บข้อมูลตัวเลขทศนิยม

3) ผังงาน



ภาพที่ 1.11 ผังโปรแกรมแบบทางเลือก การหาเงินรับหลังหักประกันสังคม

ตัวอย่างที่ 1.5 หาเงินจ่ายค่าสินค้า หลังหักส่วนลด โดยมีเงื่อนไข ดังนี้

ราคาสินค้า 5,000 บาท ขึ้นไป ได้ส่วนลด 7 เปอร์เซ็นต์

ราคาสินค้า 1,000 บาท แต่ไม่ถึง 5,000 บาท ได้ส่วนลด 5 เปอร์เซ็นต์ .

ราคาสินค้าตั้งแต่ 100 บาท แต่ไม่ถึง 1,000 บาท ได้ส่วนลด 3 เปอร์เซ็นต์

ถ้าราคาสินค้าน้อยกว่า 100 บาท ไม่ได้รับส่วนลด

ขั้นตอนที่ 1 วิเคราะห์

ข้อมูลนำออก : เงินจ่ายจริง, เงินส่วนลด

ประมวลผล : เงินจ่ายจริง = ราคาสินค้า – เปอร์เซ็นต์ลด

เงินส่วนลด = (ราคาสินค้า * เปอร์เซ็นต์ลด) / 100

ราคาสินค้า >= 5000 เปอร์เซ็นต์ลด = 7

ราคาสินค้า >= 1000 และ < 5000 เปอร์เซ็นต์ลด = 5

ราคาสินค้า >= 100 และ < 1000 เปอร์เซ็นต์ลด = 3

ราคาสินค้า >= 0 และ < 100 เปอร์เซ็นต์ลด = 0

ข้อมูลนำเข้า : ราคาสินค้า

ขั้นตอนที่ 2 ออกแบบ

1) อัลกอริทึม

1.1) รับข้อมูลราคาสินค้า

1.2) ตรวจสอบส่วนลด

ราคาสินค้า >= 5000 เปอร์เซ็นต์ลด = 7

ราคาสินค้า >= 1000 และ < 5000 เปอร์เซ็นต์ลด = 5

ราคาสินค้า >= 100 และ < 1000 เปอร์เซ็นต์ลด = 3

ราคาสินค้า >= 0 และ < 100 เปอร์เซ็นต์ลด = 0

1.3) คำนวณเงินส่วนลด

เงินส่วนลด = (ราคาสินค้า * เปอร์เซ็นต์ลด) / 100

1.4) คำนวณเงินจ่ายจริง

เงินจ่ายจริง = ราคาสินค้า – เงินส่วนลด

1.5) แสดงเงินจ่ายจริง

1.6) จบการทำงาน

2) กำหนดข้อมูล

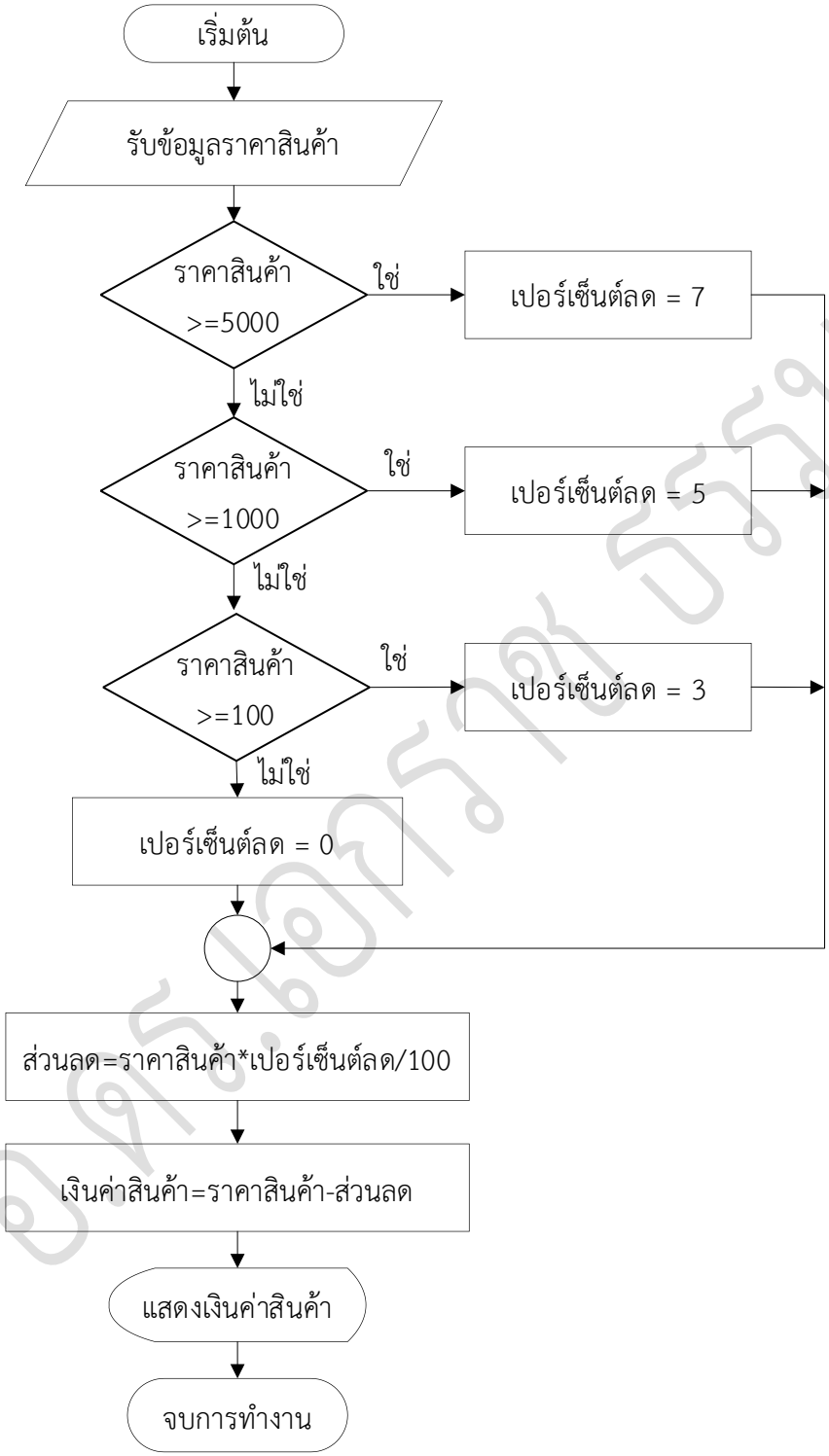
2.1) ราคาสินค้า เก็บข้อมูลตัวเลขทศนิยม

2.2) เปอร์เซ็นต์ลด เก็บข้อมูลตัวเลขทศนิยม

2.3) เงินส่วนลด เก็บข้อมูลตัวเลขทศนิยม

2.4) เงินจ่ายจริง เก็บข้อมูลตัวเลขทศนิยม

3) ผังโปรแกรม



ภาพที่ 1.12 ผังโปรแกรมแบบทางเลือก การหาเงินจ่ายจริงหลังหักส่วนลด

ตัวอย่างที่ 1.6 หาอายุเฉลี่ยของนักศึกษา จำนวน 5 คน

ขั้นตอนที่ 1 วิเคราะห์

ข้อมูลนำออก : อายุเฉลี่ย, อายุรวม

ประมวลผล : อายุเฉลี่ย = อายุรวม / จำนวนนักศึกษา

อายุรวม = อายุรวม + อายุ

จำนวนนักศึกษา = จำนวนนักศึกษา +1

ข้อมูลนำเข้า : อายุ

ขั้นตอนที่ 2 ออกแบบ

1) อัลกอริทึม

1.1) รับข้อมูลอายุ

1.2) คำนวณอายุรวม

อายุรวม = อายุรวม + อายุ

1.3) ตรวจสอบจำนวนนักศึกษา

จำนวนนักศึกษา = จำนวนนักศึกษา +1

1.4) ตรวจสอบจำนวนนักศึกษา

ถ้าจำนวนนักศึกษาน้อยกว่า 5

กลับไปทำงานในข้อที่ 1.1

1.5) คำนวณอายุเฉลี่ย

อายุเฉลี่ย = อายุรวม / จำนวนนักศึกษา

1.6) แสดงอายุเฉลี่ย

1.7) จบการทำงาน

2) กำหนดข้อมูล

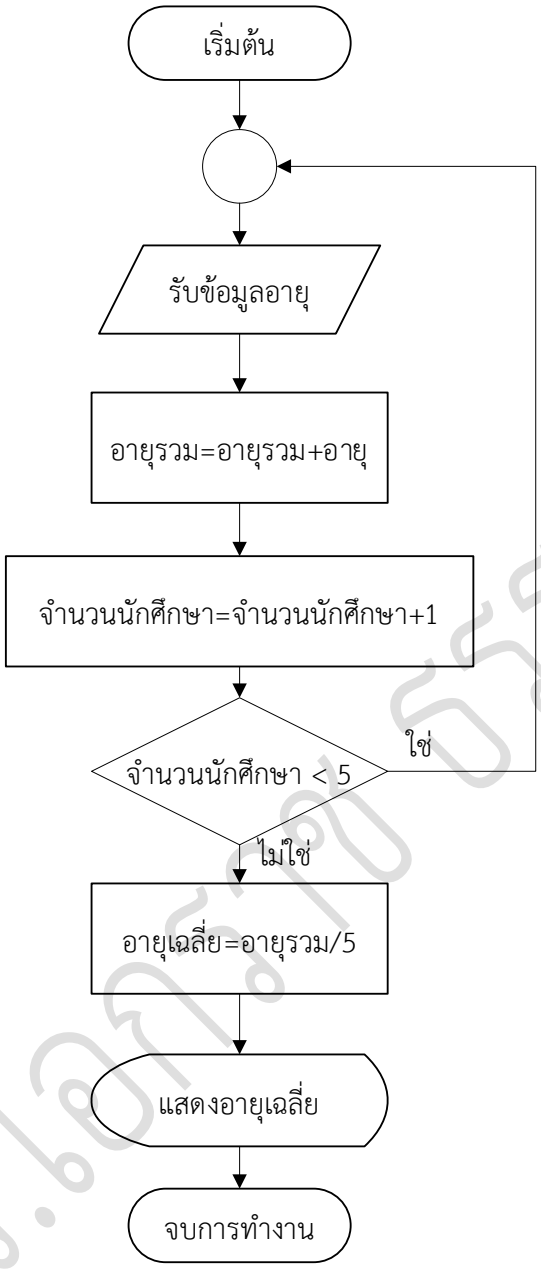
2.1) อายุ เก็บข้อมูลตัวเลขทศนิยม

2.2) จำนวนนักศึกษา เก็บข้อมูลตัวเลขจำนวนเต็ม

2.3) อายุรวม เก็บข้อมูลตัวเลขทศนิยม

2.4) อายุเฉลี่ย เก็บข้อมูลตัวเลขทศนิยม

3) ผังโปรแกรม



ภาพที่ 1.13 ผังโปรแกรมแบบทำซ้ำ หาอายุเฉลี่ยของนักศึกษา จำนวน 5 คน

จากตัวอย่างที่ 1.2 และ 1.3 ขั้นตอนการทำงานโครงสร้างของโปรแกรมแบบเรียงลำดับ ตัวอย่างที่ 1.4 และ 1.5 ขั้นตอนการทำงานโครงสร้างแบบทางเลือกและตัวอย่างที่ 1.6 ขั้นตอนการทำงานโครงสร้างแบบทำซ้ำ ข้อมูลทั้งหมดได้จากการวิเคราะห์และออกแบบ ขั้นตอนต่อไปจะเป็นการนำรายละเอียดที่ได้ไปเขียนโปรแกรม ซึ่งสามารถนำไปเขียนได้ทุกภาษา ขึ้นอยู่กับผู้เขียนโปรแกรมต้องการพัฒนาด้วยภาษาโปรแกรมคอมพิวเตอร์ใด ซึ่งเป็นขั้นตอนของการพัฒนาโปรแกรมต่อไป

สรุป

โปรแกรมคอมพิวเตอร์ มีหน้าที่ในการสั่งให้คอมพิวเตอร์ให้ทำงานตามที่ต้องการ โดยโปรแกรมคอมพิวเตอร์จะถูกสร้างขึ้นด้วยภาษาคอมพิวเตอร์แบ่งเป็น 2 ระดับ ประกอบด้วย ภาษาระดับต่ำและภาษาระดับสูง ซึ่งแต่ละภาษาจะมีคุณสมบัติ ข้อดีและข้อเสียแตกต่างกันออกไป โดยที่ภาษาระดับต่ำประกอบไปด้วย ภาษาเครื่องและภาษาแอสเซมบลี จะมีความยากในการสั่งงานคอมพิวเตอร์ แต่จะมีความรวดเร็วในการทำงานและใช้หน่วยความจำน้อย ส่วนภาษาระดับสูงนั้นจะต้องใช้ตัวแปลภาษา ดังนั้นจะสั่งงานง่ายเพราะเป็นภาษาที่ใกล้เคียงกับภาษามนุษย์ แต่ต้องผ่านตัวแปลภาษา จึงทำให้การสั่งงานคอมพิวเตอร์ทำงานได้ช้า เนื่องจากภาษาของคอมพิวเตอร์ คือ ภาษาเครื่อง ดังนั้นเมื่อโปรแกรมภาษาคอมพิวเตอร์ใดไม่ใช่ภาษาเครื่อง จะต้องใช้ตัวแปลภาษาเพื่อให้สามารถสั่งงานคอมพิวเตอร์ได้ การแปลภาษาประกอบด้วย ภาษาแอสเซมบลี ใช้ตัวแปลภาษาแอสเซมเบลอร์และภาษาระดับสูงจะใช้ตัวแปลภาษาออกเป็น 2 ชนิด ได้แก่ คอมไพเลอร์และอินเทอร์พรีเตอร์ ซึ่งคอมไพเลอร์จะมีลักษณะของการแปลภาษาที่แปลชุดคำสั่งทั้งหมดก่อนไปสั่งงานคอมพิวเตอร์ ในช่วงระหว่างการตรวจสอบชุดคำสั่งหากมีข้อผิดพลาด ตัวแปลภาษาจะทำการหยุดการแปลและแจ้งเตือนให้ทราบรายละเอียด ให้ทำการแก้ไขให้ถูกต้องและทำการแปลใหม่อีกครั้ง แต่อินเทอร์พรีเตอร์จะแปลคำสั่งทีละบรรทัด ถ้าคำสั่งถูกต้องในบรรทัดนั้น ก็สามารถสั่งงานคอมพิวเตอร์ได้ทันที โดยไม่ต้องหยุดการทำงาน

รหัสเทียม หรือ ซูโดโค้ด คือ การเขียนอัลกอริทึมโดยใช้ประโยคภาษาอังกฤษที่สื่อความหมายเข้าใจง่าย สามารถอ่านแล้วเข้าใจได้โดยทันที การพัฒนาโปรแกรมคอมพิวเตอร์ อาศัยหลักทฤษฎีของวงจรการพัฒนาซอฟต์แวร์ ซึ่งแบ่งขั้นตอนการพัฒนาระบบออกเป็นทั้งหมด 5 ขั้นตอน ประกอบไปด้วย การวางแผนแก้ปัญหา การวิเคราะห์ปัญหา การออกแบบขั้นตอนการทำงาน การพัฒนาโปรแกรม การประเมินผล ซึ่งการพัฒนาโปรแกรมให้ถูกต้องกับการนำไปใช้งาน จะต้องเข้าใจคุณสมบัติและการทำงานในระบบคอมพิวเตอร์ การวิเคราะห์และออกแบบโปรแกรม มีความสำคัญในกระบวนการของการพัฒนาโปรแกรม โดยนำปัญหาได้มา มาทำการวิเคราะห์ ซึ่งประกอบไปด้วย 3 ด้าน โดยเริ่มต้นจาก การวิเคราะห์ข้อมูลนำออก การประมวลผลและข้อมูลนำเข้า นำผลของการวิเคราะห์ที่ได้ไปสู่กระบวนการออกแบบขั้นตอนการพัฒนาโปรแกรม โดยการออกแบบโปรแกรมประกอบด้วย การเขียนอัลกอริทึมหรือขั้นตอนการทำงานของโปรแกรม นำข้อมูลที่ได้จากการวิเคราะห์ โดยเริ่มต้นจากข้อมูลนำเข้า การประมวลผลและข้อมูลนำออก กำหนดเป็นขั้นตอนเรียงลำดับการทำงาน นำข้อมูลนำเข้าและข้อมูลนำออกที่ได้จากการวิเคราะห์ ทำการกำหนดชนิดของข้อมูล นำอัลกอริทึมที่ได้มาสร้างเป็นแผนภาพการทำงานของโปรแกรม ก่อนนำไปเขียนเป็นโปรแกรมคอมพิวเตอร์ต่อไป

แบบฝึกหัด

1. ให้ผู้เรียนอธิบายหลักการทำงานของระบบคอมพิวเตอร์และโปรแกรมคอมพิวเตอร์
2. ให้ผู้เรียนอธิบายความแตกต่างระหว่างภาษาคอมพิวเตอร์มีอย่างไรบ้าง พร้อมทั้งอธิบายถึงข้อดีและข้อเสียของแต่ละภาษาคอมพิวเตอร์
3. ให้ผู้เรียนบอกตัวแปลภาษาคอมพิวเตอร์มีทั้งหมดกี่ประเภท พร้อมทั้งอธิบายการทำงานของแต่ละตัวแปลภาษาโปรแกรมคอมพิวเตอร์
4. ให้ผู้เรียนยกตัวอย่างโปรแกรมที่ใช้ตัวแปลภาษาแบบคอมไพเลอร์และอินเทอร์พรีเตอร์ พร้อมทั้งอธิบายลักษณะการทำงานประกอบ
5. ให้ผู้เรียนอธิบายขั้นตอนของการพัฒนาโปรแกรมและแต่ละขั้นตอนมีการทำงานอย่างไร
6. ให้ผู้เรียนอธิบายแผนภาพของผังโปรแกรม พร้อมทั้งความหมายและหน้าที่ของแต่ละสัญลักษณ์
7. ให้ผู้เรียนอธิบาย การวิเคราะห์โจทย์ปัญหามีทั้งหมดกี่ขั้นตอน แต่ละขั้นตอนมีหลักการวิเคราะห์อย่างไรบ้าง
8. ให้ผู้เรียนอธิบายการออกแบบขั้นตอนในการแก้โจทย์ปัญหา เพื่อพัฒนาโปรแกรม พร้อมทั้งอธิบายวิธีการการออกแบบแต่ละขั้นตอน
9. ให้ผู้เรียนบอกขั้นตอนการเขียนอัลกอริทึม พร้อมทั้งอธิบายแต่ละขั้นตอน
10. ให้ผู้เรียนสรุปขั้นตอนของการวิเคราะห์และออกแบบโปรแกรม โดยแต่ละส่วนประกอบด้วยอะไรบ้าง

เอกสารอ้างอิง

ราชบัณฑิตยสภา. (2564). **ศัพท์บัญญัติสำนักงานราชบัณฑิตยสภา.** (ออนไลน์) 10 เมษายน 2564 (อ้างเมื่อ 10 เมษายน 2564). จาก: https://coined-word.orst.go.th/

โอภาส เอี่ยมสิริวงศ์ และ สมโภชน์ ชื่นเอี่ยม. (2560). **การเขียนโปรแกรมคอมพิวเตอร์**. กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

โอภาส เอี่ยมสิริวงศ์. (2559). **โครงสร้างข้อมูลและอัลกอริทึม**. กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

Evans, David. (2011). **Introduction to Computing Explorations in Language, Logic, and Machines**. Charlottesville: University of Virginia.

Farrell, J. (2011). **A Beginner's Guide to Programming Logic and Design: Comprehensive Version**. 6thed. Massachusetts: Cengage Learning.

Haverbeke, Marijn. (2014). **Eloquent JavaScript A Modern Introduction to Programming**. Cambridge: Massachusetts Institute of Technology.

Jorgensen, Ed. (2016). x86-64 Assembly Language Programming with Ubuntu. Las Vegas: University of Nevada.

Karumanchi, Narasimha. (2014). **Data Structures and Algorithms Mad Easy**. Mumbai: Indian Institute of Technology Bombay.

Leon, Alexis. (2015). **Software Configuration Management Handbook**. 3rded. London. British Library.

Liang, Y. Daniel. (2015). **Introduction to Java Programming**. 10thed. Georgia: Pearson.

Norton, Peter. (2008). **Introduction to Computers**. 6thed. New Delhi: Tata McGraw-Hill.

Oram, Etuari and Naik, Bighnaraj. 2010. **PROGRAMMING IN "C"**. Odisha: Veer Surendra Sai University of Technology.

Prettyman, S. (2016). **Learn PHP 7**. New York: Appress.

Suchato, Atiwong. (2011). **Learning Computer Programming Using Java**. Bangkok: Chulalongkorn University.

Thakur, Sumit. (2021). **Difference Between Software and Program.** (online) (cited 9 June 2021). Available from: http://careersplay.com/difference-between-software-and-program/

Watt, D. A., and Brown, D. F. (2000). **Programming language processors in Java: compilers and interpreters**. London: Pearson.

Wikipedia. (2021). Flowchart. (online) (cited 2 April 2021). Available from: https://en.wikipedia.org/wiki/Flowchart