บทที่ 2 หลักการเขียนโปรแกรม (Programming Fundamental)

สาระการเรียนรู้

- แนวคิดการเขียนโปรแกรม
- ความหมายของ Structure Programming
- การเริ่มต้นเขียนโปรแกรม
- หลักการเขียนโปรแกรมเบื้องต้น
- คุณสมบัติของนักเขียนโปรแกรมที่ดี
- ลักษณะของโปรแกรมที่ดี
- ตัวอย่างการวิเคราะห์งานและการเขียนโปรแกรม

สมรรถนะการเรียนรู้

- ขอกความหมายของ Structure Programming ได้
- อธิบายโครงสร้างพื้นฐานของ Structure Programming ได้
- ขอกองค์ประกอบหลักของโครงสร้างพื้นฐานได้
- ขอกลำดับการพัฒนาการเขียนโปรแกรมสำหรับผู้เริ่มต้นได้
- อธิบายการวิเคราะห์งานได้
- อธิบายการออกแบบโปรแกรมได้
- ขอกข้อพิจารณาในการเขียนโปรแกรมได้
- ขอกความหมายของการเขียนโปรแกรมได้
- อธิบายขั้นตอนการตรวจสอบข้อผิดพลาดของโปรแกรมได้
- อธิบายวิธีการทดสอบความถูกต้องของโปรแกรมได้
- อธิบายการทำเอกสารประกอบโปรแกรมได้
- อธิบายการบำรุงรักษาโปรแกรมได้
- ขอกคุณสมบัติของนักโปรแกรมที่ดีได้
- ขอกลักษณะของโปรแกรมที่ดีได้
- แก้ปัญหาโจทย์การวิเคราะห์งานได้

แผนผังความคิด (Mind Mapping) ของหน่วยการเรียนรู้

แนวคิดการเขียนโปรแกรม

แนวทางและวิธีการเขียนโปรแกรมคอมพิวเตอร์ คือ **“การเขียนโปรแกรมทุกภาษานั้นเหมือนกัน”** สิ่งที่แตกต่างกันของแต่ละภาษาคือ Syntax หรือไวยากรณ์ภาษา โดยอาศัยประสบการณ์จากภาษาหนึ่งไปใช้ในอีกภาษาหนึ่ง โดยเรียนรู้ในเรื่องการเขียนโปรแกรมเชิงโครงสร้าง (Structure Programming) มีโครงสร้างหลัก 3 โครงสร้าง คือ แบบลำดับ แบบการเลือกกระทำตามเงื่อนไข และแบบทำซ้ำ กระบวนการของแต่ละโครงสร้างประกอบด้วย Input, Process, Output ถ้าเราเขียนโปรแกรมอะไรในภาษาหนึ่งได้แล้ว การเขียนโปรแกรมแบบนั้นในภาษาอื่นย่อมไม่ใช่เรื่องยากอีกต่อไป เพียงแต่ต้องศึกษาถึง Syntax หรือ รูปแบบการเขียนของภาษาใหม่นั้นเพิ่มเติม แล้วนำประสบการณ์ที่เคยเขียน ไปสั่งให้ภาษาใหม่ทำงานตามต้องการ ผู้สอนจึงมักสนับสนุนให้ผู้เรียนได้ศึกษาภาษาที่ไม่มี ตัวช่วยมาก เพื่อให้เข้าใจในหลักการและขั้นตอนการทำงานอย่างละเอียดชัดเจนจากการทำงานของตัวแปลภาษาที่มีตัวช่วยน้อย ทำงานบน DOS สามารถแปลเป็น exe และนำไปใช้ได้โดยไม่ยุ่งยาก เช่น C, Pascal, Basic หรือ Clipper เป็นต้น

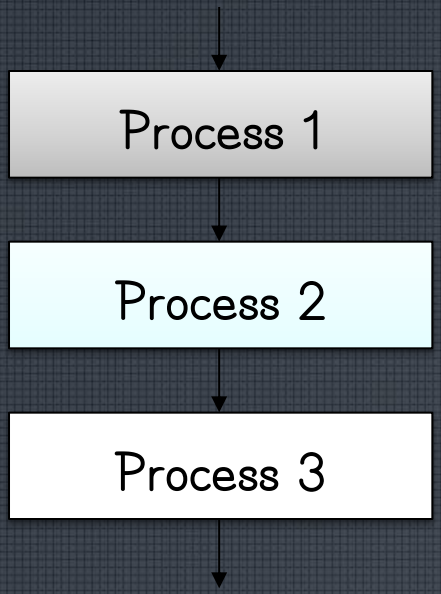
การเขียนโปรแกรมโครงสร้าง (Structure Programming)

การเขียนโปรแกรมแบบมีโครงสร้าง หรือ การโปรแกรมโครงสร้าง คือ การกำหนดขั้นตอนให้เครื่องคอมพิวเตอร์ทำงานโดยมีโครงสร้างการควบคุมพื้นฐาน 3 รูปแบบ ได้แก่

- การทำงานแบบตามลำดับ (Sequence)
- การเลือกกระทำตามเงื่อนไขหรือการตัดสินใจทางเลือก (Decision)
- การทำซ้ำ (Loop)

การทำงานแบบตามลำดับ (Sequence)

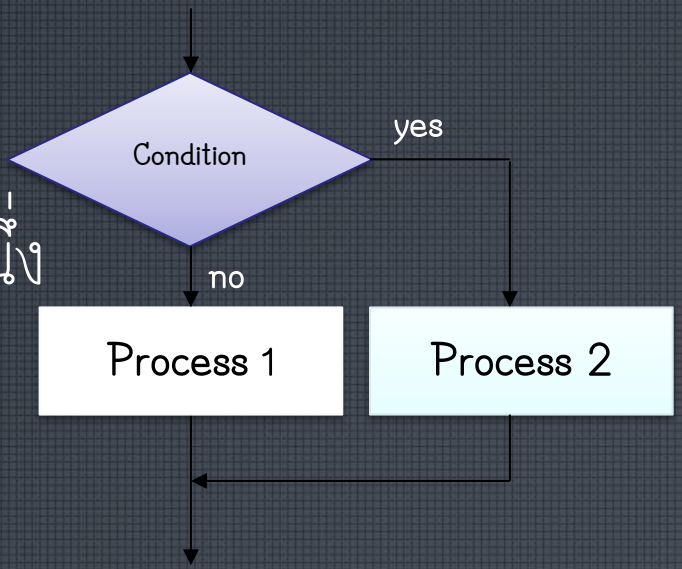
คือ การเขียนให้ทำงานจากบนลงล่างเขียนคำสั่งเป็นบรรทัด และ ทำทีละบรรทัดจากบรรทัดบนสุดลงไปจนถึงบรรทัดล่างสุด ส่วนใหญ่จะให้มีการทำงาน 3 กระบวนการ คือ อ่านข้อมูล คำนวณ และพิมพ์



การเลือกกระทำตามเงื่อนไขหรือการตัดสินใจทางเลือก (Decision)

คือการเขียนโปรแกรมเพื่อตัดสินใจในการเลือกกระทำกระบวนการใดกระบวนการหนึ่ง โดยปกติจะมีเหตุการณ์ให้ทำ 2 กระบวนการ คือ

- เงื่อนไขเป็นจริงจะกระทำกระบวนการหนึ่ง
- เงื่อนไขเป็นเท็จจะกระทำอีกกระบวนการหนึ่ง หรือไม่ทำ



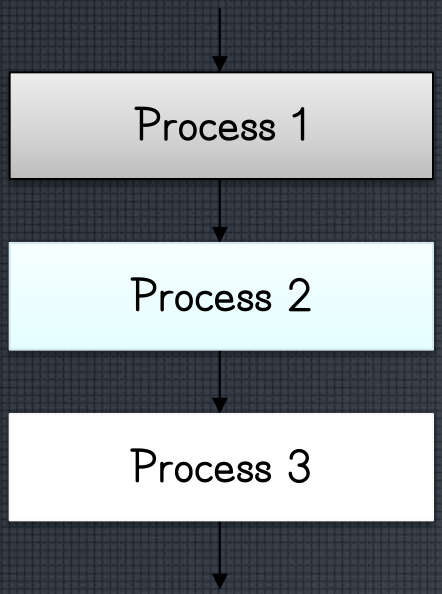
แต่ถ้าซับซ้อนมากขึ้น จะต้องใช้เงื่อนไขหลายชั้น เช่น การตัดเกรด เป็นต้น

การทำซ้ำหรือการวนรอบ (Repeation or Loop)

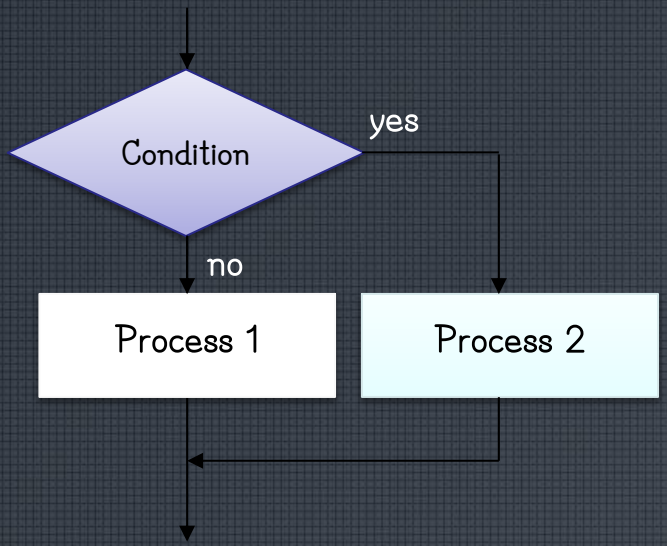
คือ การทำกระบวนการหนึ่งหลายครั้งโดยมีเงื่อนไขในการควบคุม หมายถึง การทำซ้ำเป็นหลักการที่ทำความเข้าใจได้ยากกว่า 2 รูปแบบแรก เพราะการเขียนโปรแกรมแต่ละภาษาจะไม่แสดงภาพอย่างชัดเจนเหมือนการเขียนผังงาน(Flowchart) ผู้เขียนโปรแกรมต้องจินตนาการ ถึงรูปแบบการทำงาน และใช้คำสั่งควบคุมด้วยตนเองตัวอย่างผังงานที่นำมาแสดงนี้เป็นการแสดงคำสั่งทำซ้ำ(do while) ซึ่งหมายถึง การทำซ้ำในขณะที่เป็นจริง และเลิกการทำซ้ำเมื่อเงื่อนไขเป็นเท็จ

แสดงผังงานตามโครงสร้างพื้นฐานทั้ง 3 แบบ

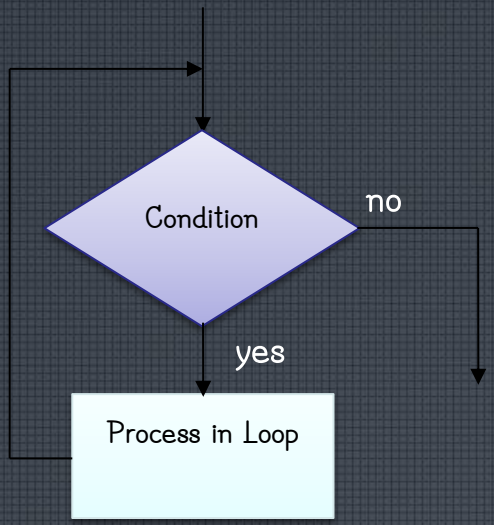
Sequence



Decision



Repeation



เริ่มต้นเขียนโปรแกรม

เริ่มต้นเขียนโปรแกรม เริ่มต้นตรงไหน : สำหรับผู้หัดเริ่มต้นเขียนโปรแกรม คงจะต้องเริ่มจากรูปแบบที่ง่าย ๆ คือโครงสร้างแบบตามลำดับ (Sequence) ซึ่งจะต้องมีองค์ประกอบ 4 ส่วน คือ

- ส่วนประกาศตัวแปร
- ส่วนอินพุต หรือ กำหนดค่า
- ส่วนประมวลผล
- ส่วนเอาต์พุต หรือ ส่วนแสดงผล

เริ่มต้นเขียนโปรแกรมทำอย่างไร

- เลือกภาษา สำหรับนักเรียน/นักศึกษานั้นง่ายที่จะเลือก เพราะผู้สอนจะคอยชี้แนะให้
- แหล่งเรียนรู้หรือข้อมูลอ้างอิง ได้แก่ หนังสือหรือตำราจากห้องสมุด แต่ในปัจจุบันแหล่งเรียนรู้ที่นิยมกันมากที่สุดคือการสืบค้นข้อมูลทางอินเทอร์เน็ต มีคลิปวีดีโอหรือเว็บไซต์เกี่ยวกับการเรียนรู้การเขียนโปรแกรมอยู่มากมาย
- หาตัวแปลภาษา ทุกภาษาต้องมีตัวแปลภาษา มีหลายภาษาที่ถูกสร้างเป็น Free Compiler สามารถหาดาวน์โหลดได้จากเว็บไซต์ต่าง ๆ สืบค้นใน google ซึ่งเป็น Search Engine ที่นิยมใช้กันมากที่สุด
- เขียนโปรแกรมตัวแรกที่ง่าย เช่น พิมพ์ข้อความ หรือ พิมพ์ตัวเลข เป็นต้น
- ลำดับต่อไปจึงเขียนโปรแกรมประกาศตัวแปร กำหนดค่าให้กับตัวแปร แล้วนำมาประมวลผลและแสดงผล
- จากนั้นเขียนโปรแกรมรับค่าข้อมูลจากแป้นพิมพ์ ทั้งที่เป็นข้อความและตัวเลขชนิดต่าง ๆ
- เขียนโปรแกรมคำนวณหาค่า โดยใช้สูตร เช่น การหาพื้นที่ ปริมาตร การแปลงหน่วยข้อมูล เป็นต้น

เริ่มต้นเขียนโปรแกรมทำอย่างไร

- ศึกษาการเลือกตามเงื่อนไข เช่น การคิดเกรด ค่าส่วนลด ค่าตอบแทน ที่มีเงื่อนไขแตกต่างกัน เป็นต้น
- ศึกษาการทำซ้ำ ในรูปแบบต่าง ๆ เช่น พิมพ์ 1 ถึง 10 หรือพิมพ์ตารางสูตรคูณ เป็นต้น
- พัฒนาโปรแกรมให้อยู่ในรูปแบบของโปรแกรมย่อย หรือเขียนฟังก์ชันขึ้นใช้เอง
- เขียนเมนู เพื่อเลือกกระทำโปรแกรมตามตัวเลือก
- เขียนโปรแกรมใช้ตัวแปรอาร์เรย์ ตัวแปรโครงสร้าง และการจัดเรียงข้อมูล
- ติดต่อแฟ้มข้อมูล เพื่ออ่านมาแสดงผล หรือปรับปรุงข้อมูลได้
- ทำรายงานจากการเชื่อมแฟ้มหลายแฟ้ม โดยกำหนดได้หลายตัวเลือก
- เขียนโปรแกรมเพิ่มข้อมูล เช่น ซื้อ ขาย ยืม คืนหรือโปรแกรมลงทะเบียนนักศึกษา เป็นต้น
- สร้างโปรแกรมขึ้นมาระบบหนึ่งให้สมบูรณ์ (ความสมบูรณ์ก็คือการสนองทุกความต้องการของผู้ใช้)

หลักการเขียนโปรแกรมเบื้องต้น

คอมพิวเตอร์เป็นอุปกรณ์ทางอิเล็กทรอนิกส์อย่างหนึ่ง ซึ่งไม่สามารถทำงานด้วยตนเองได้ แต่จะสามารถทำงานได้ตามชุดคำสั่งในโปรแกรมที่ป้อนเข้าสู่เครื่อง ซึ่งจะทำงานตามคำสั่งทีละคำสั่ง (Step by Step) โดยคำสั่งที่เครื่องคอมพิวเตอร์สามารถเข้าใจได้ จะต้องอยู่ในรูปแบบของภาษาเครื่อง (Machine Language) แต่ถ้ามีการเขียนด้วยภาษาอื่นที่ไม่ใช่ภาษาเครื่อง ถ้าเป็นภาษาระดับต่ำ (Low-level Language) เช่น ภาษาแอสเซมบลี (Assembly) ก็จะต้องมีตัวแปลภาษาเรียกว่า แอสเซมเบลอร์ (Assembler) ทำการแปลให้เป็นภาษาเครื่อง หรือที่เป็นภาษาระดับสูง (High-level Language) ก็จะต้องมีตัวแปลภาษา อย่างเช่น คอมไพเลอร์ (Compiler) หรือ อินเตอร์พรีเตอร์ (Interpreter) ทำการแปลภาษาระดับสูงนั้นให้เป็นภาษาเครื่องอีกทีหนึ่ง

หลักการเขียนโปรแกรมเบื้องต้น

ในการเขียนโปรแกรมหรือภาษาคอมพิวเตอร์นี้ โดยทั่วไปแล้วแต่ละภาษาจะมีหลักเกณฑ์ในการเขียนและการออกแบบโปรแกรมเหมือนกัน ซึ่งสามารถที่จะแบ่งขั้นตอนการเขียนโปรแกรมออกได้เป็น 7 ขั้นตอน ดังนี้

1.ขั้นตอนการวิเคราะห์งานหรือการวิเคราะห์ปัญหา (Analysis the Problem)

2.ขั้นตอนการออกแบบโปรแกรม (Design a Program)

3.ขั้นตอนการเขียนโปรแกรม (Coding)

4.ขั้นตอนการตรวจสอบข้อผิดพลาดของโปรแกรม (Testing and Debugging)

5.ขั้นตอนการทดสอบความถูกต้องของโปรแกรม (Testing and Validating)

6.ขั้นตอนการทำเอกสารประกอบโปรแกรม (Documentation)

7.ขั้นตอนการบำรุงรักษาโปรแกรม (Program Maintenance)

ขั้นตอนที่ 1 ขั้นตอนการวิเคราะห์งานหรือการวิเคราะห์ปัญหา

ขั้นตอนนี้เป็นขั้นตอนแรกสุดที่นักเขียนโปรแกรมจะต้องทำก่อนที่จะลงมือเขียนโปรแกรมจริง ๆ เพื่อทำความเข้าใจกับปัญหาที่เกิดขึ้น และค้นหาจุดมุ่งหมายหรือสิ่งที่ต้องการ

1. **สิ่งที่ต้องการ**
2. **การระบุข้อมูลเข้า** (Input)
3. **การระบุข้อมูลออก** (Output)
4. **กำหนดตัวแปร**
5. **กำหนดวิธีการประมวลผล**

ขั้นตอนที่ 2 การออกแบบโปรแกรม (Design a Program)

โดยใช้เครื่องมือมาช่วยในการออกแบบ ในขั้นตอนนี้ยังไม่ได้เป็นการเขียนโปรแกรมจริง ๆ แต่จะช่วยให้การเขียนโปรแกรมทำได้ง่ายขึ้น โดยสามารถเขียนตามขั้นตอนที่ได้ออกแบบไว้ในขั้นตอนนี้ และช่วยให้การเขียนโปรแกรมมีข้อผิดพลาดน้อยลง ช่วยให้การตรวจสอบการทำงานของโปรแกรม ทำให้ทราบขั้นตอนการทำงานของโปรแกรมได้อย่างรวดเร็ว โดยไม่ต้องไปไล่ดูจากตัวโปรแกรมจริง ๆ โดยเครื่องมือที่ใช้ในการออกแบบโปรแกรมมีอยู่หลายอย่าง ซึ่งวิธีการที่นิยมมาใช้สำหรับการออกแบบโปรแกรม เช่น

- อัลกอริทึม (Algorithm)
- ผังงาน (Flowchart)
- รหัสเทียมหรือรหัสจำลอง (Pseudo-code)
- แผนภูมิโครงสร้าง (Structure Chart)

อัลกอริทึม (Algorithm)

อัลกอริทึมเป็นเครื่องมือที่ช่วยในการออกแบบโปรแกรม โดยใช้ข้อความที่เป็นภาษาพูดจะเป็นภาษาอังกฤษหรือภาษาไทยก็ได้ ในการอธิบายการทำงานของโปรแกรมที่เป็นลำดับขั้นตอนจะข้ามไปข้ามมาไม่ได้ นอกจากจะต้องเขียนสั่งไว้ต่างหาก

ผังงาน (Flowchart)

ผังงานเป็นเครื่องมือที่ช่วยในการออกแบบโปรแกรม โดยใช้สัญลักษณ์รูปภาพ แสดง ขั้นตอนการเขียนโปรแกรม หรือขั้นตอนในการแก้ปัญหาทีละขั้น และมีเส้นที่แสดงทิศทางการไหลของข้อมูล ตั้งแต่จุดเริ่มต้นจนกระทั่งได้ผลลัพธ์ตามที่ต้องการ ซึ่งจะทำให้ผู้อ่านสามารถอ่านและทำความเข้าใจได้โดยง่าย โดยเฉพาะขั้นตอนหลัก ๆ ที่จำเป็นในโปรแกรมหนึ่ง ๆ

รหัสเทียมหรือรหัสจำลอง (Pseudo-code)

รหัสจำลองจะมีการใช้ข้อความที่เป็นภาษาอังกฤษ ในการแสดงขั้นตอนการแก้ปัญหา แต่จะมีการใช้คำเฉพาะ (Reserve words) ที่มีอยู่ในภาษาโปรแกรม มาช่วยในการเขียน โครงสร้างของรหัสจำลองจึงมีส่วนที่คล้ายกับการเขียนโปรแกรมมาก ดังนั้น รหัสจำลองจึงเป็นเครื่องมืออีกแบบหนึ่งที่เป็นที่นิยมใช้กันในการออกแบบโปรแกรม

แผนภูมิโครงสร้าง (Structure Chart)

แผนภูมิโครงสร้าง การใช้แผนภูมิโครงสร้างจะเป็นการแบ่งงานใหญ่ออกเป็นมอดูลย่อย ๆ ซึ่งเรียกว่า การออกแบบจากบนลงล่าง

(Top-Down Design)

แต่ละมอดูลย่อยก็ยังสามารถแตกออกได้อีกจนถึงระดับล่างสุดที่สามารถเขียนโปรแกรมได้อย่างง่าย มักใช้ในการออกแบบระบบมากกว่ารายละเอียดของการเขียนโปรแกรม

Module

ถ้าลักษณะงานมีความซับซ้อน ขั้นตอนแรกในการออกแบบจะต้องทำการแบ่งการทำงานออกเป็นการทำงานย่อยที่ไม่มีความซับซ้อนมาก แล้วนำการทำงานย่อยเหล่านั้น มารวมกันและจัดการทำงานย่อยนั้นให้เป็นการทำงานของวิธีการแก้ปัญหาทั้งหมด จะทำให้สามารถทำความเข้าใจถึงขั้นตอนทั้งหมดของโปรแกรม

การทำงานของขั้นตอนนี้เป็นขั้นตอนแยกรายละเอียดของขั้นตอนวิธีการทำงาน ที่ได้จากขั้นตอนของการรวบรวมลักษณะของปัญหาให้เป็นส่วนของส่วนจำเพาะหรือมอดูล (Module) และลักษณะของโครงสร้างข้อมูลที่ใช้

ตัวอย่าง

ต้องการโปรแกรมสำหรับคำนวณหาค่าเฉลี่ยของคะแนนสอบ และแสดงค่าเฉลี่ยของคะแนน สามารถทำการแยกเป็น 4 มอดูล ได้ดังนี้

- **มอดูลรับข้อมูลเข้า**

ทำหน้าที่สำหรับเก็บข้อมูลเข้าสำหรับการทำงานของโปรแกรม ข้อมูลเข้า คือค่าของคะแนนสอบ นำไปเก็บไว้ในตารางสำหรับเก็บคะแนนสอบ

- **มอดูลตรวจสอบข้อมูล**

ทำหน้าที่สำหรับตรวจสอบความถูกต้องของคะแนนในตารางเก็บคะแนนสอบว่าคะแนนที่เก็บไว้นั้นมีค่าเกินช่วงของคะแนนหรือไม่

- **มอดูลคำนวณหาค่าเฉลี่ย**

ทำหน้าที่สำหรับคำนวณหาค่าเฉลี่ยของคะแนนสอบ เฉพาะข้อมูลที่ถูกต้อง ไม่สนใจข้อมูลที่ผิดพลาด

- **มอดูลแสดงผลลัพธ์**

ทำหน้าที่แสดงค่าเฉลี่ยของคะแนนสอบ

ผลลัพธ์ที่ได้จากขั้นตอนนี้

- มอดูลทั้งหมดที่มีการใช้งานสำหรับการแก้ปัญหานี้
- ความสัมพันธ์ระหว่างมอดูลแต่ละมอดูลว่ามีมอดูลใดถูกเรียกใช้งานโดยมอดูลใดบ้าง
- วิธีการเรียกใช้งานแต่ละมอดูล เป็นการระบุลักษณะของข้อมูลที่ส่งผ่านระหว่างแต่ละมอดูล
- ลักษณะและการทำงานของแต่ละมอดูล เป็นรายละเอียดของการทำงานในแต่ละมอดูล
- ลักษณะชนิดของข้อมูลที่ใช้และการกระทำต่าง ๆ ที่เกิดขึ้นกับลักษณะของชนิดข้อมูล

การออกแบบมอดูลโปรแกรม

หลังจากที่ทำการแยกว่าการทำงานของโปรแกรมทั้งหมดว่ามีจำนวนมอดูลที่ใช้สำหรับการทำงานทั้งหมดจำนวนกี่มอดูล ขั้นตอนต่อไป คือการนำแต่ละมอดูลมาทำการออกแบบ การออกแบบมอดูลให้ทำการออกแบบทีละมอดูล โดยเลือกวิธีการทำงานที่เหมาะกับหน้าที่การทำงานของมอดูลนั้น แล้วนำมาเขียนเป็นขั้นตอนวิธีการ (Algorithm) สำหรับการทำงานของมอดูลนั้น ตั้งแต่ตอนแรกจนถึงขั้นตอนสุดท้าย เพื่อนำแต่ละมอดูลไปเขียนเป็นโปรแกรมคอมพิวเตอร์ต่อไป

เมื่อเลือกขั้นตอนวิธีการทำงานที่เหมาะสมสำหรับแต่ละมอดูล การเขียนอธิบายขั้นตอนวิธีการทำงานของแต่ละมอดูล จะเขียนในลักษณะของรหัสเทียม (Pseudo code) หรือเขียนเป็นผังงาน (Flow chart) เพื่อที่จะนำไปเขียนเป็นโปรแกรมคอมพิวเตอร์ต่อไป

ขั้นตอนที่ 3 การเขียนโปรแกรม (Coding)

เป็นการนำเครื่องมือที่ถูกสร้างขึ้นจากขั้นตอนการออกแบบมาแปลให้เป็นโปรแกรมคอมพิวเตอร์ ซึ่งในการสร้างโปรแกรมคอมพิวเตอร์นั้น เราสามารถเลือกใช้ ***ภาษาคอมพิวเตอร์หรือภาษาโปรแกรม*** ได้หลายภาษา ตั้งแต่ภาษาระดับต่ำ เช่น ภาษาแอสเซมบลี จนถึงภาษาระดับสูง เช่น ภาษาเบสิก ภาษาโคบอล ภาษาปาสคาล ภาษาซี ซึ่งแต่ละภาษาจะมีรูปแบบ โครงสร้าง หรือไวยากรณ์ของภาษาที่แตกต่างกันออกไป

ข้อพิจารณาในการเขียนโปรแกรม

- ความสามารถของเครื่องคอมพิวเตอร์และระบบปฏิบัติการที่ใช้
- ความถนัดและความชำนาญของผู้เขียนโปรแกรม
- ลักษณะและประเภทของงาน มีความเหมาะสมกับภาษาหรือโปรแกรมที่ใช้
- ควรเลือกภาษาที่ได้รับความนิยม และมีการพัฒนาอย่างต่อเนื่อง

การเขียนโปรแกรมคอมพิวเตอร์(Computer Programming)

การเขียนโปรแกรม (Programming)) หรือ การเขียนโค้ด (Coding) เป็นขั้นตอนการเขียน การตรวจสอบหรือทดสอบโปรแกรมภาษาคอมพิวเตอร์ หรือจะเรียกรวม ๆ ว่าเป็นการสร้างโปรแกรมต้นฉบับ (Source Code) ซึ่งซอร์สโค้ดนั้นจะเขียนด้วยภาษาโปรแกรม ภาษาโปรแกรมแต่ละภาษาจะมีลักษณะหรือรูปแบบการเขียนที่แตกต่างกัน การเลือกภาษาโปรแกรมหรือภาษาคอมพิวเตอร์เพื่อนำมาเขียนโปรแกรมนั้นขึ้นอยู่กับปัจจัยหลาย ๆ อย่าง เช่น นโยบายของบริษัท, ความเหมาะสมของโปรแกรมกับลักษณะงานที่จะถูกนำไปใช้ การเข้ากันได้กับโปรแกรมอื่น ๆ หรืออาจเป็นความถนัดของแต่ละคน ภาษาโปรแกรมที่มีแนวโน้มในการนำมาเขียนมักเป็นภาษาที่มีคนที่สามารถเขียนได้ทันที หรือหากมีความจำเป็นที่จะต้องเลือกใช้ภาษาอื่น เช่น ต้องการเน้นประสิทธิภาพในการทำงานของโปรแกรม ก็อาจจำเป็นต้องหานักเขียนโปรแกรมขึ้นมาจำนวนหนึ่งซึ่งมีความรู้ความเข้าใจในภาษาโปรแกรมที่ต้องการ และต้องมีคอมไพเลอร์ที่รองรับภาษาเหล่านั้นด้วย การเขียนโปรแกรมจะได้มาซึ่งโปรแกรมต้นฉบับหรือซอร์สโค้ดของโปรแกรมนั้น ๆ โดยปกติแล้วจะอยู่ในรูปแบบของไฟล์ข้อความธรรมดา ซึ่งไม่สามารถนำไปใช้งานได้ จะต้องผ่านการคอมไพล์ตัวซอร์สโค้ดนั้นให้เป็นภาษาเครื่อง (Machine Language) เสียก่อนจึงจะได้เป็นโปรแกรมที่พร้อมใช้งาน

ขั้นตอนที่ 4 การตรวจสอบข้อผิดพลาดของโปรแกรม (Testing and Debugging)

เป็นการตรวจสอบก่อนว่า มีข้อผิดพลาด (Error) ในโปรแกรมหรือไม่ ซึ่งอาจเกิดจากการเขียนโปรแกรมที่ผิดหลักไวยากรณ์ของภาษา (Syntax Error) โดยทั่วไปจะมีวิธีที่จะตรวจสอบข้อผิดพลาดของโปรแกรม 2 ขั้นตอน ดังนี้

- ตรวจสอบด้วยตนเอง (Self-Checking)
- ตรวจสอบด้วยการแปลภาษา (Translating)

ตรวจสอบด้วยตนเอง (Self-Checking)

ส่วนใหญ่จะเป็นการทดสอบสูตรหรือขั้นตอนการทำงานตามอัลกอริทึม โดยอาจทดลองเขียนโปรแกรมลงบนกระดาษหรือกระดาษอิเล็กทรอนิกส์ (โปรแกรมตารางงาน) แล้วกำหนดข้อมูลสำหรับตรวจสอบสูตรหรือการทำงานของโปรแกรมทีละขั้นด้วยตนเอง ว่าสูตรหรือโปรแกรมมีการทำงานที่ถูกต้อง ได้ผลลัพธ์ตรงตามความเป็นจริงหรือไม่

ตรวจสอบด้วยการแปลภาษา (Translating)

หลังจากที่เขียนโปรแกรมและมีการตรวจสอบด้วยตนเองเรียบร้อยแล้ว ก็จะป้อนโปรแกรมเข้าสู่เครื่องคอมพิวเตอร์เพื่อทำการแปลโปรแกรม โดยจะต้องเรียกใช้ตัวแปลภาษาโปรแกรมซึ่งส่วนใหญ่เป็นตัวแปลแบบคอมไพเลอร์ (Compiler) ทำการแปลภาษาโปรแกรมให้เป็นภาษาเครื่อง การแปลนี้จะเป็นการตรวจสอบความผิดพลาดทางไวยากรณ์ (Syntax Error) ของโปรแกรม ซึ่งถ้ามีข้อผิดพลาดใด ๆ โปรแกรมจะแจ้งให้ทราบทางหน้าจอ ก็ให้ทำการแก้ไขและแปลจนไม่เกิดข้อผิดพลาด ใด ๆ การแก้ไขข้อผิดพลาดทางไวยากรณ์นี้ถือว่าไม่ยากเพราะปัจจุบันโปรแกรมจะช่วยให้แก้ไขปัญหานี้ได้ง่าย

ขั้นตอนที่ 5 การทดสอบความถูกต้องของโปรแกรม (Testing and Validating)

ในบางครั้งโปรแกรมอาจผ่านการแปลโดยไม่มีข้อผิดพลาดใด ๆ แจ้งออกมา แต่เมื่อทดสอบโปรแกรมปรากฏว่าได้ผลลัพธ์ที่ไม่เป็นจริง เรียกว่าเกิดข้อผิดพลาดทางตรรกะ หรือ Logical Error แต่สิ่งที่ยากกว่า คือ เมื่อนำไปใช้งานเกิดข้อผิดพลาดขึ้นเรียกว่า Runtime Error

ขั้นตอนที่ 5 การทดสอบความถูกต้องของโปรแกรม (Testing and Validating)

ดังนั้นจึงต้องมีขั้นตอนการทดสอบความถูกต้องของโปรแกรม ซึ่งมีอยู่หลายวิธี ดังต่อไปนี้

- **การใส่ข้อมูลที่ถูกต้อง (Valid Case)** ป้องกันการเกิดปัญหา Logical Error
- **ข้อมูลเป็นไปตามข้อกำหนด** ป้องกันการเกิดปัญหา Runtime Error
- **ข้อมูลที่เป็นตัวเลขและตัวอักษร** ป้องกันการเกิดปัญหา Runtime Error
- **การใช้ขอบเขตและความถูกต้องของข้อมูลเป็นการทดสอบ** ป้องกันการเกิดปัญหา Runtime Error
- **การใช้ความสมเหตุสมผล** ป้องกันการเกิดปัญหา Runtime Error

การทดสอบโปรแกรม

นอกจากนี้ การทดสอบสามารถแบ่งได้ 2 วิธีใหญ่ ๆ คือ

1.ทดสอบด้วยข้อมูลจำนวนมาก (Empirical Testing) เป็นการทดสอบด้วยข้อมูลจริงจำนวนมาก

2.การทวนสอบอย่างมีแบบแผน (Formal Verification) ทดสอบโดยใช้หลักเกณฑ์ทางคณิตศาสตร์และตรรกศาสตร์มาสนับสนุนความถูกต้องของโปรแกรม คือทดสอบทุกกรณีที่ชุดข้อมูลสามารถเป็นไปได้

ขั้นตอนที่ 6 การทำเอกสารประกอบโปรแกรม (Documentation)

คือ การอธิบายรายละเอียดของโปรแกรมว่า จุดประสงค์ของโปรแกรมคืออะไร สามารถทำงานอะไรได้บ้าง และมีขั้นตอนการทำงานของโปรแกรมเป็นอย่างไร เครื่องมือที่ช่วยในการออกแบบโปรแกรม เช่น ผังงาน หรือรหัสจำลอง ก็สามารถนำมาประกอบกันเป็นเอกสารประกอบได้

การทำเอกสารประกอบโปรแกรม

โดยทั่วไปจะมีอยู่ด้วยกัน 2 แบบ คือ

1.เอกสารประกอบโปรแกรมสำหรับผู้ใช้ (User Documentation)

2.เอกสารประกอบโปรแกรมสำหรับผู้เขียนโปรแกรม

(Technical Documentation)

เอกสารประกอบโปรแกรมสำหรับผู้ใช้ (User Documentation)

คือ เอกสารที่บอกถึงวิธีการใช้งานโปรแกรมที่ได้ทำการเขียนขึ้น จัดทำในลักษณะคู่มือการใช้งานโปรแกรมจะมีประโยชน์สำหรับผู้ใช้งานโปรแกรม เพื่อศึกษาให้สามารถนำโปรแกรมที่เขียนขึ้นมาใช้งานได้อย่างถูกต้อง จึงเหมาะสำหรับผู้ใช้ที่ไม่ต้องเกี่ยวข้องกับการพัฒนาโปรแกรม แต่เป็นผู้ที่ใช้งานโปรแกรมอย่างเดียว โดยเน้นการอธิบายเกี่ยวกับการใช้งานโปรแกรมเป็นหลัก

- อธิบายเกี่ยวกับประสิทธิภาพ และความสามารถของโปรแกรม
- โปรแกรมนี้ทำอะไร ใช้งานในด้านไหน
- ข้อมูลเข้า มีลักษณะอย่างไร
- ข้อมูลออกหรือผลลัพธ์มีลักษณะอย่างไร
- การเรียกใช้โปรแกรม ทำอย่างไร
- คำสั่งหรือข้อมูลที่จำเป็นให้โปรแกรมเริ่มทำงาน มีอะไรบ้าง

เอกสารประกอบโปรแกรมสำหรับผู้เขียนโปรแกรม (Technical Documentation)

คือ เอกสารที่เป็นรายละเอียดของโปรแกรม ตามความต้องการของผู้ใช้งานโปรแกรม ซึ่งประกอบด้วย

- วิธีการออกแบบส่วนของมอดูลต่าง ๆ ที่มีใช้งานในโปรแกรม
- ขั้นตอนวิธี (Algorithm) ที่ใช้สำหรับนำมาเขียนโปรแกรม
- รหัสเทียม (Pseudo code) หรือ ผังงาน (Flowchart)
- ลักษณะของโครงสร้างข้อมูลที่มีการใช้งานในโปรแกรม
- โปรแกรมต้นฉบับ (Source code) ที่สมบูรณ์

เอกสารประกอบโปรแกรม

เอกสารประกอบโปรแกรมสำหรับผู้เขียนโปรแกรม สามารถแบ่งออกได้เป็น 2 ส่วน คือ

- ส่วนที่เป็นคำอธิบายหรือหมายเหตุในโปรแกรม หรือเรียกอีกอย่างหนึ่งว่า คอมเมนต์ (Comment) ซึ่งส่วนใหญ่มักจะเขียนแทรกอยู่ในโปรแกรม เพื่ออธิบายการทำงานของโปรแกรมเป็นส่วน ๆ
- ส่วนอธิบายด้านเทคนิค ซึ่งส่วนนี้จะทำเป็นเอกสารแยกต่างหากจากโปรแกรม เพื่ออธิบายในรายละเอียดที่มากขึ้น ชื่อโปรแกรมย่อยต่าง ๆ มีอะไรบ้าง แต่ละโปรแกรมย่อยทำหน้าที่อะไร และคำอธิบาย ย่อ ๆ เกี่ยวกับวัตถุประสงค์ของโปรแกรม เป็นต้น

ขั้นตอนที่ 7 การบำรุงรักษาโปรแกรม (Program Maintenance)

เมื่อโปรแกรมผ่านการตรวจสอบตามขั้นตอนเรียบร้อยแล้ว และถูกนำมาให้ผู้ใช้ได้ใช้งาน ในช่วงแรกผู้ใช้อาจจะยังไม่คุ้นเคยก็อาจทำให้เกิดปัญหาขึ้นมาบ้าง ดังนั้นจึงต้องมีผู้คอยควบคุมดูแล และตรวจสอบการทำงาน การบำรุงรักษาโปรแกรมจึงเป็นขั้นตอนที่ผู้เขียนโปรแกรมต้องคอยเฝ้าดูแลและหาข้อผิดพลาดของโปรแกรมในระหว่างที่ผู้ใช้ใช้งานโปรแกรมและปรับปรุงแก้ไขโปรแกรมเมื่อเกิดข้อผิดพลาดขึ้น หรือในการใช้งานโปรแกรมไปนาน ๆ ผู้ใช้อาจต้องการเปลี่ยนแปลงการทำงานของระบบเดิมเพื่อให้เหมาะกับเหตุการณ์ เช่น ต้องการเปลี่ยนแปลงหน้าตาของรายงาน มีการเพิ่มเติมข้อมูลหรือลบข้อมูลเดิม นักเขียนโปรแกรมก็จะต้องคอยปรับปรุง แก้ไขโปรแกรมตามความต้องการของผู้ใช้ที่เปลี่ยนแปลงไปนั้น

ขั้นตอนที่ 7 การบำรุงรักษาโปรแกรม (Program Maintenance)

เมื่อโปรแกรมใช้งานไปช่วงระยะเวลาหนึ่งอาจมีการเปลี่ยนแปลงวิธีการทำงาน หรืออีกกรณีหนึ่งเมื่อเวลาเปลี่ยนไป เทคโนโลยีมีการเปลี่ยนแปลงไป ทั้งทางด้านซอฟต์แวร์และฮาร์ดแวร์ โปรแกรมที่ใช้งานเดิมอาจจะทำงานได้เฉพาะบนเครื่องในระบบเดิม เพื่อให้การทำงานของโปรแกรมมีประสิทธิภาพมากยิ่งขึ้น หรือรองรับข้อมูลที่มีขนาดใหญ่กว่าเดิมและสามารถทำงานบนฮาร์ดแวร์ระบบใหม่ได้ จึงต้องทำการแก้ไขโปรแกรมเพิ่มเติม ดังนั้นขั้นตอนของการบำรุงรักษาโปรแกรมจึงมีความจำเป็นด้วยเหตุผลดังกล่าวข้างต้น

คุณสมบัติของนักเขียนโปรแกรมที่ดี

นักเขียนโปรแกรมหรือเรียกอีกอย่างหนึ่งว่า โปรแกรมเมอร์นั้น ควรจะมีคุณสมบัติดังต่อไปนี้ จึงจะเรียกได้ว่าเป็นโปรแกรมเมอร์ที่ดี

- รักและชอบในการเขียนโปรแกรม
- มีความคิดริเริ่มสร้างสรรค์ และใฝ่ที่จะเรียนรู้
- มีความอดทนต่อการเขียนโปรแกรม ซึ่งบางครั้งอาจต้องใช้เวลานานในการเขียนโปรแกรม
- ต้องหมั่นทำเอกสารประกอบโปรแกรมไว้ตลอด เพื่อให้ง่ายต่อการพัฒนาต่อไปในภายหลัง
- ต้องรู้จักการทำงานเป็นทีมหรือเป็นกลุ่มคณะ

ลักษณะของโปรแกรมที่ดี

โปรแกรมที่ดี จะต้องมีลักษณะ ดังนี้

- ต้องทำงานได้อย่างถูกต้อง รวดเร็วและมีประสิทธิภาพ ไม่เกิด Logic Error และมี Algorithm ที่ดี
- เมื่อนำไปใช้งานไม่ควรเกิดข้อผิดพลาดใด ๆ ไม่ควรให้เกิด Runtime Error ในขณะใช้งานไม่ว่ากรณีใด ๆ
- โปรแกรมควรมีความยืดหยุ่นสูง สามารถเข้าไปปรับแก้ไขหรือขยายความสามารถของโปรแกรมได้โดยง่าย
- สามารถอ่านแล้วมีความเข้าใจง่าย มีคำอธิบายโปรแกรมหรือคอมเมนต์ (Comment) สอดแทรกอยู่ในแต่ละมอดูล เพื่อให้ง่ายต่อการทำความเข้าใจในการทำงานของมอดูลนั้น
- ควรหลีกเลี่ยงการใช้คำสั่ง GOTO เพื่อสั่งให้โปรแกรมกระโดดไปทำงานที่จุดต่าง ๆ ขาดความเป็นระบบ ระเบียบ จะทำให้ผู้อ่านโปรแกรมเกิดความสับสนได้ง่าย

ตัวอย่างการวิเคราะห์งานและการเขียนโปรแกรม

โจทย์

ต้องการคำนวณหาพื้นที่และเส้นรอบวงของวงกลมแสดงผลออกทางจอภาพ โดยรับค่ารัศมีจากแป้นพิมพ์

1.การวิเคราะห์งาน มี 5 ขั้นตอน

- สิ่งที่ต้องการ
  - รับค่ารัศมี (Radius) ทางแป้นพิมพ์
  - คำนวณหาพื้นที่ (Area) ของวงกลม
  - คำนวณหาเส้นรอบวง (Perimeter) ของวงกลม
- การระบุข้อมูลเข้า (Input)
  - ค่ารัศมี (Radius) เป็นเลขทศนิยม โดยรับค่าจากแป้นพิมพ์ (Keyboard)

ตัวอย่างการวิเคราะห์งานและการเขียนโปรแกรม

- การระบุข้อมูลออก (Output)
  - แสดงผลค่าพื้นที่ของวงกลม (Area) เป็นเลขทศนิยม
  - แสดงเส้นรอบวงของวงกลม (Perimeter) เป็นเลขทศนิยม
- กำหนดตัวแปร
  - ค่าคงที่ PI = 3.1415926
  - ตัวแปรอินพุต คือ Radius เป็นชนิดเลขทศนิยม
  - ตัวแปรเอาต์พุต ได้แก่ Area เป็นชนิดเลขทศนิยม และ Perimeter เป็นชนิดเลขทศนิยม
- กำหนดวิธีการประมวลผล (Process) เลือกวิธีการประมวลผลโดยใช้สูตร
  - Area = PI * Radius * Radius
  - Perimeter = 2 * PI * Radius

ตัวอย่างการวิเคราะห์งานและการเขียนโปรแกรม (ต่อ)

2. การออกแบบอัลกอริทึม

2.1 **เขียนขั้นตอนวิธีการทำงาน (Algorithm)** ได้ดังนี้

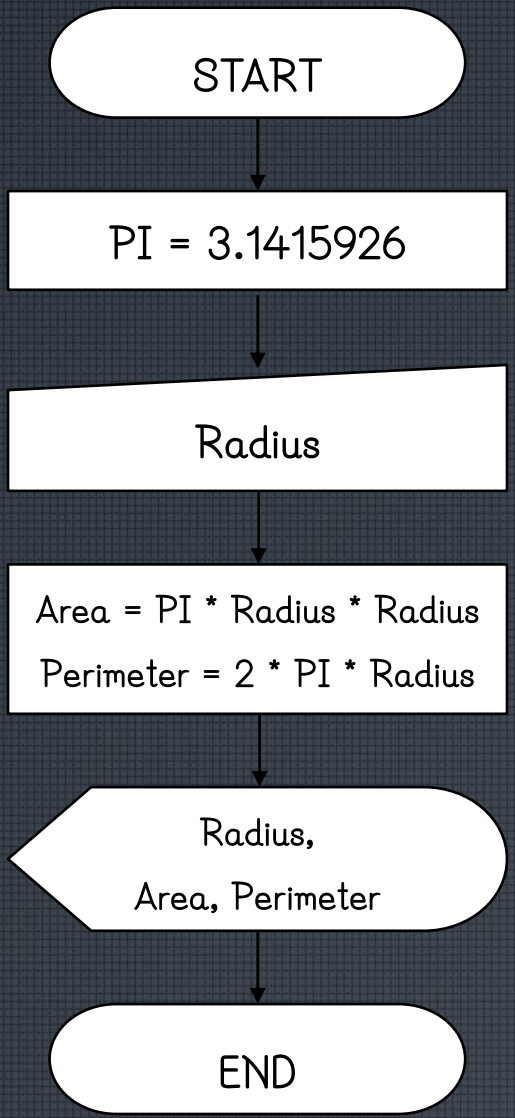
- เริ่มต้น
- กำหนดค่าคงที่ PI = 3.1415926
- รับค่ารัศมี Radius ทางแป้นพิมพ์ (โดยป้อนค่าตัวเลขเข้าไป)
- คำนวณค่าพื้นที่ของวงกลม และเส้นรอบวง จากสูตร
  - พื้นที่วงกลม Area = PI x Radius x Radius (มาจากสูตร area = πr^2)
  - เส้นรอบวง Perimeter = 2 x PI x Radius (มาจากสูตร perimeter = $2\pi r$)
- แสดงค่ารัศมี ค่าพื้นที่ และค่าเส้นรอบวงของวงกลม ออกทางจอภาพ
- จบการทำงาน

ตัวอย่างการวิเคราะห์งานและการเขียนโปรแกรม (ต่อ)

2.2 **เขียนรหัสเทียม** ได้ดังนี้

```
Algorithm Calculate Area and Perimeter of Circle
Begin
1       Set PI = 3.1415926
2       Read  Radius
3       Compute  Area = PI * Radius * Radius
4       Compute  Perimeter = 2 * PI * Radius
5       Display  Radius, Area, Perimeter
End
```

2.3 นำมาเขียนผังงาน ได้ดังนี้



การเขียนโปรแกรม

จากตัวอย่างการวิเคราะห์งานและการเขียนซูโดโค้ดของการหาพื้นที่และเส้นรอบวงของวงกลม สามารถนำมาเขียนในโปรแกรมซีได้ดังนี้

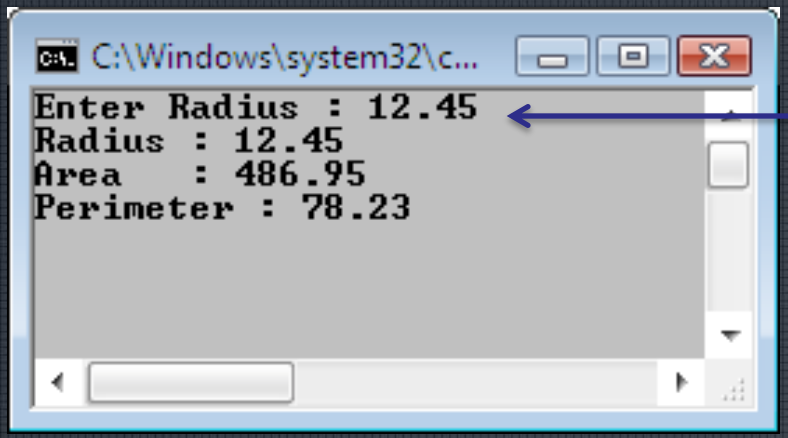
```
/* Area and Perimeter of Circle */
#include <conio.h>
#include <stdio.h>

#define PI 3.1415926   /* Constant Declaration */

void main()
{
        float radius,area,perimeter;          /* Variable Declaration */
        clrscr();
        printf("Enter Radius : "); scanf("%f",&radius);      /* Read  Radius */
        area = PI * radius * radius;          /* Area Calculate */
        perimeter = 2 * PI * radius;          /* Perimeter Calculate */
        printf("Radius : %0.2f\n",radius);
        printf("Area : %0.2f\nPerimeter : %0.2f",area,perimeter);
        getch();
}
```

ทดสอบ คอมไพล์และรันโปรแกรม

มีผลการรันดังนี้



ป้อนข้อมูลผ่านทางคีย์บอร์ด
ในที่นี้คือป้อนเลข 12.45
แล้วกด Enter

สรุป

แนวคิดการเขียนโปรแกรม ในแนวทางและวิธีการเขียนโปรแกรมคอมพิวเตอร์ คือ การเขียนโปรแกรมทุกภาษาเหมือนกัน สิ่งที่แตกต่างกันคือไวยากรณ์ภาษา

การเขียนโปรแกรมเชิงโครงสร้าง คือการกำหนดขั้นตอนให้เครื่องคอมพิวเตอร์ทำงานโดยมีโครงสร้างควบคุม มี 3 รูปแบบ คือ โครงสร้างแบบลำดับ โครงสร้างแบบเลือกกระทำตามเงื่อนไข และโครงสร้างแบบการทำซ้ำ

สรุป (ต่อ)

การเริ่มต้นเขียนโปรแกรมควรเริ่มเขียนโดยมีโครงสร้างแบบลำดับก่อน ซึ่งจะต้องมีองค์ประกอบ 4 ส่วน คือ ส่วนประกาศตัวแปร ส่วนอินพุต ส่วนประมวลผล และส่วนเอาต์พุต และควรมีการพัฒนาการเขียนโปรแกรมตามลำดับจากง่ายไปยาก จนกระทั่งสร้างโปรแกรมสมบูรณ์ขึ้นมาได้ตามความต้องการ

สรุป (ต่อ)

หลักการเขียนโปรแกรมเบื้องต้น มี 7 ขั้นตอน คือ

- การวิเคราะห์งานหรือการวิเคราะห์ปัญหา
- การออกแบบโปรแกรม
- การเขียนโปรแกรม
- การตรวจสอบข้อผิดพลาดของโปรแกรม
- การทดสอบความถูกต้องของโปรแกรม
- การทำเอกสารประกอบโปรแกรม
- การบำรุงรักษาโปรแกรม

สรุป (ต่อ)

คุณลักษณะของนักเขียนโปรแกรมที่ดี ควรมีความรักและชอบในการเขียนโปรแกรม มีความริเริ่มสร้างสรรค์และใฝ่ที่จะเรียนรู้ มีความอดทน ต้องหมั่นทำเอกสารประกอบโปรแกรมไว้ตลอด และต้องรู้จักการทำงานเป็นทีมหรือเป็นกลุ่มคณะ

ลักษณะของโปรแกรมที่ดี สามารถอ่านแล้วเข้าใจง่าย มีโครงสร้างโปรแกรมที่ดี สามารถนำมาพัฒนาปรับปรุงแก้ไขได้ง่าย มีคำอธิบายโปรแกรม และที่สำคัญคือต้องทำงานได้อย่างถูกต้อง รวดเร็วและมีประสิทธิภาพ

ควรศึกษาและค้นคว้าเพิ่มเติมกับตัวอย่างการวิเคราะห์งานและการเขียนโปรแกรม จะได้ทั้งเทคนิคและวิธีการเพื่อเป็นประสบการณ์ในการเป็นนักโปรแกรมเมอร์ที่ดีต่อไป

จบการนำเสนอ

บทที่ 2
หลักการเขียนโปรแกรม