บทที่ 1

บทนำ

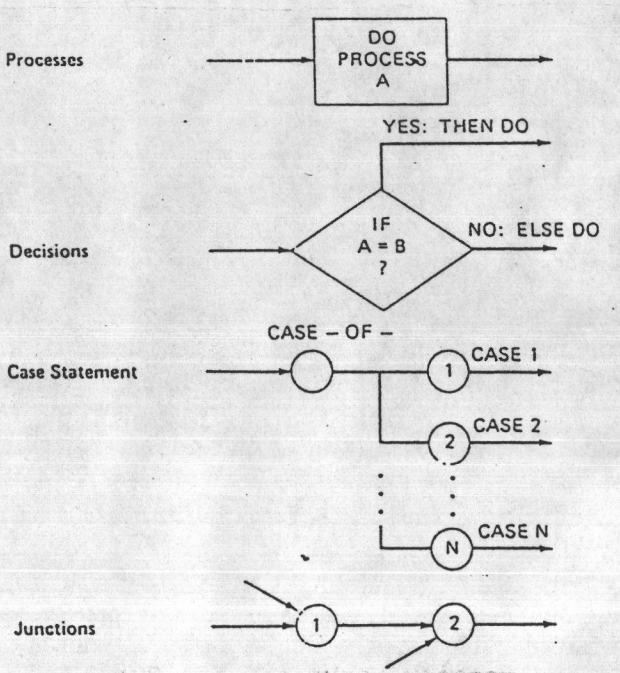
1.1 ความเป็นมาของปัญหา

ปัจจุบันเทคโนโลยีทางด้านคอมพิวเตอร์ได้ก้าวหน้าซับซ้อนขึ้นอย่างมากมาย ทั้งทางด้านฮาร์ดแวร์ และซอฟท์แวร์ ความก้าวหน้าต่างๆนี้เป็นผลให้ โปรแกรมคอมพิวเตอร์มีขนาดใหญ่มาก ซึ่งต้องใช้เวลาและค่าใช้จ่ายต่างๆ อย่างมากในการพัฒนา และบำรุงรักษาโปรแกรม ปัญหาที่สำคัญในการพัฒนาโปรแกรมคือ การหา และแก้ไขข้อผิดพลาดในระหว่างการพัฒนาโปรแกรม ซึ่งในการที่จะหาจุดบกพร่อง หรือ ข้อผิดพลาดทั้งหมดให้ได้นั้นค่อนข้างยาก โดยเฉพาะอย่างยิ่งโปรแกรมที่มีขนาดใหญ่

วิธีการทั่วไปที่ใช้กันมากในการทดสอบข้อผิดพลาดของโปรแกรมก็คือ การสร้างข้อมูลขึ้นมาเพื่อทดสอบกับโปรแกรม แล้วเปรียบเทียบผลลัพธ์ที่ได้กับผลลัพธ์ที่ถูกต้อง แต่การทดสอบแบบนี้อาจจะมีบางคำสั่ง หรือบางโมดุล ของโปรแกรมที่ไม่ได้ถูก ประมวลผล (execute)เลย ซึ่งอาจจะเกิดจากข้อมูลที่ใช้ทดสอบนั้นไม่ทั่วถึง หรืออาจเป็นเพราะโปรแกรมนั้น ไม่มีทางที่จะประมวลผลคำสั่ง หรือโมดุล นั้นได้เลย ซึ่งอาจจะทำให้โปรแกรมด้อยประสิทธิภาพ หรือมีข้อผิดพลาดเกิดขึ้นได้

1.2 แนวคิดและทฤษฎีสำคัญ

1.2.1 กราฟการควบคุมสายงาน (Control Flowgraphs) [1],[2] : กราฟรูปภาพ ที่ใช้แสดงลักษณะโครงสร้างการทำงานของโปรแกรม รูปแบบที่ใช้ดังรูปที่ 1.1



รูปที่ 1.1 แสดงรูปของโฟลว์กราฟ

1.2.1.1 **ชุดกระบวนการ** (Process Block) [1],[2] : เป็นชุดคำสั่งที่เรียงลำดับกัน ซึ่งถ้ามีคำสั่งใดคำสั่งหนึ่งถูกประมวลผล แล้ว ทุกๆ คำสั่งในชุดกระบวนการก็จะถูกประมวลผล ทั้งหมดด้วย จะมีลักษณะเป็นมีทางเข้าทางเดียว และทางออกทางเดียว

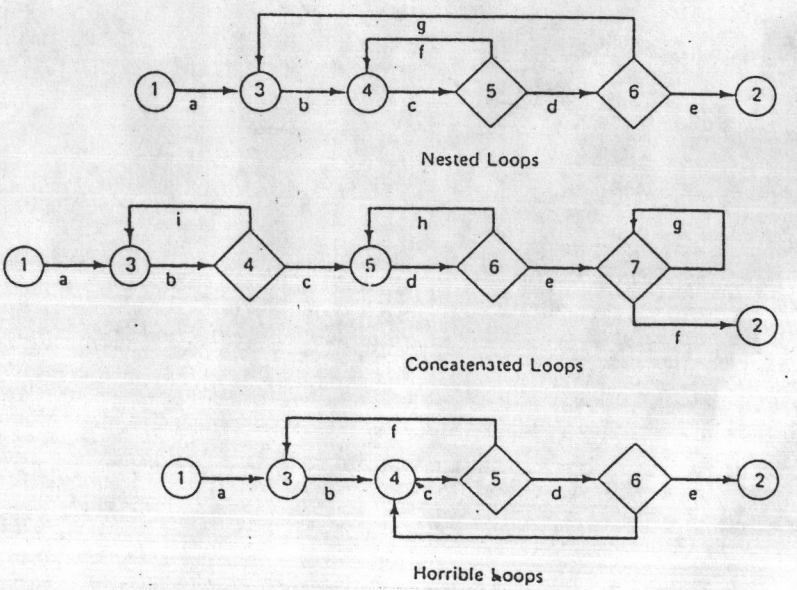
1.2.1.2 **การตัดสินใจ** (Decision) [1],[2] : เป็นจุดเปลี่ยนการควบคุมในโปรแกรม ซึ่งอาจเป็นการกระโดดไปตามเงื่อนไขที่ต้องการ หรืออาจข้ามบางคำสั่งตามเงื่อนไข โดยทั่วไปแล้วจะมีลักษณะทางเลือกสองทาง (binary) คือ จริงและเท็จ

1.2.1.3 **คำสั่งเงื่อนไข** (Case Statement) [1],[2] : เป็นลักษณะที่มีการตัดสินใจทางเลือกหลายทางขึ้นอยู่กับตัวเงื่อนไขที่กำหนด โดยทั่วไปแล้วจะมีทางเข้าได้ทางเดียว และมีทางออกได้หลายทาง

1.2.1.4 **จุดร่วม** (Junction) [1],[2] : เป็นจุดที่การควบคุมการทำงานรวมเข้าหากัน ซึ่งอาจเกิดจากการใช้คำสั่งที่ควบคุมการทำงานโดยตรงเช่นคำสั่ง GO TO เป็นต้น

1.2.1.5 **กราฟการควบคุมสายงาน กับ ผังงาน** (Flowchart) [1] : ทั้งสองจะมีลักษณะคล้ายกันมาก จะแตกต่างกันเพียง ในกราฟการควบคุมสายงานจะไม่แสดงรายละเอียดของคำสั่งที่อยู่ในชุดกระบวนการ นั่นคือ ในชุดกระบวนการ เราจะไม่บอกว่ามีกี่คำสั่งที่อยู่ภายในแต่จะบอกเป็นเพียง 1 ชุดกระบวนการเท่านั้น ส่วนในผังงานนั้น ทุกๆคำสั่งใน ชุดกระบวนการจะถูกแสดงทั้งหมด สมมติเช่น ถ้าเรากำหนดว่า 1 รูป (box) ต่อ 1 คำสั่ง แล้วถ้าในชุดกระบวนการ เรามี 100 คำสั่ง ก็จะมี 100 รูปในผังงาน แต่ในกราฟการควบคุมสายงานจะมี 1 ชุดกระบวนการเพียงรูปเดียวเท่านั้น

1.2.2 **วงวน** (Loops) [1],[2] : เราแบ่งวงวนได้เป็น 3 ลักษณะคือ วงวนซ้อน (nested loops), วงวนต่อกัน (concatenated loops), และ วงวนคาบเกี่ยว (horrible loops) ดังรูปที่ 1.2



รูปที่ 1.2 แสดงลักษณะของ วงวน ชนิดต่างๆ

1.2.3 เส้นทาง (Path) [1],[2] : คือ ลำดับของคำสั่งที่เริ่มจาก ตำแหน่งเข้า (entry), จุดร่วม หรือ จุดตัดสินใจ และจบลงที่ จุดร่วม, จุดตัดสินใจ หรือ ทางออก (exit) ในเส้นทางสามารถมี จุดร่วม, กระบวนการ หรือ จุดตัดสินใจ ได้หลายๆ ครั้ง

1.2.3.1 เชื่อมโยง (link) [1],[2] : คือกระบวนการ (process) ที่เชื่อมต่อระหว่างจุดแตกกิ่ง (node) สองจุด เช่น จุดร่วม-กระบวนการ-จุดร่วม (junction-process-junction), จุดร่วม-กระบวนการ-จุดตัดสินใจ (junction-process-decision)

1.2.3.2 เส้นทางบริบูรณ์ (entry/exit path, complete path) [1]: หมายถึงเส้นทาง ที่เริ่มต้นจากตำแหน่งเข้า และจบลงที่จุดทางออก

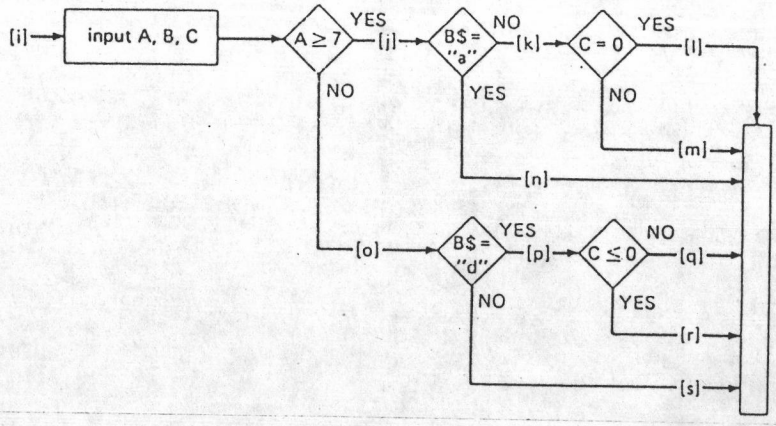
1.2.3.3 เพรดิเคต (Predicate) [1],[2] : หมายถึง ตัวเงื่อนไขที่ใช้ในการตัดสินใจ เช่น "X > 0", "X < (Y -5)" เป็นต้น

1.2.3.4 เส้นทางเพรดิเคต (Path predicate) [1],[2]:หมายถึงตัวเพรดิเคตทั้งหมดที่อยู่ในเส้นทางนั้น

1.2.3.5 เส้นทางตัดสินใจ (Decision-to-Decision path) [1],[2]: ได้แก่ เส้นทาง ที่เริ่มต้นจากจุดตำแหน่งเข้า หรือ บล็อกตัดสินใจ (decision block) และจบลงที่จุดทางออก หรือบล็อกตัดสินใจ และไม่มีบล็อกตัดสินใจอยู่ภายในเส้นทางนั้น

1.2.3.6 การตรวจสอบเส้นทาง (Path Instrumentation) [1],[2] : เป็นการตรวจสอบการทำงานของคำสั่ง หรือของเส้นทางที่เราต้องการทราบว่าได้ถูกประมวลผลแล้วหรือไม่ เช่นเราอาจจะแทรกคำสั่งที่ให้พิมพ์ ข้อความใดข้อความหนึ่งโดยแทรกไว้ยังหลังคำสั่งที่เราต้องการ แล้วตรวจสอบดูผลลัพธ์ที่ได้ว่ามีข้อความที่เราแทรกไว้หรือไม่ โดยทั่วไปลักษณะการแทรกตัวตรวจสอบนี้ จะแทรกไว้หลังคำสั่งที่เป็นตัวควบคุมสายงาน (control flow) เช่น คำสั่งเงื่อนไข, คำสั่งวนซ้ำ เป็นต้น

1.2.3.6.1 ตัวตรวจสอบเชื่อมโยง (Link Markers) [1],[2] : เป็นการตรวจสอบการทำงานของแต่ละเชื่อมโยง ที่อยู่ใน กราฟสายงาน (flowgraph) โดยแทรกตัวตรวจสอบไว้ที่ต้นของแต่ละเชื่อมโยง ดังตัวอย่างในรูปที่ 1.3 บางครั้งการใส่ตัวตรวจสอบไว้ที่ต้นของเชื่อมโยง เพียงตัวเดียวอาจตรวจสอบการทำงานของเชื่อมโยงไม่เพียงพอเนื่องจากการควบคุมการทำงานของสายงาน อาจจะหลุดในช่วงกลางของเชื่อมโยงก็ได้ อย่างไรก็ตามมีเทคนิคอย่างหนึ่งในการตรวจสอบ เชื่อมโยง โดยการแทรกตัวตรวจสอบสองตัวไว้ที่ต้น และปลายของเชื่อมโยง แล้วนำตัวตรวจสอบทั้งสองตัวมาเปรียบเทียบกัน ถ้าผลที่ได้ไม่เหมือนกัน ก็แสดงว่าอาจมีข้อผิดพลาดเกินขึ้นในส่วนของเชื่อมโยง นั้น



รูปที่ 1.3 การตรวจสอบการเชื่อมโยง

1.2.3.6.2 **ตัวนับเชื่อมโยง (Link Counters)** [1],[2] : เป็นการตรวจสอบการทำงานของเชื่อมโยงอย่างหนึ่งโดยแทรกตัวนับ (counter) ไว้ที่ต้น และปลายของเชื่อมโยง โดยตัวนับจะถูกบวกทีละหนึ่งเมื่อมีการประมวลผลตัวนับ การตรวจสอบที่สำคัญของการใส่ตัวนับก็คือการตรวจสอบค่าของตัวนับที่ต้นและปลายของเชื่อมโยง มีค่าเท่ากันหรือไม่, การตรวจสอบค่าของผลรวมของตัวนับทั้งหมดที่ต้นของเชื่อมโยง เท่ากับค่าผลรวมทั้งหมดของตัวนับที่ปลายของเชื่อมโยงหรือไม่, หรือการตรวจสอบค่าตัวนับว่าเป็นศูนย์ซึ่งเป็นค่าเริ่มต้นของตัวนับหรือไม่ ถ้าเป็นศูนย์ก็แสดงว่าเชื่อมโยง นั้นยังไม่เคยถูกประมวลผลเลย

1.3 วัตถุประสงค์ของการวิจัย

ในการวิจัยนี้ จะพัฒนาเครื่องมือที่ใช้ทดสอบโปรแกรมภาษาซี เพื่อใช้ตรวจสอบคำสั่งและโมดุล ทั้งหมดของโปรแกรม โดยการแทรกตัวตรวจสอบลงในรหัสต้นฉบับ (source code) ภาษาซี โดยเฉพาะในส่วนที่เป็นเส้นทางการตัดสินใจ (Decision-to-Decision path : DD-path) และตรวจดูถึงการเรียกใช้งานในแต่ละคำสั่ง หรือแต่ละโมดุล ในขณะที่ทดสอบโปรแกรมด้วยข้อมูล แล้วจัดทำเป็นสถิติเพื่อบอกถึงความถี่ของการใช้งานเพื่อให้ผู้พัฒนาโปรแกรมทราบว่ามีคำสั่ง หรือมีโมดุล ใดบ้างที่ยังไม่ถูกประมวลผล หรือมีคำสั่ง หรือโมดุล ใดบ้างที่ถูกเรียกใช้บ่อย เพื่อนำไปปรับปรุงโปรแกรมให้มีประสิทธิภาพมากยิ่งขึ้น หรือตัดโมดุลที่ไม่จำเป็นออก และจะได้แน่ใจว่าทุกๆส่วนของโปรแกรมได้ถูกประมวลผล หรือทดสอบกับข้อมูลแล้ว

1.4 ขอบเขตของการวิจัย

1.4.1 การวิจัยนี้พัฒนาโดยใช้ภาษาซีของตัวแปลชุดคำสั่ง เทอร์โบซี รุ่น 2.0 บนเครื่องไมโครคอมพิวเตอร์ IBM PC XT/AT หรือเครื่องใช้แทนกันได้ (compatable) ภายใต้ระบบการปฏิบัติการดอส รุ่น 3.3

1.4.2 การวิจัยนี้เพื่อเป็นเครื่องมือในการทดสอบโปรแกรมภาษาซี ที่พัฒนาโดยใช้ตัวแปลชุดคำสั่ง เทอร์โบซี รุ่น 2.0 ไม่ใช่การตรวจสอบไวยากรณ์ (syntax) ของการเขียนโปรแกรมภาษาซี โปรแกรมที่ใช้ทดสอบกับเครื่องมือนี้ต้องมีไวยากรณ์ถูกต้องแล้ว

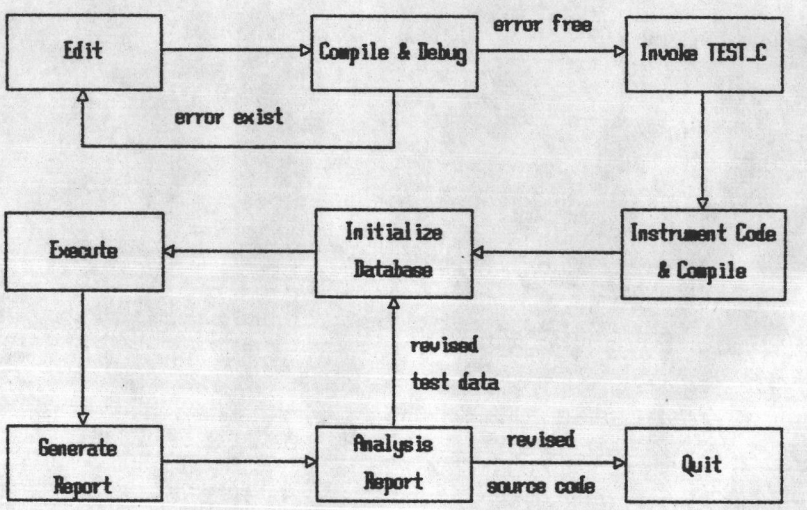
1.4.3 การวิจัยนี้เพื่อเป็นการตรวจสอบดูว่ามีคำสั่งใด หรือโมดุลใดบ้างในโปรแกรมที่ยังไม่ได้ถูกประมวลผลเลย หรือมีการประมวลผลด้วยความถี่มากน้อยเท่าไหร่ ไม่ใช่เป็นการตรวจสอบทางตรรกะ (logic) ของโปรแกรมภาษาซี

1.4.4 การวิจัยนี้จะไม่ทำการสร้างข้อมูลที่ใช้สำหรับการทดสอบกับโปรแกรมให้ แต่ผู้ใช้จะต้องมีข้อมูลที่ใช้สำหรับทดสอบ และผลลัพธ์ที่ถูกต้องอยู่แล้ว ซึ่งในการวิจัยนี้จะบอกเพียงว่าในขณะที่ทดสอบโปรแกรมด้วยข้อมูลนั้นมีคำสั่งใด หรือโมดุลใดบ้างในโปรแกรมที่ยังไม่ได้ถูกประมวลผล ซึ่งอาจเกิดจากข้อมูลที่ใช้ทดสอบไม่ทั่วถึง หรืออาจเป็นเพราะตรรกะของโปรแกรมไม่มีทางที่จะประมวลผลได้เลย

1.5 ขั้นตอนการดำเนินการวิจัย

1.5.1 ศึกษาทฤษฎีเกี่ยวกับวิธีการทดสอบโปรแกรม

1.5.2 ออกแบบระบบ (System Design) : ออกแบบระบบของเครื่องมือที่จะพัฒนา ซึ่งได้แก่ การออกแบบโครงสร้างระบบ, การออกแบบโครงสร้างข้อมูล, การออกแบบโครงสร้างโมดุล ดูแผนภาพ (diagram) รูปที่ 1.4



รูปที่ 1.4 แผนภาพสายงาน (Control Flow Diagram)

หอสมุดกลาง สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

1.5.3 ออกแบบรายละเอียดระบบ (Detail Design) : ออกแบบรายละเอียดทั้งหมดของระบบได้แก่ การออกแบบรายละเอียดข้อมูล, การออกแบบรายละเอียดโมดุล, การออกแบบระบบติดต่อกับผู้ใช้ (User interface), การออกแบบรายละเอียดจอภาพ, การออกแบบผลลัพธ์และรายงาน

1.5.4 พัฒนาโปรแกรมและทดสอบ : พัฒนาเครื่องมือทดสอบภาษาซีโดย ใช้ตัวแปลชุดคำสั่งของเทอร์โบซี รุ่น 2.0 และทดสอบกับข้อมูลซึ่งเป็นโปรแกรมภาษาซีของตัวแปลชุดคำสั่งเทอร์โบซี รุ่น 2.0 ซึ่งมีขนาดโปรแกรมประมาณไม่เกิน 2,000 บรรทัด

1.5.5 สรุปผลการวิจัย และข้อเสนอแนะ

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1.6.1 ผู้ใช้จะได้ระวังถึง คำสั่ง หรือโมดุล ที่ยังไม่ได้ทดสอบในโปรแกรม

1.6.2 ช่วยในการพัฒนาโปรแกรมในช่วงของการทดสอบข้อมูล เพื่อให้แน่ใจว่าทุกๆ คำสั่งในโปรแกรมได้ผ่านการประมวลผลมาแล้วอย่างน้อยหนึ่งครั้ง

1.6.3 จะได้ทราบถึงคำสั่ง หรือโมดุลใดบ้างที่ถูกเรียกใช้งานมาก หรือคำสั่งใดบ้างที่แทบไม่ได้ถูกเรียกใช้เลย เพื่อช่วยในการปรับปรุงโปรแกรมให้มีประสิทธิภาพยิ่งขึ้น