ภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้น

Computer Language for Beginner

นาย คณิศร์ ชาญทวีคุณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

ภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้น

Computer Language for Beginner

โดย

นาย คณิศร์ ชาญทวีคุณ

อาจารย์ที่ปรึกษา

ผศ. สมเกียรติ วังศิริพิทักษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้น

Computer Language for Beginner

ผู้จัดทำ

นายคณิศร์ ชาญทวีคุณ รหัสนักศึกษา 46010081

อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ สมเกียรติ วังศิริพิทักษ์)

ภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้น

นาย คณิศร์ ชาญทวีคุณ 46010081
ผศ. สมเกียรติ วังศิริพิทักษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2549

บทคัดย่อ

ปัจจุบันนี้การเขียนโปรแกรมคอมพิวเตอร์กลายเป็นปัจจัยพื้นฐานในหลายๆ ด้าน เช่น ด้านการศึกษา ด้านวิศวกรรม ด้านเทคโนโลยีสารสนเทศ ฯลฯ โดยการเขียนโปรแกรมแบบที่เคยมีมานั้น ผู้ที่จะเริ่มต้นศึกษาการเขียนโปรแกรมคอมพิวเตอร์จะต้องเผชิญกับความซับซ้อนของภาษาคอมพิวเตอร์ เพราะว่าในแต่ละภาษาจะมีรูปแบบคำสั่งที่แตกต่างกัน ทำให้ผู้เริ่มต้นเหล่านั้นมุ่งแต่ที่จะทำความเข้าใจรูปแบบ (syntax) ของภาษาแต่ละภาษา จนกระทั่งลืมให้ความสำคัญกับกระบวนการคิดวิเคราะห์หาวิธีการ (algorithm) แก้ปัญหา ซึ่งถือเป็นหัวใจของการเขียนโปรแกรมคอมพิวเตอร์ โดยทั่วไปผู้เริ่มต้นศึกษาการเขียนโปรแกรมจนถึงผู้พัฒนาโปรแกรมทั่วไปมีการนำ Flowchart มาใช้ในการอธิบายความคิดออกมาให้เป็นรูปธรรม หลังจากนั้นจึงแปลง Flowchart เป็น Source code ภาษาต่างๆ ตามที่ผู้พัฒนาใช้ ผลลัพธ์ที่ได้จึงเป็นโปรแกรม EXE ที่สามารถใช้งานได้ทั่วไป "ภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้น (Computer Language for Beginner)" ได้นำสองขั้นตอนของการพัฒนาโปรแกรมรวมเหลือเพียงขั้นตอนเดียว กล่าวคือผู้พัฒนาโปรแกรมเพียงถ่ายทอดความคิดขั้นตอนการแก้ปัญหามาเป็น Flowchart จากนั้น Visual Programming Environment จะทำการแปลงให้เป็นโปรแกรม EXE นอกจาก "ภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้น" จะเป็นเครื่องมือที่ใช้ในการพัฒนาโปรแกรมคอมพิวเตอร์แล้ว ยังสามารถใช้เป็นเครื่องมือที่ใช้ในการสอนเรื่องหลักการเขียนโปรแกรมที่เข้าใจง่ายและมีประสิทธิภาพได้อีกด้วย

Computer Language for Beginner

Mr. Kanis Charntawcekhun 46010081

Asst.Prof. Somkiat Wangsiripitak Advisor

Academic Year 2006

ABSTRACT

Nowadays a computer programming become the fundamental knowledge for people studying or working in a computer field, e.g. computer engineering, computer science, information technology, etc. Programmer not only considers the algorithm carefully but also performs the coding according to the syntax of programming language used. Therefore writing a computer program is a difficult task, especially for a beginner. In fact, a novice often uses a flowchart as a tool to help him summarize the idea into all necessary steps of solution, but this flowchart cannot be understood by the computer. It must be converted to the program by using some programming language and then be compiled and run. The visual programming using flowchart proposed in this paper allows the programmer to write the program in the format of flowchart, to compile and to run, without the coding step. There is no necessity to remember the syntax of any programming languages. It takes the benefits of easy-to-understand and easy-to-perform of the flowchart, whereas eliminates the weakness of a conventional programming, e.g. the requirement of remembering the syntax, the error in coding step. Furthermore, a debugging of program by the proposed system is straightforward and easy to discover the error. The proposed system could be used as a tool for teaching the basic concept of structural programming as well. Experimental results show the powerfulness, easiness, and user friendliness of the proposed system.

กิตติกรรมประกาศ

ปริญญานิพนธ์เล่มนี้และโปรแกรมภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้นสำเร็จลุล่วงได้ด้วยดี ก็เนื่องมาจากการให้โอกาส การดูแล ให้คำแนะนำต่างๆ การสนับสนุน การให้คำสั่งสอนและให้คำปรึกษาเป็นอย่างดีเสมอมาจาก ท่านผู้ช่วยศาสตราจารย์สมเกียรติ วังศิริพิทักษ์ ซึ่งต้องขอบพระคุณท่านอาจารย์เป็นอย่างสูง ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ และสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ได้จัดเตรียมสิ่งอำนวยความสะดวก เพื่อให้งานวิจัยดำเนินไปได้อย่างสะดวกและรวดเร็ว

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณบิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกๆเรื่อง

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นาย คณิศร์ ชาญทวีคุณ

สารบัญ

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1. ความสำคัญและที่มาของโครงงาน

เนื่องด้วยคอมพิวเตอร์กลายเป็นสิ่งที่ขาดไม่ได้ในหลายๆ ด้าน เราสามารถพบเห็นคอมพิวเตอร์ได้ในทุกที่ตั้งแต่ตื่นจนถึงกระทั่งนอนหลับ เช่น ในด้านธุรกิจมีการนำคอมพิวเตอร์เข้ามาใช้ในระบบเครื่องทอนเงินอัตโนมัติ การติดต่อสื่อสารก็ทำได้สะดวกขึ้นผ่านช่องทางอินเตอร์เน็ต แม้กระทั้งในเครื่องซักผ้ายังนำระบบคอมพิวเตอร์มาช่วยให้การทำความสะอาดเสื้อผ้าสะดวกสบายยิ่งขึ้น ซึ่งประชาชนเกือบทั้งหมดใช้คอมพิวเตอร์ในฐานะของผู้ใช้ มีเพียงกลุ่มคนกลุ่มเล็กๆ ที่พยายามที่จะเป็นนักพัฒนาที่จะสรรค์สร้างนวัตกรรมใหม่ๆ อย่างไรก็ตามมีบางคนยอมทิ้งความฝันที่จะเป็นนักพัฒนาของตัวเองลงเพียงเพราะไม่เข้าใจถึงหลักการเขียนโปรแกรม

ท่ามกลางภาษาคอมพิวเตอร์ที่มีมากกว่า 50 ภาษาในโลก บางภาษาเป็นภาษาเฉพาะทาง บางภาษาก็เป็นภาษาสำหรับการพัฒนาโปรแกรมทั่วไป นักพัฒนาส่วนใหญ่ให้ความสำคัญส่วนใหญ่ที่รูปแบบคำสั่งของภาษาต่างๆ ซึ่งมีความเจาะจงและไม่ยืดหยุ่น เช่น ในภาษา JAVA การใช้ตัวอักษรตัวเล็กตัวใหญ่มีผลไม่เหมือนกัน หรือในภาษา Python ที่ใช้การจัดย่อหน้าเพื่อแสดงขอบเขตของโปรแกรม ภาษา C ต้องมีเครื่องหมาย " ; " ปิดท้ายทุกคำสั่ง ซึ่งจริงๆ แล้วภาษาคอมพิวเตอร์เป็นเพียงส่วนหนึ่งของการพัฒนาโปรแกรม แต่ส่วนที่สำคัญที่สุดคือกระบวนการแก้ปัญหา (Algorithm) ในการพัฒนาโปรแกรมโดยการพิมพ์คำสั่ง หลายครั้งที่นักพัฒนาจะต้องประสบกับการคอมไพล์ (compile) ไม่ผ่าน สาเหตุมาจากรูปแบบคำสั่งของโปรแกรมไม่ถูกต้อง หลายครั้งที่โปรแกรมสามารถทำงานได้แต่ไม่ตรงตามวัตถุประสงค์ นักพัฒนาจึงต้องกลับไปหาจุดผิดใน Source Code ซึ่งมีความยากมากที่จะหาพบ นอกจากนี้ยังใช้เวลานานมากอีกด้วย จนกระทั่งหลายๆ คนตัดสินใจลบโปรแกรมเดิมแล้วเริ่มพัฒนาโปรแกรมขึ้นมาใหม่ การใช้ Flowchart มาช่วยในการคิดหาวิธีการทำงานของโปรแกรม เป็นวิธีที่ดีและมีประสิทธิภาพมากในการที่จะอธิบายความคิดออกมาเป็น Algorithm ผู้พัฒนาบางคนจึงนำ Flowchart มาใช้ในการวางแผนการเขียนโปรแกรม เพราะสามารถใช้อธิบายการทำงานที่ซับซ้อนให้เข้าใจได้ง่ายด้วย Flowchart ดังนั้น Flowchart จึงกลายเป็นเครื่องมือที่ใช้ในการแก้ไขโปรแกรมหรือใช้ในการศึกษาการเขียนโปรแกรมของผู้เริ่มต้น แต่ Flowchart ที่ได้ไม่สามารถนำไปแปลงเป็นโปรแกรม EXE ได้โดยตรง ต้องนำไปเขียนเป็น source code ก่อน ทำให้ไม่สะดวกและถือเป็นจุดด้อย

จากการศึกษากรณีการสอนภาษาคอมพิวเตอร์ให้กับผู้ใหญ่ พบว่า 20% ต้องการที่จะใช้ภาษาคอมพิวเตอร์จริงๆ ในการเข้าใจการเขียนโปรแกรม 40 % ต้องการใช้ Flowchart และอีก 40% ต้องการใช้ pseudo code ผู้ที่ได้รับการอบรม 100 % ต้องการที่จะเห็นผลลัพธ์ของการเขียนโปรแกรมเป็นโปรแกรม EXE ที่พวกเขาสามารถทดลองใช้งานได้ และจากการทดลองพบอีกว่า

การเขียนโปรแกรมแบบมองเห็นภาพ (Visual Programming) เช่น Lab View (Laboratory Virtual Instrumentation Engineering Workbench) เหมาะกับผู้เริ่มต้นมากกว่าการพิมพ์คำสั่ง ถึงแม้ภาษาคอมพิวเตอร์แบบมองเห็นภาพจะมีอยู่แล้วในปัจจุบัน แต่ส่วนใหญ่เป็นภาษาเฉพาะทาง ผู้พัฒนาจึงตัดสินใจพัฒนา "ภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้น (Visual Programming using Flowchart)" เพื่อให้เป็นภาษาที่ใช้ในการแก้ปัญหาทั่วๆไป

1.2. วัตถุประสงค์ของโครงงาน

1.2.1. เพื่อสร้างสภาพแวดล้อมใหม่สำหรับการพัฒนาโปรแกรมทั่วไป

1.2.2. เพื่อพัฒนาภาษาคอมพิวเตอร์ใหม่ที่ง่ายต่อความเข้าใจ

1.2.3. เพื่อพัฒนาเครื่องมือที่ง่ายต่อการศึกษาการเขียนโปรแกรม

1.2.4. เพื่อเปลี่ยนทัศนคติในแง่ลบต่อการเขียนโปรแกรม

1.3. ประโยชน์ที่คาดว่าจะได้รับ

1.3.1. พัฒนารูปแบบการเขียนโปรแกรมแบบใหม่ที่มีประสิทธิภาพ

1.3.2. พัฒนาเครื่องมือที่ช่วยในการเข้าใจการเขียนโปรแกรมแบบโครงสร้าง

1.3.3. สามารถนำไปใช้ในการพัฒนาโปรแกรมได้จริง

1.3.4. สามารถนำไปใช้เป็นสื่อประกอบการสอนวิชาเบื้องต้นที่เกี่ยวกับการเขียนโปรแกรม

1.4. ขอบเขตของโครงงาน

1.4.1. Visual Programming using Flowchart เป็นเครื่องมือที่ใช้พัฒนาโปรแกรมคอมพิวเตอร์แบบ Structural Programming

1.4.2. Visual Programming using Flowchart Environment สามารถเขียนโปรแกรมโดยการวาด Flowchart แล้วให้ระบบช่วยแปลง flowchart ให้ได้ผลลัพธ์เป็นโปรแกรม EXE

1.4.3. Visual Programming using Flowchart ทำงานภายใต้ JVM

1.4.4. Visual Programming using Flowchart มีหน้าที่ในการแปลง Flowchart เป็น Source code ภาษา C แล้วให้ C Compiler แปลงเป็นโปรแกรม EXE

1.4.5. มีเครื่องมือในการ debugger

1.4.6. สามารถพิมพ์ flowchart ออกทางเครื่องพิมพ์ได้

1.4.7. สามารถเรียกใช้งาน flowchart ที่มีอยู่ให้ทำงานในลักษณะของการเรียก function ได้

1.4.8. สามารถเก็บข้อมูลของ flowchart ในรูปแบบของ vector

1.4.9. คำสั่งที่ใช้สามารถใช้ได้ทั้งคำสั่ง ภาษา C และคำสั่งเฉพาะของ Visual Programming using Flowchart Environment

1.4.10. รองรับชนิดข้อมูล 3 ชนิดคือ ตัวอักษร จำนวนเต็ม และเลขทศนิยม

1.5. ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บทด้วยกัน คือ

บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของวิทยานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีที่เกี่ยวข้อง ประกอบด้วย วิธีการเขียนโปรแกรมแบบต่างๆ ภาษาและเครื่องมือในการเขียน Visual Programming ที่มีอยู่ในปัจจุบัน หลักการเขียน flowchart การเก็บรูปภาพแบบ Bitmap และ Vector และการติดต่อระหว่าง process

บทที่ 3 กล่าวถึงการออกแบบโปรแกรม โดยจะแบ่งเป็นส่วนย่อยๆ 3 ส่วน คือ การออกแบบโปรแกรมส่วนการวาด flowchart การออกแบบโปรแกรมส่วนการแปลง flowchart และส่วนการ run และ debug โปรแกรม

บทที่ 4 กล่าวถึงการทดลองและผลการทดลอง โดยทำการวาด flowchart แบบต่างๆ แล้วแปลงเป็น source code จากนั้นทดลอง run โปรแกรม debug โปรแกรม และวิเคราะห์ผลการทดลอง

บทที่ 5 กล่าวถึงข้อสรุปของโครงงาน รวมทั้งปัญหาและอุปสรรค์ที่พบ แนวทางการพัฒนาต่อและข้อเสนอแนะ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1. วิธีการเขียนโปรแกรมแบบต่างๆ

การเขียนโปรแกรมนั้นเป็นศาสตร์ที่ต้องใช้ทักษะและการฝึกฝนอย่างต่อเนื่อง ถึงจะสามารถเขียนโปรแกรมได้อย่างมีประสิทธิภาพ ผู้ที่เขียนโปรแกรม จะต้องมีกระบวนการคิดแก้ปัญหาอย่างเป็นแบบแผนและเป็นระบบด้วย และในปัจจุบันนี้เองคนที่จะเป็นโปรแกรมเมอร์นั้นไม่จำเป็นต้องเรียนจบในสาขาคอมพิวเตอร์เสมอไป คนที่เรียนจบสาขาอื่น ๆ เช่น บัญชี สถิติ วิศวกรรม แพทย์ ก็สามารถเขียนโปรแกรมได้ ซึ่งบุคคลเหล่านี้ก็อาศัยการฝึกฝนทั้งนั้น รูปแบบของการเขียนโปรแกรมของโปรแกรมเมอร์แต่ละคนก็ขึ้นอยู่กับความถนัดในการแก้ปัญหาของแต่ละคนออกไป บางครั้งการที่ต่างคนต่างเรียนรู้และเริ่มเขียนโปรแกรมที่ตนเองต้องการขึ้นมานั้น อาจจะทำให้ไม่มีรูปแบบการเขียนที่เป็นมาตรฐานและเป็นที่ยอมรับ โดยแท้จริงแล้วการเขียนโปรแกรมในหลักของวิศวกรรมซอร์ฟแวร์แล้วก็จะแบ่งประเภทได้เป็น 3 ลักษณะ คือ Structural Programming, Object-Oriented Programming และ Generic Programming

2.1.1. การเขียนโปรแกรมแบบ Structural Programming

การเขียนโปรแกรมแบบ Structural Programming หรือเรียกอีกชื่อหนึ่งว่า Procedural Programming ภาษาที่ใช้ส่วนใหญ่จะเป็นภาษาระดับสูง มีประโยชน์ตรงที่เขียนครั้งเดียวแล้วนำไป Compile ใน Platform ไหนก็ได้ ตัวอย่างเช่น เขียนโค้ดโปรแกรมบนซีพียูของ Digital ถ้าต้องการนำมาใช้บนซีพียูของ Intel เราก็เพียงนำ Source Code โปรแกรมมา compile อีกครั้งบนซีพียูของ Intel ประโยชน์นี้เป็นเหตุผลหลักในการสร้างภาษาระดับสูงขึ้นมา มิฉะนั้นแล้วเราจะต้องเขียนโค้ดหลาย ๆ ครั้ง ถ้าใช้ภาษาระดับล่าง หรือภาษาเครื่อง

การเขียนโปรแกรมแบบ Structural Programming จะเป็นการเขียนโปรแกรมจากบนลงมาล่าง (Top-Down) เป็นการแก้ปัญหาโดยการมองจากบนลงมาล่างที่เป็นลำดับขั้นตอน หรือตาม Flowchart ซึ่งทำให้ผู้เขียนสามารถแบ่งโปรแกรมเป็นฟังก์ชัน (Function) ย่อย ๆ หรือเรียกว่า Procedure หรือ Subroutine การเขียนโปรแกรมแบบนี้จะเป็นการเขียนตามหน้าที่การทำงาน เมื่อต้องการคำนวณบางอย่าง จะต้องเขียน function เพื่อกระทำงานนั้น นั่นหมายความว่าการเขียนโปรแกรมแบบ Structural Programming จะเน้นไปทาง Algorithm โดยไม่สนใจข้อมูล มีผลให้ภาษาที่ใช้เขียนโปรแกรมแบบนี้ ไม่เหมาะกับการพัฒนาโปรแกรมขนาดใหญ่

การเขียนโปรแกรมแบบ Structural Programming จะมีโครงสร้างหลักอยู่ 3 รูปแบบ คือ อนุกรม ทางเลือก และทำซ้ำ ดังนี้

- โครงสร้างอนุกรม โปรแกรมส่วนใหญ่เป็นชุดคำสั่งที่ประกอบด้วยหลายคำสั่งเรียงติดกัน โครงสร้างอนุกรมคือการทำงานทีละคำสั่งจากบนลงล่าง
- โครงสร้างทางเลือก การทำงานจะต้องเลือกทำงานตามเงื่อนไขที่ระบุไว้ และจุดปลายของการทำงานทุกเงื่อนไขต้องรวมกันเป็นเส้นทางเดียว
- โครงสร้างทำซ้ำ จะเป็นการทำงานซ้ำภายใต้เงื่อนไขบางอย่าง

2.1.2. การเขียนโปรแกรมแบบ Object-Oriented Programming

ในปัจจุบันนี้จะนิยมใช้กันมาก ซึ่งจะเน้นข้อมูลหรือเรียกว่า Abstract Data Type (ADT) หรือชนิดข้อมูลใหม่ที่กำหนดขึ้นเอง (User-defined type) การเขียนโปรแกรมเชิงวัตถุนั้น จะมองจากระดับล่างไปยังด้านบน (ต่างจากการเขียนโปรแกรมแบบ Structural Programming ที่มองจากด้านบนลงมายังด้านล่าง) ในระดับล่างจะมองที่รายละเอียดปัญหาเหล่านั้นให้เป็นรูปแบบชนิดข้อมูลใหม่ที่สอดคล้องกับปัญหาเหล่านั้น เมื่อรวมกันเราเรียกว่า เป็นวัตถุหนึ่งหรือ Object หนึ่ง เช่นในภาษา C++ จะใช้คลาส Class เป็นตัวอธิบายรายละเอียดออบเจ็กต์ คลาสก็จะประกอบด้วยข้อมูลและฟังก์ชันการกระทำต่าง ๆ ด้วย ต่อไปจะมองไปยังระดับบน คือภาพรวมการออกแบบโปรแกรม ก่อนที่จะเริ่มต้นเขียนโปรแกรมจริง ๆ

2.1.3 การเขียนโปรแกรมแบบ Generic Programming

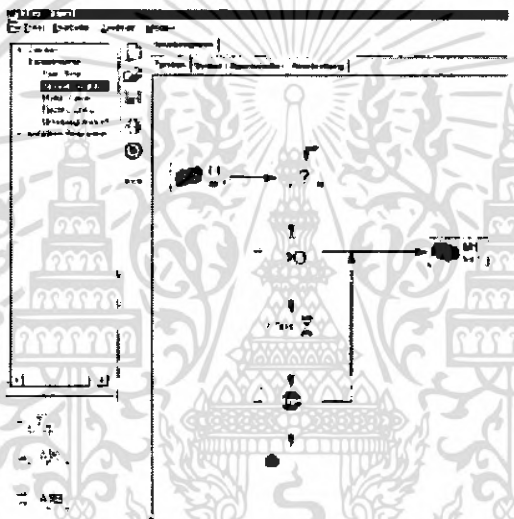
คำว่า Generic ในที่นี้หมายถึง ชนิดข้อมูลเป็นชนิดใด ๆ ก็ได้ โดยไม่เจาะจงเฉพาะการเขียนโปรแกรมแบบนี้จะเป็นอีกระดับหนึ่งของการเขียนโปรแกรมเชิงวัตถุ และจะเน้นที่ Algorithm ในการคำนวณและตัวเก็บคลาส (Container) เช่น อัลกอริทึมในการเรียงข้อมูล เป็นต้น ในการเขียนโปรแกรมแบบนี้จะใช้ Template เป็นหลัก

2.2. Visual Programming Language

Visual programming language (VPL) เป็นภาษาที่ใช้ในการเขียนโปรแกรมคอมพิวเตอร์ ที่ให้ผู้พัฒนาโปรแกรมเขียนโปรแกรมจากการนำรูปภาพต่าง ๆ มาประกอบกันแทนที่การพิมพ์ทำสั่ง VPL อนุญาตให้เขียนโปรแกรมโดยการใช้คำสั่งที่เป็นรูปภาพประกอบกับข้อความ ส่วนใหญ่สัญลักษณ์ที่ใช้ใน VPL จะเป็นกล่องและลูกศร กล่าวคือกล่องหรือรูปทรงต่าง ๆ แสดงถึงการทำงาน ส่วนลูกศรแสดงถึงลำดับการทำงาน ซึ่ง VPL แต่ละภาษาถูกออกแบบมาเพื่อจุดประสงค์ที่ต่างกัน ตัวอย่างของ VPN ที่ใช้กันอย่างแพร่หลาย ได้แก่

- Actor-lab ที่ใช้ในการสอนเรื่อง control technology

- Simulink เป็น Library ของ Mathlab ที่ใช้ในการสร้าง Model จำลองสถานการณ์ การวิเคราะห์ต่างๆ ในหลายแขนง
- Ladder Logic เป็นโปรแกรมที่ใช้สำหรับสร้างตรรกะทางไฟฟ้า ที่ใช้กันอย่างแพร่หลายในการพัฒนาโปรแกรมด้วยภาษา PLC
- LabView เป็นโปรแกรมใช้เขียนโปรแกรมสำหรับงานด้าน data acquisition, instrument control และ industrial automation
- VisSim เป็นโปรแกรมลักษณะเดียวกับ Simulink
- ROBO Pro เป็นภาษาที่ออกแบบมาเพื่อพัฒนาโปรแกรมสำหรับหุ่นยนต์ที่รองรับโปรแกรม ROBO Pro

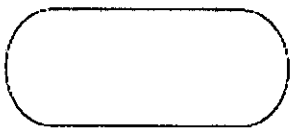


รูปที่ 2.1 โปรแกรม ROBO Pro

2.3. การเขียน flowchart

2.3.1. หลักการเขียน flowchart

การเขียน flowchart เป็นการเขียนแผนการหรือสัญลักษณ์ต่าง ๆ เพื่อแสดงขั้นตอนของงานจากการวิเคราะห์ ถ้าเขียนโปรแกรมจากการวิเคราะห์งานโดยตรงจะทำได้ยาก เพราะการใช้ข้อความหรือคำอธิบายคำสั่งนั้นอาจทำให้งานผิดพลาดได้ และในกรณีที่จะแก้ไข ปรับปรุง หรือเพิ่มเติมส่วนของโปรแกรมบางส่วนจะเข้าใจได้ง่าย flowchart เป็นขั้นตอนวิธีที่เขียนโดยใช้รูปสัญลักษณ์ มีเส้นเชื่อมและหัวลูกศรบอกขั้นตอนการทำงาน การเขียนขั้นตอนจากการวิเคราะห์ด้วยวิธีนี้เป็นที่นิยมมากกว่าแบบอื่น ๆ เนื่องจากมีเส้นลากโยงใยทำให้เห็นขั้นตอนการทำงานที่ชัดเจน มีลูกศรกำกับทิศทางการทำงานช่วยให้เข้าใจง่ายขึ้น และสามารถตรวจสอบความถูกต้องได้ง่าย รูปที่ 2.2 แสดงตัวอย่างสัญลักษณ์ของ flowchart



แทนการเริ่มต้นหรือการหยุดชั่วขณะ

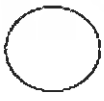
หรือการสิ้นสุดของขั้นตอนวิธี

แทนการรับหรือแสดงข้อมูล โดยไม่ระบุชนิดของสื่อบันทึกข้อมูลที่ใช้

แทนการกำหนดค่าหรือการคำนวณค่า

แทนการเปรียบเทียบตัดสินใจ

แทนกลุ่มขั้นตอนวิธี(โปรแกรมย่อย)



แทนจุดเชื่อมที่อยู่ภายในหน้าเดียวกัน

แทนเส้นแสดงทิศทางการทำงานตามปลายลูกศร

รูปที่ 2.2 ตัวอย่างสัญลักษณ์ flowchart

Flowchart เปรียบเสมือนแบบแปลนการสร้างบ้าน ที่สถาปนิกสามารถนำผังงานมาใช้แสดงลำดับขั้นตอนการกระทำต่าง ๆ ในชีวิตประจำวัน ในด้านการเขียนโปรแกรมคอมพิวเตอร์ ผังงานเป็นเครื่องมือที่ช่วยในการเขียนโปรแกรมดำเนินไปอย่างมีประสิทธิภาพ ใช้เป็นหลักฐานทางเอกสารสำหรับการแก้ไขปรับปรุง บำรุงรักษาโปรแกรมในอนาคตได้อย่างรวดเร็ว ดังนั้นเพื่อให้ผู้ใช้มีความเข้าใจที่ถูกต้องร่วมกัน จึงเป็นต้องมีกฎเกณฑ์เกี่ยวกับผังงาน และการกำหนดรูปสัญลักษณ์เพื่อใช้แทนความหมายต่าง ๆ ซึ่งหลักในการเขียนผังงานที่ดีมีดังนี้

- เลือกใช้สัญลักษณ์ที่มีรูปแบบมาตรฐานสากล ส่วนขนาดของสัญลักษณ์ขึ้นอยู่กับความเหมาะสม
- ควรเขียนให้ทิศทางของ flowchart เริ่มจากบนลงล่าง (Top to bottom) หรือจากซ้ายไปขวา (Left to right) และควรใช้หัวลูกศรกำกับทิศทางของ flowchart ด้วย
- เขียนข้อความที่ต้องการอธิบายการปฏิบัติงานในขั้นตอนนั้น ๆ ภายในกรอบสัญลักษณ์ ข้อความที่ใช้ควรง่ายและสั้น
- ควรใช้สัญลักษณ์ตัวเชื่อม (Connection) แทน ในกรณีที่ต้องการขีดเส้นเชื่อมโยงการปฏิบัติงานซึ่งอยู่ห่างกันมาก
- สัญลักษณ์ใน flowchart จะต้องไม่มีการปล่อยจุดไว้โดยไม่มีเส้นลากเชื่อมต่อ
- ครอบคลุมทุกขั้นตอน หรือเงื่อนไขของการปฏิบัติงานในงานนั้นๆ
- Flowchart ควรที่จะชัดเจน และเป็นระเบียบเรียบร้อย สามารถเข้าใจง่ายและติดตามขั้นตอนได้ง่าย
- ควรมีวิธีการทดสอบ flowchart เพื่อนำไปเขียนโปรแกรมได้ง่ายขึ้น

2.3.2. ข้อดีและข้อเสียของการเขียน flowchart

ข้อดีของการเขียน flowchart คือ

- ทำให้เห็นภาพลำดับต่าง ๆ ของขั้นตอนการปฏิบัติการ และความเกี่ยวข้องระหว่างส่วนต่าง ๆ ของงานแต่ละขั้นตอนได้ชัดเจนและกะทัดรัด
- ทำให้เห็นลำดับขั้นตอนที่ถูกหลักตรรกะวิทยา
- ง่ายแก่บุคคลภายนอกในการติดตามขั้นตอนของการปฏิบัติงาน
- flowchartที่แสดงรายละเอียด จะช่วยให้การเขียนโปรแกรมเป็นไปได้อย่างสะดวกและรวดเร็ว

ข้อเสียของการเขียน flowchart คือผลลัพธ์ที่ได้จากการเขียน flowchart ไม่ได้ออกมาในรูปของโปรแกรม ผู้พัฒนาโปรแกรมจึงจำเป็นจะต้องแปลง flowchart เป็น source code ก่อนจึงสามารถนำไป compile เป็นโปรแกรมได้

2.4. การเก็บรูปแบบแบบ Bitmap และ Vector

รูปภาพที่สามารถนำมาประมวลผลด้วยคอมพิวเตอร์สามารถแบ่งเป็น 2 กลุ่ม คือ bitmap image (raster image) และ vector image ซึ่งทั้งสองกลุ่มมีลักษณะของการเก็บข้อมูลที่ไม่เหมือนกัน ความเหมาะสมในการนำไปใช้งานก็ไม่เหมือนกัน ทั้งสองแบบต่างก็มีทั้งข้อดีและข้อเสีย ซึ่งในหัวข้อนี้จะกล่าวถึงการเก็บข้อมูลรูปภาพทั้งสองแบบ

2.4.1. รูปภาพในรูปแบบ bitmap image

รูปภาพในรูปแบบของ bitmap image จะมีการเก็บข้อมูลโดยการแทนแต่ละจุดในภาพด้วยเลขฐานสองที่แสดงถึงค่าสี ดังนั้นรูปภาพก็เป็นเสมือนกับ matrix ของเลขฐานสองที่มีขนาดเท่ากับจำนวนจุดในรูป รูปที่ 2.x แสดงรูปภูเขาซึ่งมีรูปแบบของรูปภาพแบบ bitmap image ด้านซ้ายเป็นภาพขนาดปกติ ส่วนด้านขวาเป็นภาพที่ขยาย 250% จากภาพด้านขวาจะมองเห็นจุดสีที่ชัดเจน ซึ่งจำนวนจุดมีจำนวนมากทั้งแนวตั้งและแนวนอน

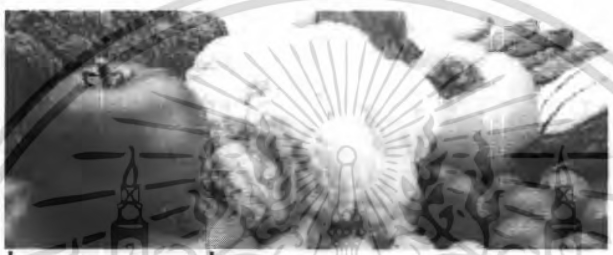
รูปที่ 2.3 รูปภูเขาที่มีลักษณะภาพแบบ bitmap image และภาพขยาย 250%

การเก็บข้อมูลรูปภาพแบบ bitmap image ยังสามารถแบ่งเป็นกลุ่มย่อย ๆ ได้ 4 กลุ่มคือ Line-art, Grayscale image, Multitones และ Full color image โดยแบ่งตามสีของรูปภาพที่สามารถเก็บได้

Line-art คือรูปภาพที่มีเพียงสองสี คือสีขาวและสีดำ จึงทำให้จำนวนเลขฐานสองที่ใช้ในการแทนค่าจุดสีใช้เพียงตำแหน่งเดียว

รูปที่ 2.4 ตัวอย่างรูปที่มีการเก็บข้อมูลแบบ Line-art

Grayscale image คือภาพที่ประกอบด้วยจุดสีที่มีค่าอยู่ในเฉดสีของสีเทา กล่าวคือสามารถไล่ระดับสีเทาได้ ซึ่งจำนวนระดับสีจะขึ้นอยู่กับจำนวนเลขฐานสองที่ใช้แทนในแต่ละจุด



รูปที่ 2.5 ตัวอย่างรูปที่มีการเก็บข้อมูลแบบ Grayscale image

Multitones คือรูปภาพที่ประกอบด้วยเฉดสีของสองสีขึ้นไป โดยมากจะเป็นสองสีคือสีดำและสีอื่น ๆ อีกหนึ่งสีเรียกว่า spot color

รูปที่ 2.6 ตัวอย่างรูปที่มีการเก็บข้อมูลแบบ Multitones

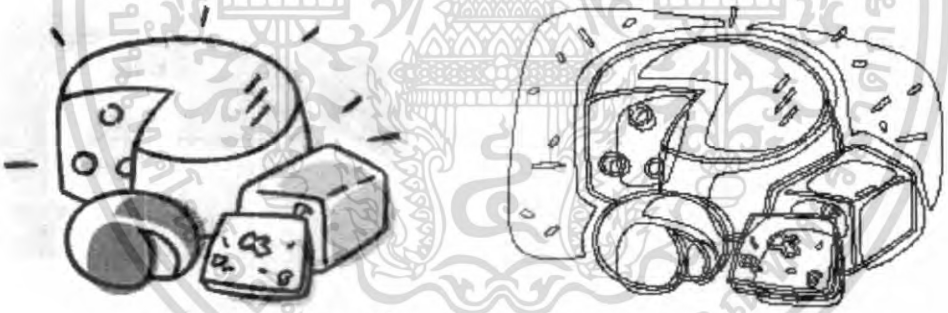
Full color image ค่าจุดสีในการเก็บข้อมูลจะใช้รูปแบบต่าง ๆ เช่น RGB, CMYK, HSI โดยค่าสีที่ได้จะสามารถครอบคลุมสีต่าง ๆ ได้จำนวนมากหรือไม่ขึ้นอยู่กับจำนวนเลขฐานสองที่ใช้ในการแทนค่าจุดสี

รูปที่ 2.7 ตัวอย่างรูปที่มีการเก็บข้อมูลแบบ Full color image

การนำภาพแบบ bitmap image ไปประยุกต์ใช้งานมักจะนำไปใช้กับงานที่มีการเก็บรูปภาพที่ไม่ใช่รูปทรงเรขาคณิต ไม่สามารถบรรยายรูปภาพเป็นคำพูดได้ ได้แก่ การเก็บภาพถ่าย การเก็บภาพวาด การเก็บภาพลายนิ้วมือ เป็นต้น ถึงแม้ว่ารูปภาพแบบ bitmap image จะสามารถแสดงรูปได้ทุกรูปแบบ แต่ก็ใช้หน่วยความจำในการเก็บภาพจำนวนมาก

2.4.2. รูปภาพในรูปแบบ vector image

การเก็บรูปภาพในแบบ vector image คือการเก็บข้อมูลที่สามารถอธิบายลักษณะข้อมูลได้ด้วยคณิตศาสตร์ จากรูปที่ 2.8 ในด้านซ้ายเป็นรูปภาพแบบ vector image ส่วนด้านขวาเป็นเส้นที่สร้างได้จากสมการคณิตศาสตร์ที่ใช้ในการวาดรูป



รูปที่ 2.8 ตัวอย่างรูปที่มีการเก็บข้อมูลแบบ Vector image

เส้นแต่ละเส้นสร้างมาจากกลุ่มของจุดหลาย ๆ จุดกับเส้นซึ่งเชื่อมต่อระหว่างจุด หรือสร้างมาจากจุดเพียงบางจุดที่เชื่อมด้วยเส้นโค้งที่เขียนเป็นสมการคณิตศาสตร์ได้

ขนาดไฟล์ของรูปภาพแบบ vector image จะมีขนาดที่เล็กเพราะข้อมูลที่เก็บมีเพียงจุดและเส้นที่ได้จากการวาดเท่านั้น ในการขยายภาพแบบ vector image จะมีไม่การสูญเสียคุณภาพเพราะรูปภาพเกิดจากสมการคณิตศาสตร์ จากคุณสมบัตินี้จึงมีคนทำลักษณะการเก็บข้อมูลแบบ vector image ไปใช้กับการเก็บข้อมูลรูปภาพเครื่องหมายการค้า

2.5. การสื่อสารระหว่าง process

ในโปรแกรมหรือระบบโดยทั่วไปจะมีวิธีการในการสื่อสารระหว่าง process ซึ่งแต่ละ process ทำงานอิสระต่อกัน เรียกการสื่อสารแบบนี้ว่าการสื่อสารระหว่าง process (IPC : Inter-process communication) ในการทำงานร่วมกันในบางครั้ง ความเร็วในการทำงานของแต่ละ process ไม่เท่ากัน จึงต้องมีการปรับให้ทำงานพร้อมกันเรียกว่า synchronization ในหัวข้อนี้จะแบ่งเป็นสองส่วนคือ การส่งข้อมูลระหว่าง process และการทำ synchronization ระหว่าง process

2.5.1. การส่งข้อมูลระหว่าง process

การส่งข้อมูลระหว่าง process สามารถแบ่งเป็นสองรูปแบบคือ การส่งข้อมูลทางตรง และการส่งข้อมูลทางอ้อม

การส่งข้อมูลทางตรงการติดต่อแบบนี้จะต้องกำหนดชื่อเฉพาะในการติดต่อทั้งผู้รับและผู้ส่ง ยกตัวอย่างเช่น ถ้าต้องการส่งข้อความจาก A ไป B จะมีการกำหนดรูปแบบคือ

Send(B, Message)

Receive(A, Message)

ในการส่งข้อมูลแบบนี้จะเสมือนกับการสร้างเส้นทางการสื่อสารเฉพาะสำหรับ 2 process เท่านั้น โดยทั้ง process A และ B จะต้องรู้หมายเลข process ของฝ่ายตรงข้ามด้วย

การส่งข้อมูลทางอ้อม การติดต่อแบบนี้โปรเซสทั้งสองที่ต้องการจะติดต่อกันจะติดต่อกันผ่านทาง Mailbox หรืออาจเรียกว่าเป็นการติดต่อทางพอร์ต และ mailbox ที่ใช้ในการติดต่อกันนี้จะต้องมีการกำหนดว่าจะใช้ mailbox ร่วมกันไว้ก่อนด้วย รูปแบบคำสั่งการรับ-ส่งข้อมูลผ่าน mailbox เป็นดังนี้

Send(MailboxA, Message)

Receive(MailboxA, Message)

การส่งข้อมูลทางอ้อมจะมีคุณสมบัติดังนี้

- จะมีการสร้างลิงค์ระหว่าง process ที่มีการแชร์ mailbox เท่านั้น
- ลิงค์หนึ่ง ๆ อาจจะมีความสัมพันธ์มากกว่าสอง process ก็ได้
- ระหว่าง process แต่ละคู่นั้นอาจมีหลายลิงค์ที่แตกต่างกันได้ แต่ละลิงค์จะมีเพียง mailbox เดียว
- การลิงค์อาจเป็นทิศทางเดียว หรือสองทิศทางก็ได้

รูปแบบต่างๆ ของ mailbox สามารถแบ่งได้เป็นประเภทต่าง ๆ ดังนี้ Queue Mailbox, Pipe Mailbox และ Stack Mailbox

Queue Mailbox คือ mailbox ที่มีโครงสร้างที่ดึงข้อมูลออกจาก mailbox ตามลำดับก่อน-หลังของข้อมูลที่ส่งมา นั่นคือข้อมูลใดที่ส่งเข้ามาใน mailbox ก่อนก็จะถูกดึงออกไปก่อน

ส่วนข้อมูลใดส่งเข้ามาภายหลังก็จะถูกดึงออกไปภายหลัง อาจเรียกการทำงานแบบนี้ว่า FIFO (First In First Out) ลักษณะโครงสร้าง mailbox แบบคิว

Pipe Mailbox คือ mailbox ที่มีโครงสร้างแบบคิว คือการดึงข้อมูลจะเป็นในลักษณะที่ข้อมูลส่งเข้ามาก่อนจะถูกดึงออกไปก่อน ข้อมูลใดส่งเข้ามาภายหลังก็จะถูกดึงออกไปใช้งานภายหลัง แต่ข้อแตกต่าง ระหว่าง mailbox แบบคิวกับ mailbox แบบ Pipe คือ mailbox แบบคิวจะมีขนาดคงที่ ลักษณะโครงสร้างของ mailbox แบบ pipe mailbox จะขยายตัวอัตโนมัติ

Stack Mailbox โครงสร้างของ mailbox แบบนี้เป็นโครงสร้างตรงข้ามกับ mailbox แบบคิวในการดึงข้อมูล นั่นก็คือข้อมูลใดส่งเข้ามา mailbox ก่อนจะถูกดึงออกไปใช้งานภายหลัง โดยจะนำข้อมูลที่ส่งเข้ามาภายหลังออกไปใช้การก่อน อาจเรียกการทำงานแบบนี้ว่า FILO(First In Last Out) ลักษะโครงสร้าง mailbox แบบ Stack

2.5.2. การทำ Synchronization

โดยปกติ process ที่กำลังทำงานอยู่จะไม่มีความเกี่ยวข้องกัน กล่าวคือไม่ได้ทำงานร่วมกัน ลักษณะความเป็นอิสระนี้เรียกว่า Asynchronous แต่ในกรณีที่ process มีความสัมพันธ์กัน เช่นต้องสลับกันทำงานเป็นจังหวะ จะต้องมีการทำให้แต่ละ process เข้าจังหวะกัน เรียกว่า synchronization เพื่อให้ process ทำงานร่วมกันตามลำดับที่วิเคราะห์ไว้

ในบางระบบปฏิบัติการ process ที่ทำงานร่วมกันอาจมีการใช้ resource ร่วมกัน เช่น หน่วยความจำ หรือสื่อจัดเก็บข้อมูล ซึ่ง process ทั้ง 2 สามารถอ่านหรือเขียน resource ได้พร้อมกัน ทำให้ผลลัพธ์เกิดการผิดพลาด สภาวะที่เกิดขึ้นนี้เรียกว่า Race Condition ซึ่งสามารถแก้ไขได้โดยต้องหยุดไม่ให้ process ที่สองทำงาน ในขณะ process แรกทำงานอยู่ ถ้ายังมี process แรกครอบครอง resource อยู่ โดยจะต้องรอจนกว่า process แรกทำงานเสร็จเรียบร้อยเพื่อป้องกันไม่ให้เกิดข้อผิดพลาด วิธีป้องกันไม่ให้ process อื่นเข้าไปใช้ resource ขณะที่ process ใด ๆ ครอบครอง resource นั้นเรียกว่า Mutual Exclusion

ในปี ค.ศ. 1965 Dijkstra ได้แนะนำให้ใช้ตัวแปรจำนวนเต็มเรียกว่า Semaphore โดย semaphore จะตรวจสอบค่าตัวแปร ถ้าค่ามากกว่า 0 จะลดค่าแล้ว ดำเนินต่อไป แต่ถ้าค่าเป็น 0 process จะเป็น sleep ตัวเองโดยไม่จำเป็นต้องให้ปฏิบัติการ down เสร็จสมบูรณ์ก่อน การตรวจสอบเปลี่ยนแปลงจนนำไปสู่การ sleep เป็นขั้นตอนเดียวโดยอัตโนมัติ จึงอาจเรียกได้ว่า เป็น indivisible automatic action เป็นการยืนยันได้ว่า ปฏิบัติการของ semaphore เพียงปฏิบัติการเดียว ที่เริ่มต้น ทำให้ process อื่นไม่สามารถเข้าถึงsemaphore ได้จนกว่าปฏิบัติการจะเสร็จสมบูรณ์หรือมีการบล็อกความเป็นอัตโนมัตินี้เองเป็นสิ่งสำคัญในการแก้ปัญหา synchronization และหลีกเลี่ยงการ race condition สำหรับปฏิบัติการ Up จะเพิ่มค่าของ address ถ้ามี process ตั้งแต่ 1 process ขึ้นไป sleep บน semaphore นั้นจะไม่สามารถทำให้ปฏิบัติการ down ก่อนหน้านั้นเสร็จสิ้นลงไปได้ แต่ระบบจะปิดบางตัว และยอมให้เสร็จสิ้นได้ ทำให้เกิดปฏิบัติการ up บน semaphore ที่สุ่มนั้นทำให้เพิ่มค่า

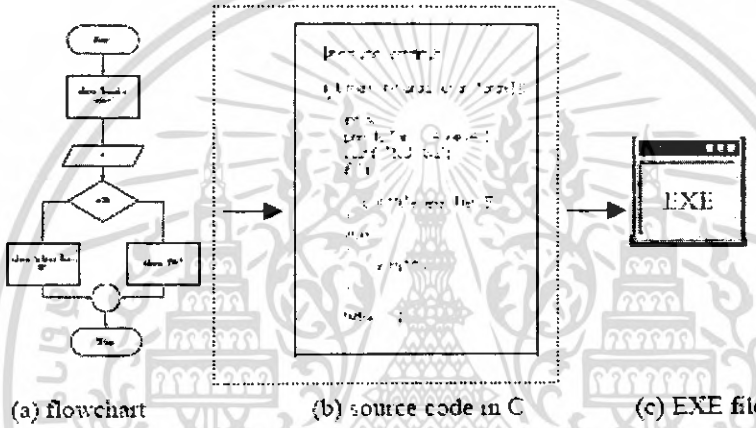
semaphore ขึ้น อีก 1 ทำให้ wakeup ได้ และเช่นเดียวกับปฏิบัติการ down ที่ในระหว่างดำเนินตามปฏิบัติการทั้งสอง จะไม่สามารถ interrupt นอกจากจะใช้ disable interrupt

Mutex เป็น Semaphore อย่างง่ายที่นำมาใช้ในการจัดการ Mutual Exclusion กับ resource ที่ใช้ร่วมกันได้อย่างมีประสิทธิภาพ โดยเฉพาะกับ Thread โดย Mutex เป็นตัวแปรที่มีสภาวะได้ 2 สภาวะ คือ unlocked และ locked ที่ใช้เพียง 1 บิต

บทที่ 3
การออกแบบและพัฒนา

3.1. บทนำ

ขั้นตอนในการพัฒนาโปรแกรมภายใต้ Visual Programming using Flowchart (VPF) Environment นั้น นักพัฒนาจะเขียนโปรแกรมโดยการวาด Flowchart จากนั้นส่งให้กับ VPF Environment จัดการแปลงเป็นโปรแกรม EXE โดยที่ VPF Environment จะทำหน้าที่แปลง Flowchart เป็น Source code ภาษาใดภาษาหนึ่ง ในที่นี้เลือกใช้ภาษาซี จากนั้นใช้ Compiler ภาษานั้นๆ เพื่อแปลง Source code เป็นโปรแกรม EXE



รูปที่ 3.1 แสดงขั้นตอนการพัฒนาโปรแกรมบน VPF Environment

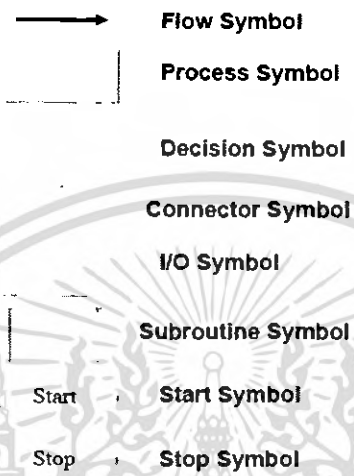
Visual Programming using Flowchart (VPF) Environment ถูกพัฒนาขึ้นด้วย JAVA โดยในขั้นตอนของการออกแบบโปรแกรมสามารถแบ่งเป็น 3 ส่วนใหญ่ๆ คือ (1) โปรแกรมส่วนเครื่องมือที่ใช้ในการวาด flowchart (2) โปรแกรมส่วนการแปลง flowchart เป็น source code (3) โปรแกรมส่วนเครื่องมือที่ใช้ run และ debug โปรแกรม

3.2. การออกแบบโปรแกรมส่วนเครื่องมือวาด flowchart

สามารถวาดสัญลักษณ์ เลือกวัตถุ ลบและแก้ไขวัตถุได้ สัญลักษณ์ที่โปรแกรมสามารถวาดได้แสดงในรูปที่ 3.1 สัญลักษณ์ที่โปรแกรมสามารถวาดได้คือ

- Flow Symbol ใช้ในการลากเส้นลำดับการทำงานของโปรแกรม
- Process Symbol ใช้ในการใส่คำสั่งต่างๆ
- Input/Output Symbol ใช้ในการรับค่าจากคีย์บอร์ดเก็บใส่ตัวแปร
- Decision Symbol ใช้ในการสร้างเงื่อนไขในการเขียนโปรแกรม

- Connector Symbol ใช้ในการวางจุดเชื่อมต่อจากการสร้างเงื่อนไข
- Start Symbol ใช้ในการกำหนดจุดเริ่มต้นของโปรแกรม
- Stop Symbol ใช้ในการกำหนดจุดสิ้นสุดของโปรแกรม
- Subroutine Symbol ใช้ในการเรียกใช้งานชุดคำสั่งที่เคยเขียนไว้ใน file อื่น



รูปที่ 3.2 สัญลักษณ์ที่สามารถวาดได้ใน VPF Environment

ซึ่งแต่ละสัญลักษณ์ก็มีข้อกำหนดที่แตกต่างกันตามหน้าที่การทำงาน ดังนี้

- Flow Symbol – ต้องมีจุดเริ่มต้นและสิ้นสุดเป็นคนละวัตถุกัน
- Process symbol, input symbol and subroutine symbol – จำนวนเส้นเข้าและจำนวนเส้นออกต้องมีอย่างละ 1 เส้นเท่านั้น
- Decision symbol – ต้องมี 1 เส้นเข้า และมี 2 เส้นออก เส้นออกหนึ่งสำหรับเงื่อนไขที่เป็นจริง อีกเส้นออกหนึ่งสำหรับเงื่อนไขที่เป็นเท็จ
- Connector symbol – ต้องมี 2 เส้นเข้า และมี 1 เส้นออก
- Start symbol - ต้องมี 1 เส้นออก และไม่มีเส้นเข้า
- Stop symbol – ต้องมี 1 เส้นเข้าและไม่มีเส้นออก

นอกจากนี้แต่ละวัตถุยังมีคุณสมบัติที่จะต้องเก็บไว้ เช่น เลขประจำตัว (Symbol ID) ชนิดของสัญลักษณ์ (Symbol Type) ข้อความ (Text) ตำแหน่ง (Position) วัตถุถัดไป (Next Symbol ID) ฯลฯ ข้อความที่กำหนดไว้สำหรับวัตถุแต่ละชนิดนั้น จะมีความหมายแตกต่างกันไป ดังนี้

- ข้อความใน Process symbol หมายถึงคำสั่ง นักพัฒนาโปรแกรมสามารถใช้คำสั่งของภาษา C หรือคำสั่งของ Visual Programming using Flowchart ก็ได้
- ข้อความใน Input symbol หมายถึงชื่อตัวแปรที่จะเก็บที่ที่ผู้ใช้ใส่ผ่านทางแป้นพิมพ์

- ข้อความใน Decision symbol หมายถึงเงื่อนไข
- ข้อความใน Start symbol หมายถึงการประกาศตัวแปรที่ใช้ประกอบการเรียกใช้ function
- ข้อความใน Stop symbol หมายถึงค่าที่จะส่งคืนเมื่อจบโปรแกรม (ค่าที่ return ใน function)
- ข้อความใน Subroutine symbol หมายถึงชื่อของ file ที่จะนำ flowchart ใน file นั้นมาเรียกใช้งาน

ภาพวาดของ flowchart จะถูกบันทึกลงในไฟล์นามสกุล vpf วัตถุต่างๆ ที่วาดอยู่ใน flowchart จะได้รับการบันทึกในไฟล์ โดยมีรูปแบบการจัดเก็บของวัตถุ ดังนี้

- วัตถุประเภท process, connector, start, stop, input and subroutine symbol จะแทนด้วย

```
symbol type [//] symbol ID [//] position [//] text[//]
next symbol ID [//] no. of "in" flow line [/end/]
```

- วัตถุประเภท decision symbol จะแทนด้วย

```
symbol type [//] symbol ID [//] position [//] text[//]
next symbol ID when condition is true [//]
next symbol ID when condition is false [//]
no. of "in" flow line [/end/]
```

- วัตถุประเภท flow symbol จะแทนด้วย

```
symbol type [//] symbol ID [//] start symbol ID [//]
stop symbol ID [//] status (true or false) [/end/]
```

ทุกๆ วัตถุใน Flowchart จะถูกเก็บอยู่ใน Object Space ซึ่งเป็นเหมือนที่เก็บข้อมูลเพื่อใช้ในการแสดงผล และแปลงเป็น source code หรือใช้ในการบันทึกข้อมูล

3.3. การออกแบบโปรแกรมส่วนการแปลง flowchart เป็น source code

ข้อมูลที่ถูกส่งให้กับเครื่องมือนี้คือ Object Space ซึ่งเป็นเหมือน List ของสัญลักษณ์ที่สร้างขึ้นโดยโปรแกรมส่วนการวาด flowchart เครื่องมือที่ใช้ในการแปลง flowchart นี้มีหน้าที่สำคัญ 3 ประการคือ

1. ตรวจสอบความถูกต้องของ Flowchart
2. แปลง Flowchart เป็น Source code
3. เรียกใช้งาน C Compiler เพื่อแปลง Source code เป็นโปรแกรม EXE

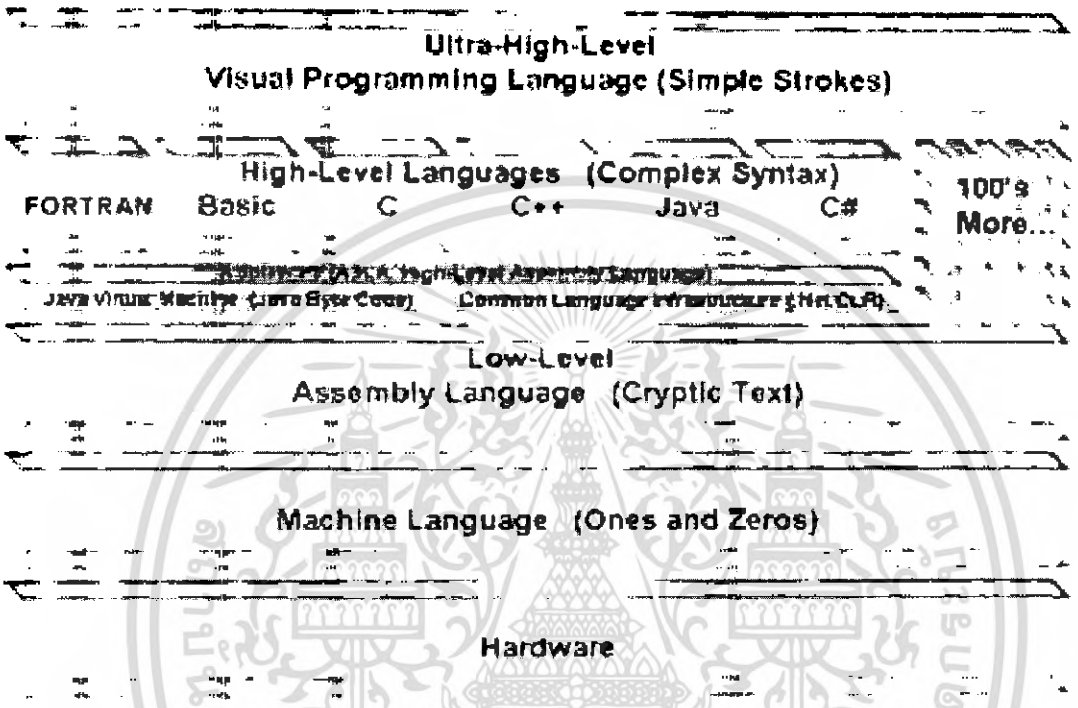
ในขั้นตอนการตรวจสอบความถูกต้องของ Flowchart เครื่องมือนี้จะทำการตรวจสอบข้อมูลที่เก็บอยู่ใน Object Space โดยแบ่งการตรวจสอบเป็น 3 ขั้นตอน ดังนี้

1. ตรวจสอบว่า Flowchart ถูกต้องตามข้อกำหนด
2. ตรวจสอบว่า Flowchart ประกอบด้วย 1 Start Object และ 1 Stop Object
3. ตรวจสอบว่าทุกเส้นทางเริ่มจาก Start Object และไปสิ้นสุดที่ Stop Object

สำหรับในขั้นตอนการตรวจสอบแปลง Flowchart เป็น Source code จะเป็นการตรวจสอบรูปแบบคำสั่ง เปลี่ยนคำสั่งต่างๆ ให้เป็นคำสั่งภาษา C ค้นหาตัวแปร และประกาศตัวแปรไว้ส่วน

บนสุด (file scope) ตามด้วย Sub Routine จากนั้นจึงทำการแปลง Flowchart ให้เป็น Source code พร้อมทั้งจัดรูปแบบย่อหน้าให้สวยงาม

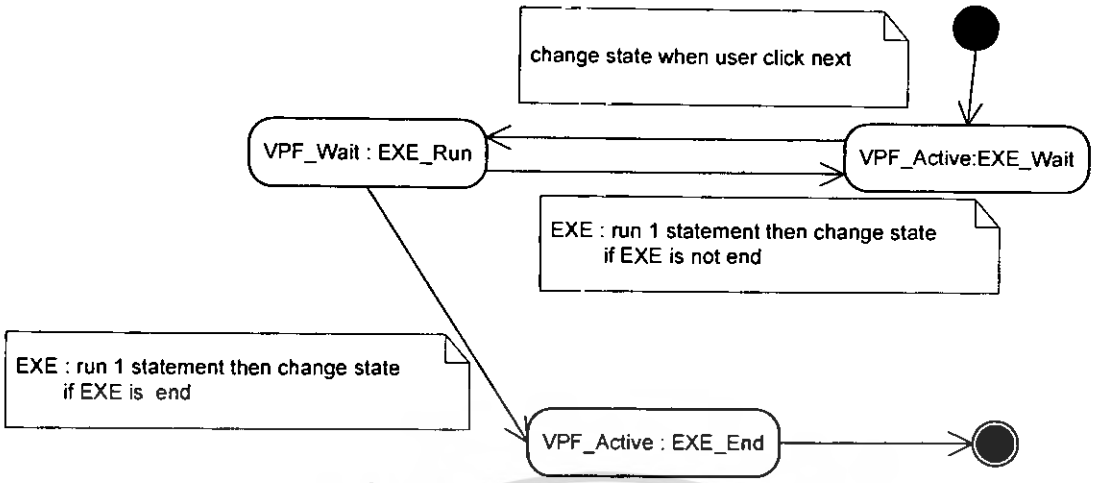
ขั้นสุดท้ายเป็นการเรียก C Compiler ให้ทำการแปลงจาก Source code เป็นโปรแกรม EXE ในที่นี้ใช้ Compiler ของ Borland C++ Compiler 5.5 ซึ่งเป็น Freeware



รูปที่ 3.3 ลำดับของภาษาคอมพิวเตอร์

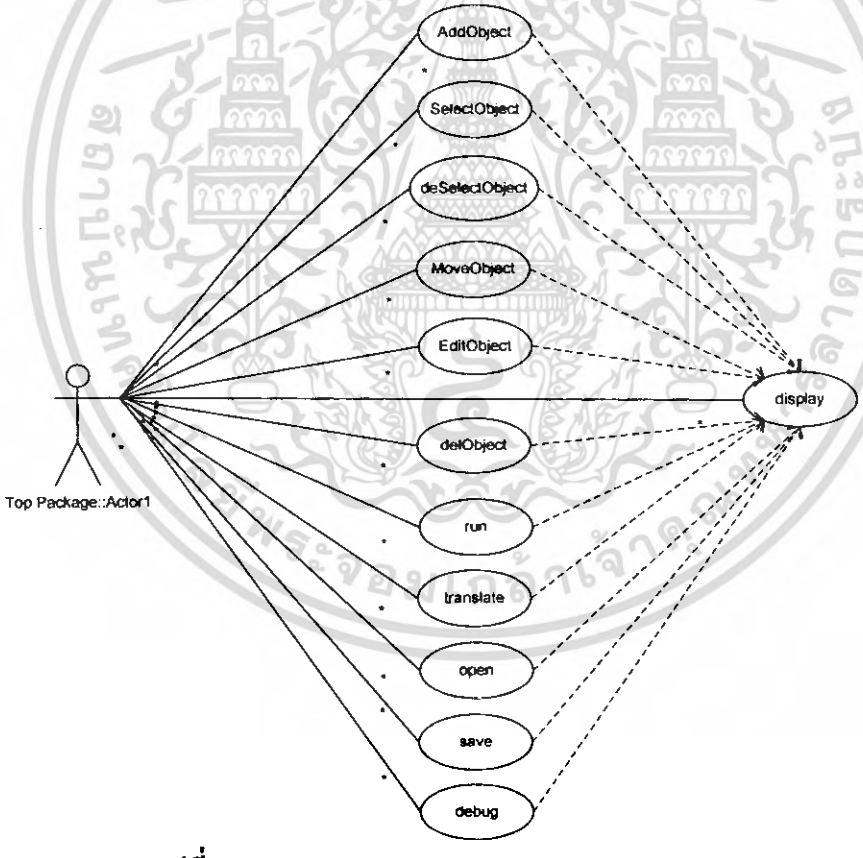
3.4. การออกแบบโปรแกรมส่วนเครื่องมือที่ใช้ debug โปรแกรม

การ debugging หมายถึงการแสดงโปรแกรมที่ละขั้นและดูค่าของตัวแปรในขณะนั้น เครื่องมือนี้จะแสดงเน้นตำแหน่งของจุดที่กำลังทำงานอยู่ ซึ่งทำให้ผู้พัฒนาสามารถมองเห็นภาพได้อย่างชัดเจน โดยไม่ต้องจินตนาการเองจาก Source code โดยที่ในการ debug จะต้องมีการสื่อสารกันระหว่าง VPF Environment และโปรแกรม EXE ที่กำลังทำงานอยู่ วิธีการสื่อสารอาศัยหลักการของ mailbox โดยใช้ไฟล์เป็นสื่อกลาง กล่าวคือเมื่อโปรแกรม EXE ทำงานเสร็จ 1 คำสั่งจะส่งข้อมูล Symbol ID ของคำสั่งถัดไปไปให้กับ VPF Environment หาก VPF Environment ยังไม่ได้อ่านข้อมูลในไฟล์ โปรแกรม EXE จะไม่สามารถทำงานต่อไปได้ เมื่อต้องการให้ทำคำสั่งถัดไปต้องกดคำสั่ง Next ที่ VPF Environment เพื่อให้ VPF เปลี่ยน flag ในไฟล์เพื่อแสดงว่าอ่านข้อมูลแล้ว เมื่อโปรแกรม EXE พบว่า flag ในไฟล์เปลี่ยนไปก็จะทำคำสั่งถัดไปพร้อมกับส่งข้อมูลไปให้ไฟล์ใหม่

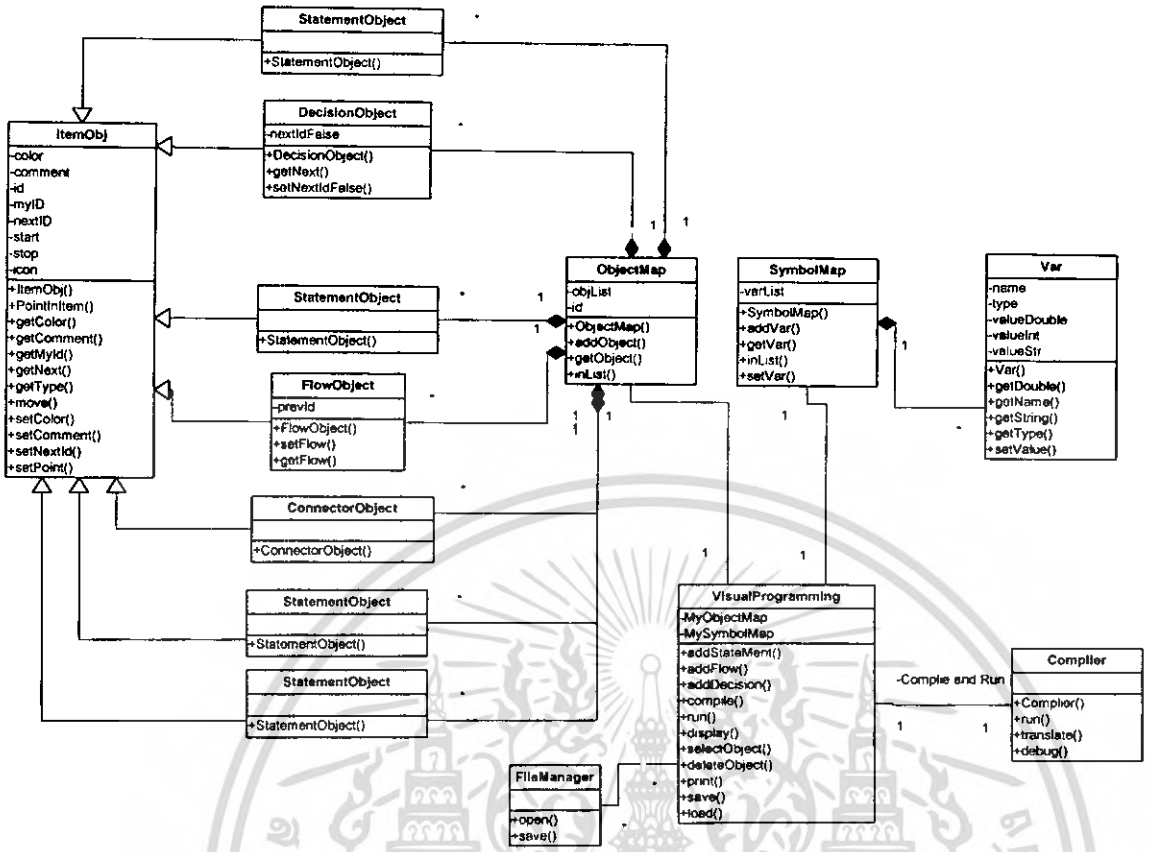


รูปที่ 3.4 State diagram ของ Debugger Tool

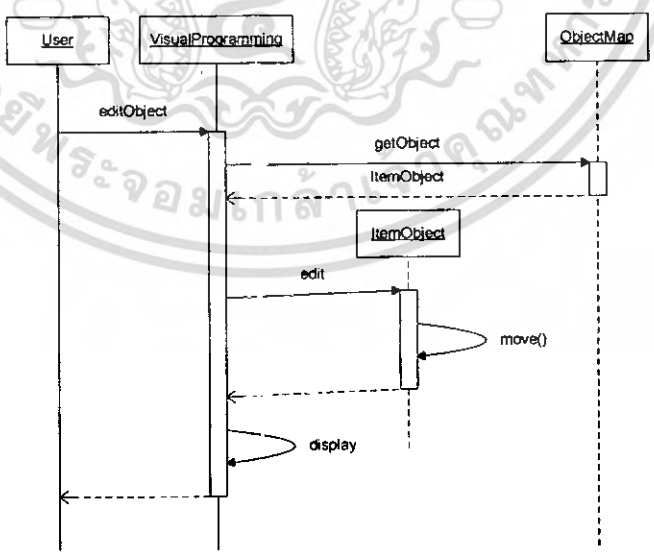
3.5. UML Diagram



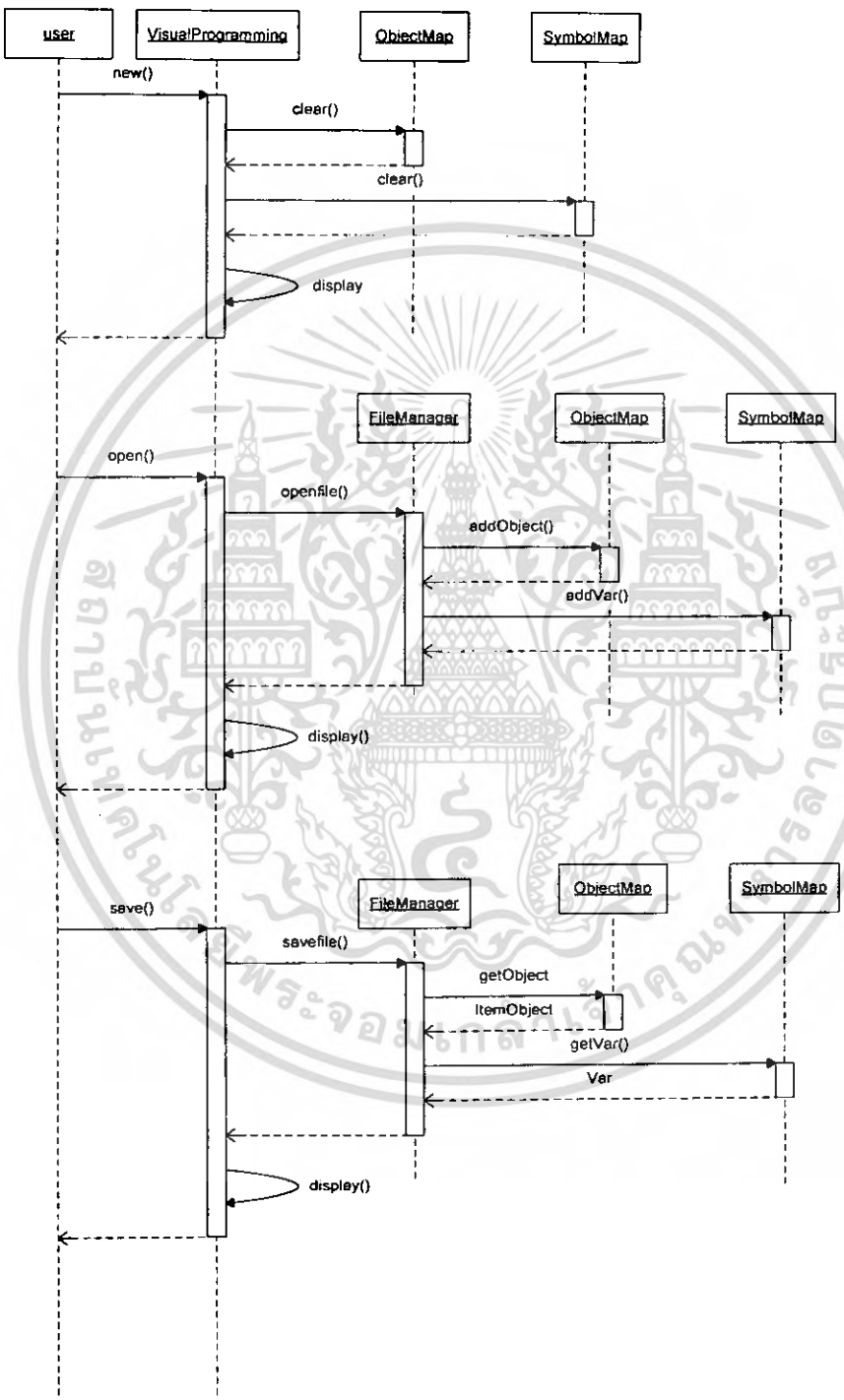
รูปที่ 3.5 Use case diagram ของ VPF Environment



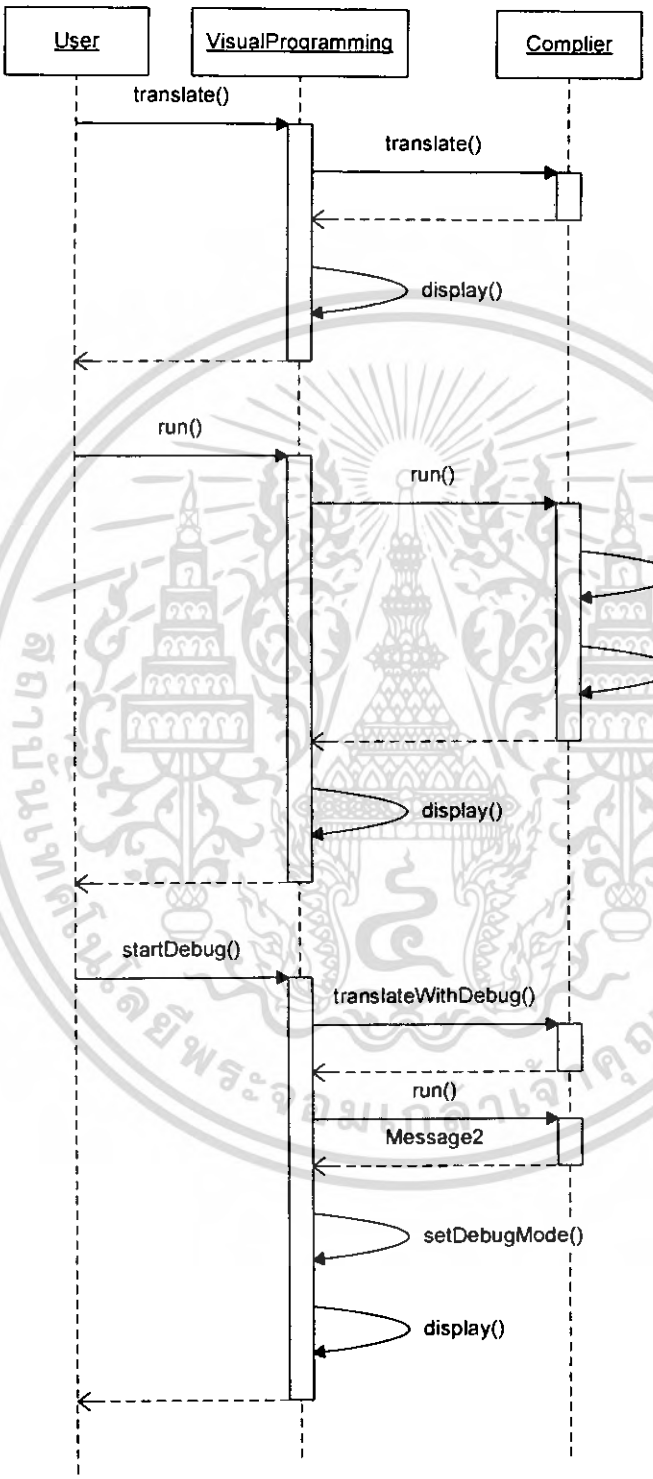
รูปที่ 3.6 Class diagram ของ VPF Environment



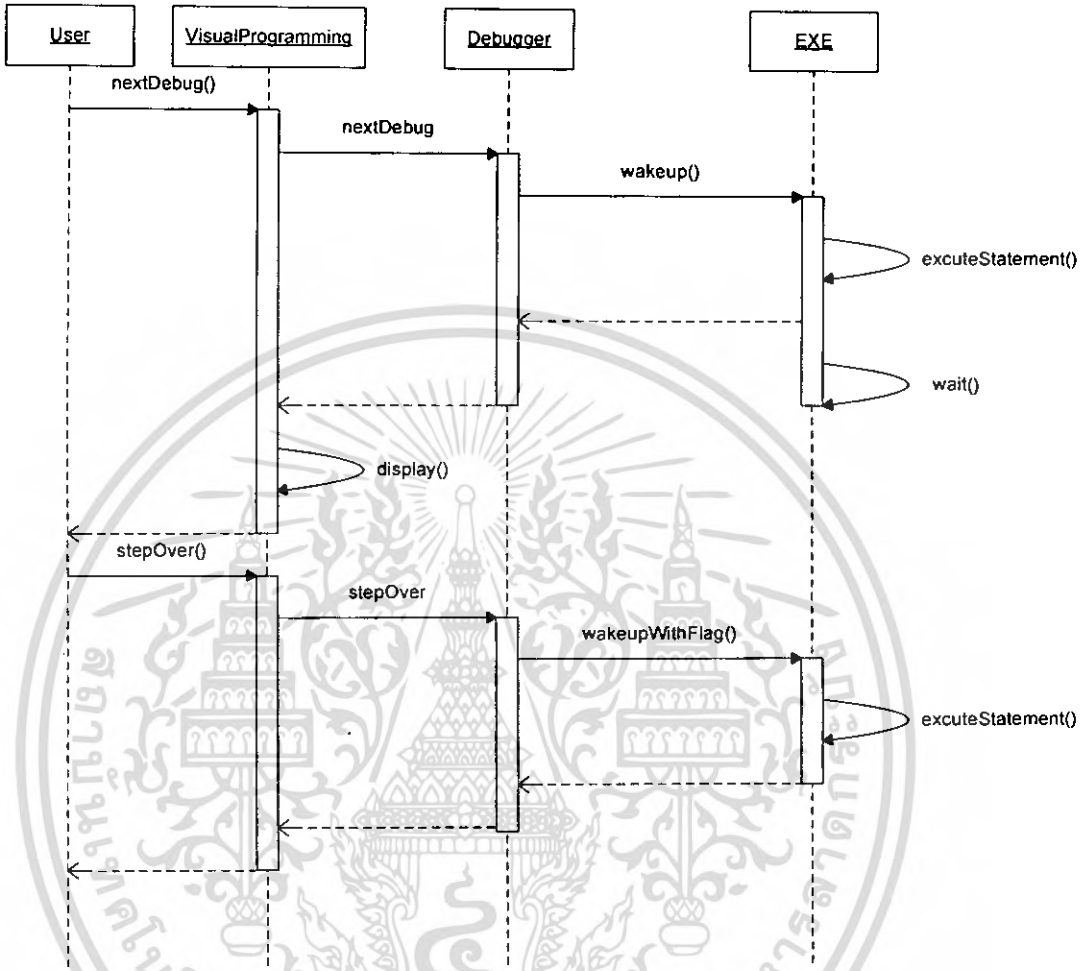
รูปที่ 3.7 Sequence diagram ของ VPF Environment



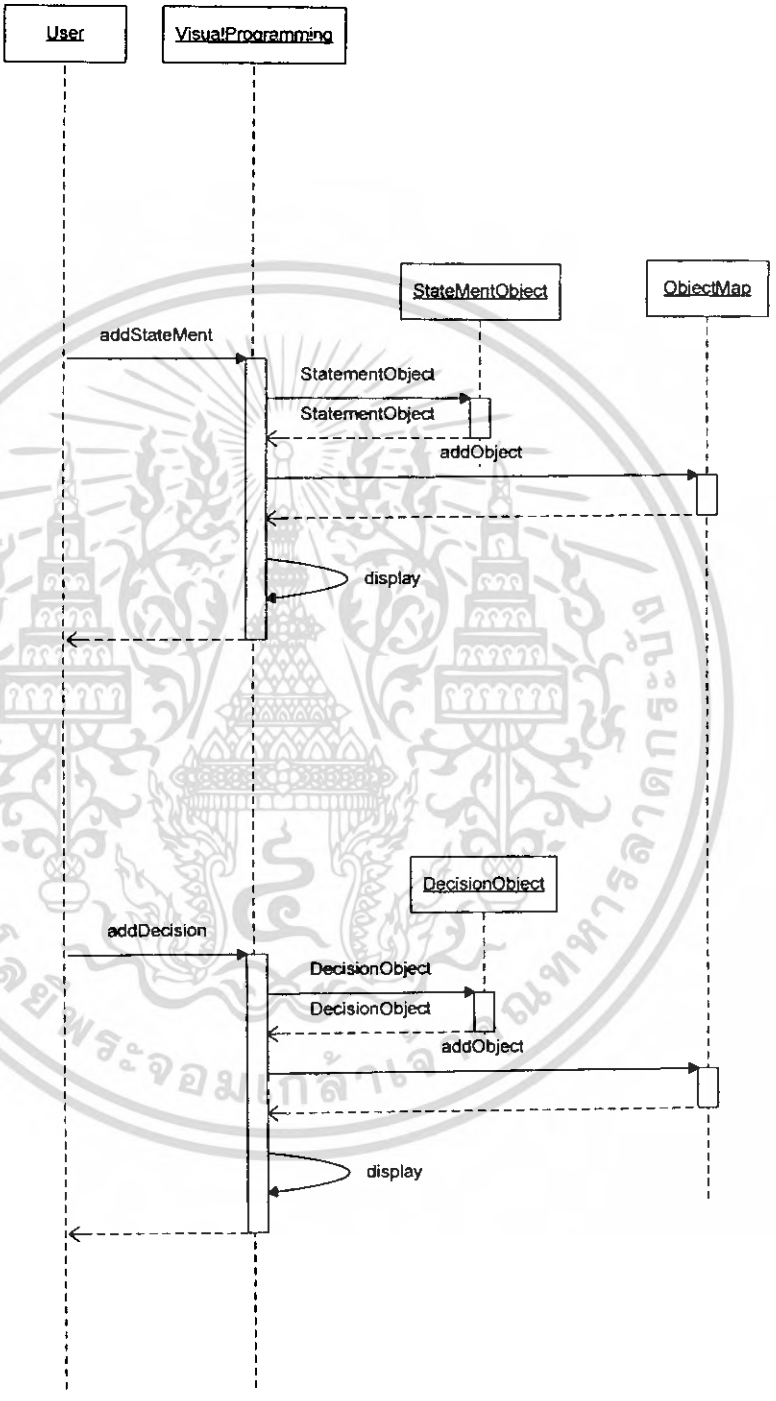
รูปที่ 3.8 Sequence diagram ของ VPF Environment (ต่อ)



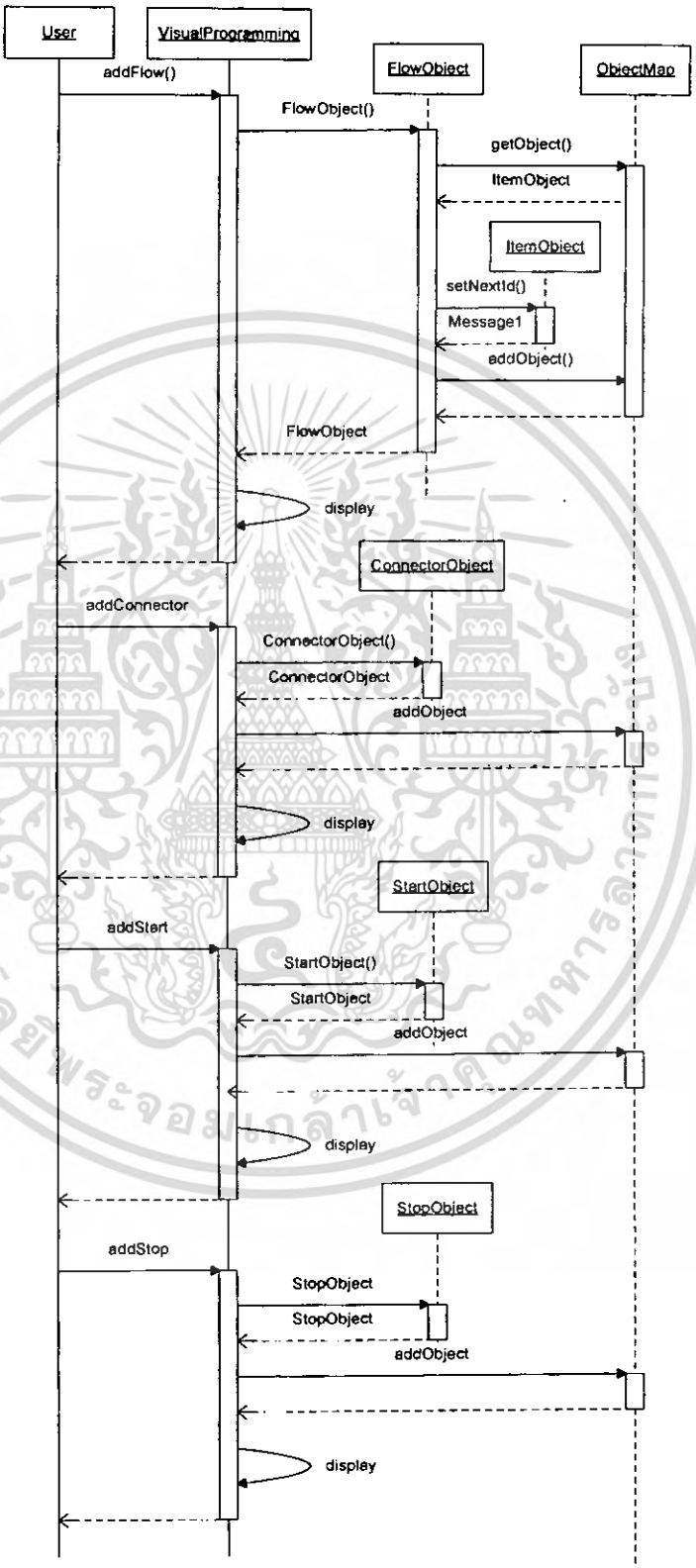
รูปที่ 3.9 Sequence diagram ของ VPF Environment (ต่อ)



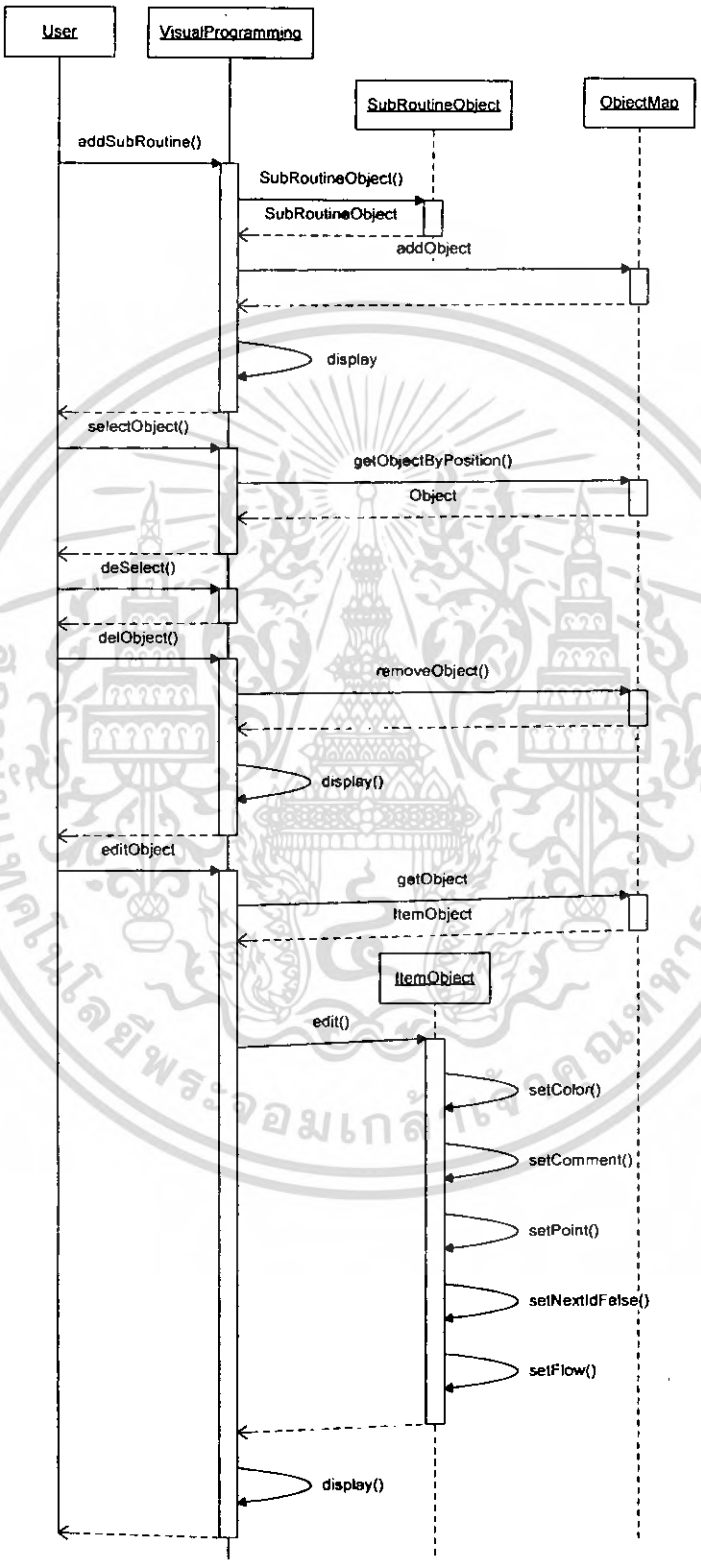
รูปที่ 3.10 Sequence diagram ของ VPF Environment (ต่อ)



รูปที่ 3.11 Sequence diagram ของ VPF Environment (ต่อ)



รูปที่ 3.12 Sequence diagram ของ VPF Environment (ต่อ)



รูปที่ 3.13 Sequence diagram ของ VPF Environment (ต่อ)

บทที่ 4

การทดลองและผลการทดลอง

4.1. บทนำ

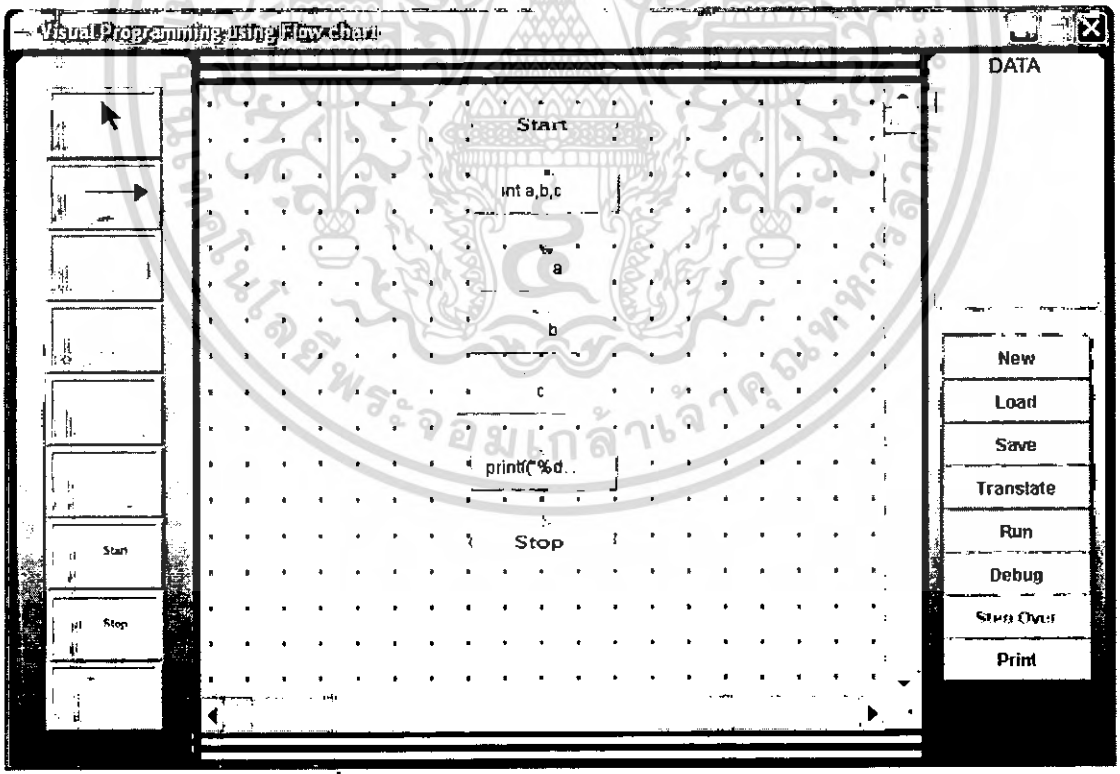
การทดลองสำหรับ Visual Programming using Flowchart Environment แบ่งเป็น 2 ส่วน ส่วนแรกจะเป็นการทดลองความถูกต้องของโปรแกรม ทั้งในส่วนของการวาด flowchart การแปลง flowchart เป็น source code ส่วนที่ 2 จะทำการทดสอบการทำงานของโปรแกรม EXE ที่ได้จากการ compile และการ debug ว่าสามารถทำงานได้อย่างถูกต้องหรือไม่

4.2. การทดสอบความถูกต้องของโปรแกรมในการแปลง Flowchart เป็น Source code

การทดลองในส่วนที่ 1 นี้เป็นการทดลองแปลง flowchart แบบต่าง ๆ เป็น source code ภาษาซี โดยมีการทดลองด้วย flowchart ลักษณะต่าง ๆ กัน 9 รูปแบบ

4.2.1. การทดลองแปลง flowchart แบบมีเส้นทางเดียว

จะทำการทดลองโดยการวาด flowchart โดยไม่มี Decision Symbol ทำให้เส้นทางจาก Start Symbol ไปยัง Stop Symbol มีเส้นทางเดียว



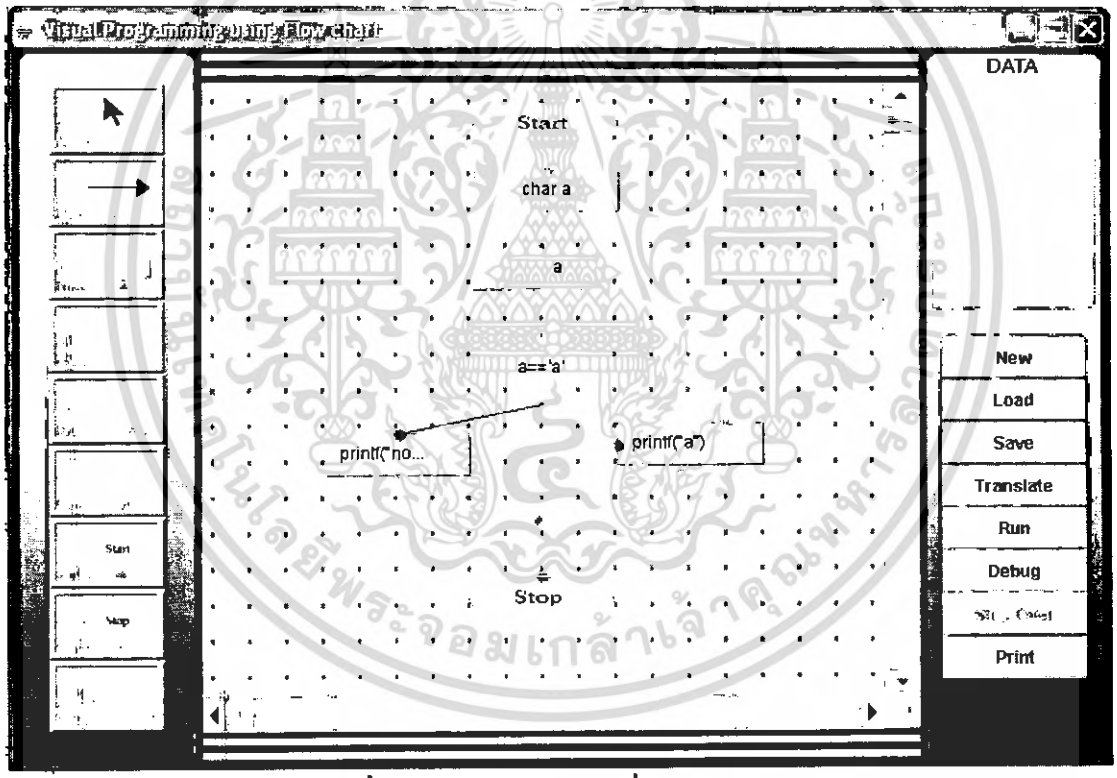
รูปที่ 4.1 flowchart แบบมีเส้นทางเดียว

```
#include <stdio.h>

void  main(int argc, char *argv[])
{

    int a;
    int b;
    int c;
    scanf("%d",&a);
    scanf("%d",&b);
    scanf("%d",&c);
    printf("%d %d %d",a,b,c);
}
```

รูปที่ 4.2 source code ที่ได้จากการแปลง flowchart ในรูปที่ 4.1

4.2.2. การทดลองแปลง flowchart แบบมีเงื่อนไข

จะทำการทดลองโดยการวาด flowchart โดยมี Decision Symbol ในลักษณะของเงื่อนไข ทำให้เส้นทางจาก Start Symbol ไปยัง Stop Symbol มีเส้นหลายเส้นทาง



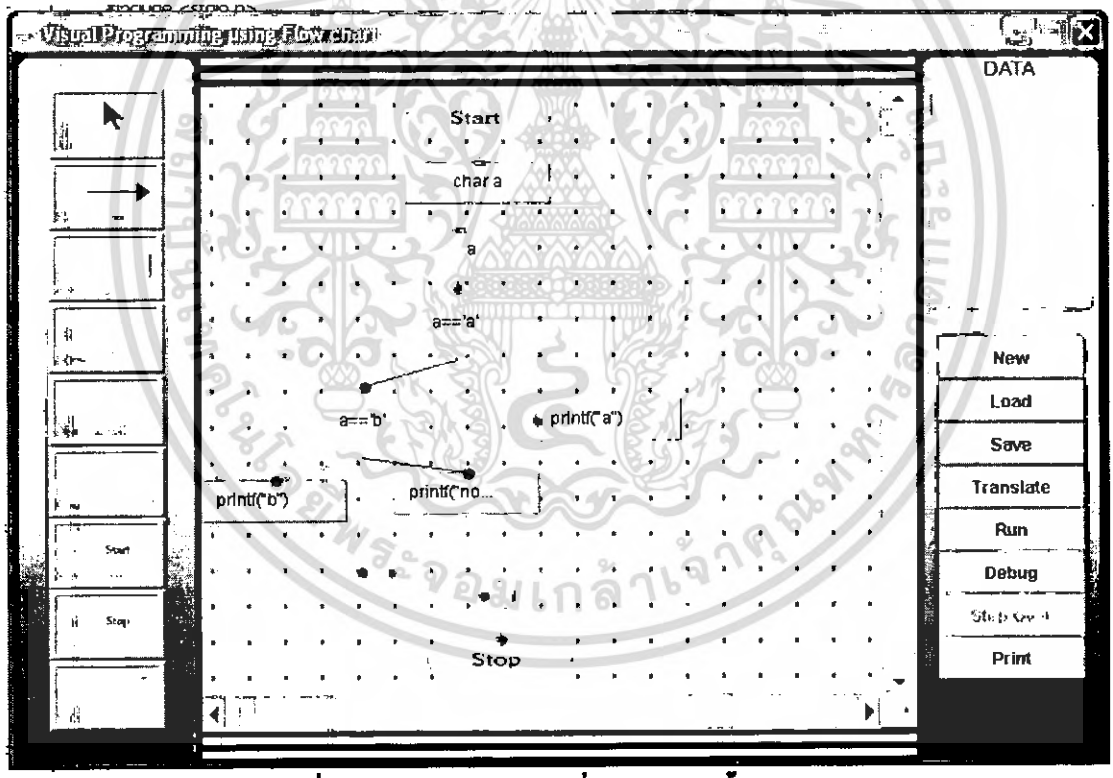
รูปที่ 4.3 flowchart แบบมีเงื่อนไข

```
#include <stdio.h>

void  main(int argc, char *argv[])
{

   char a;
   scanf("%c",&a);
   if(a=='a')
   {
      printf("a");
   }
   else
   {
      printf("not a");
   }
}
```

รูปที่ 4.4 source code ที่ได้จากการแปลง flowchart ในรูปที่ 4.3

4.2.3. การทดลองแปลง flowchart แบบมีเงื่อนไขหลายชั้น

จะทำการทดลองโดยการวาด flowchart โดยมี Decision Symbol ซ้อน Decision Symbol ในลักษณะของเงื่อนไข ทำให้เส้นทางจาก Start Symbol ไปยัง Stop Symbol มีเส้นหลายเส้นทาง



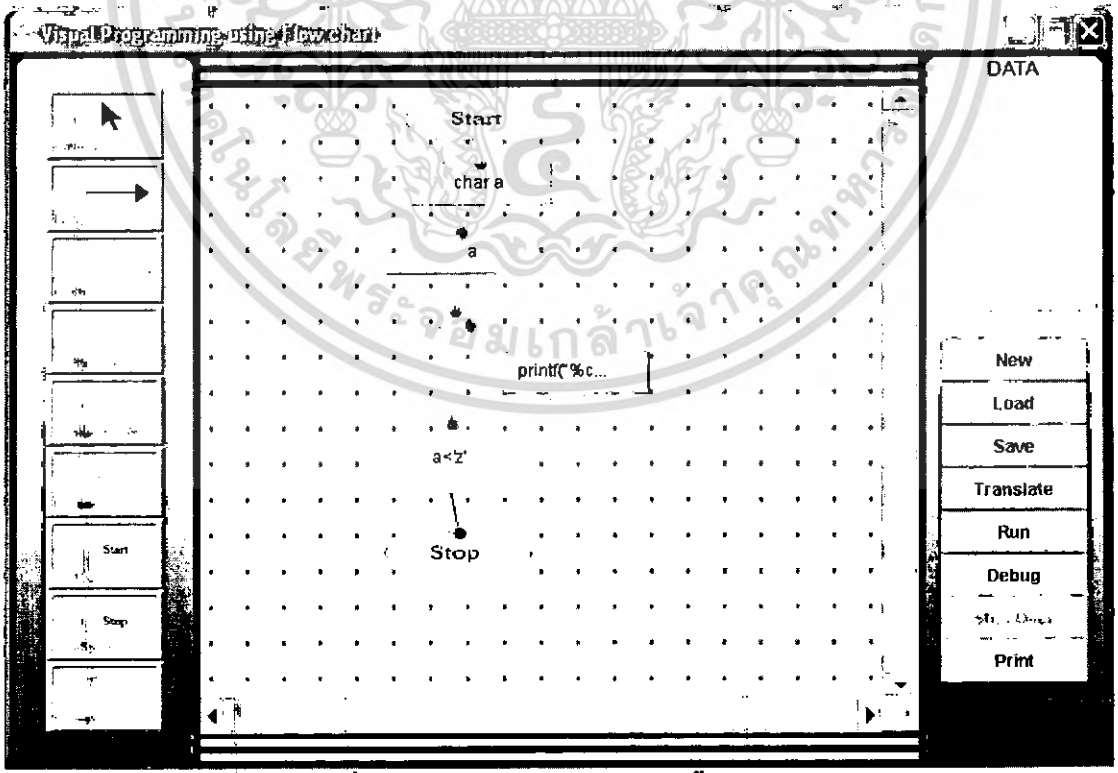
รูปที่ 4.5 flowchart แบบมีเงื่อนไขหลายชั้น

```
#include <stdio.h>

void  main(int argc, char *argv[])
{

   char a;
   scanf("%c",&a);
   if(a=='a')
   {
      printf("a");
   }
   else
   {
      if(a=='b')
      {
         printf("b");
      }
      else
      {
         printf("not a and b");
      }
   }
}
```

รูปที่ 4.6 source code ที่ได้จากการแปลง flowchart ในรูปที่ 4.5

4.2.4. การทดลองแปลง flowchart แบบมีการวนซ้ำ

จะทำการทดลองโดยการวาด flowchart โดยมี Decision Symbol ในลักษณะของการวนซ้ำ ทำให้เส้นทางจาก Start Symbol ไปยัง Stop Symbol มีเส้นหลายเส้นทาง



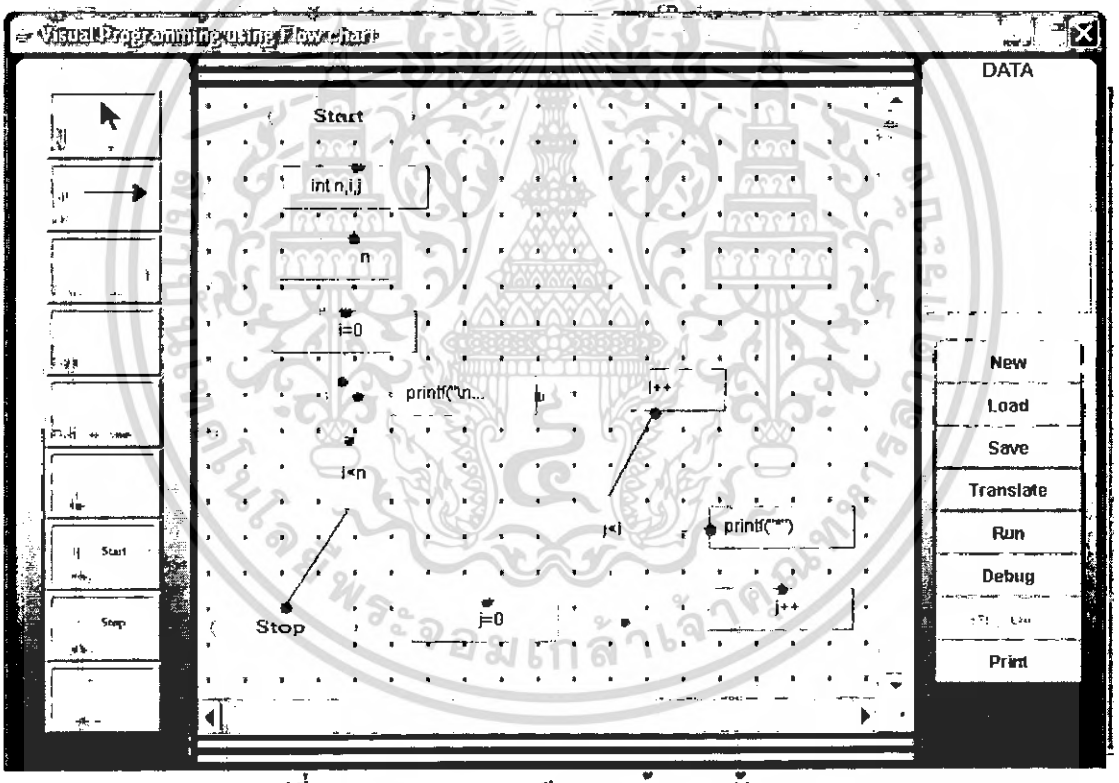
รูปที่ 4.7 flowchart แบบมีการวนซ้ำ

```
#include <stdio.h>

void  main(int argc, char *argv[])
{

   char a;
   scanf("%c",&a);
   while(a<'z')
   {
      printf("%c ",a++);
   }
}
```

รูปที่ 4.8 source code ที่ได้จากการแปลง flowchart ในรูปที่ 4.7

4.2.5. การทดลองแปลง flowchart แบบมีการวนซ้ำหลายชั้น

จะทำการทดลองโดยการวาด flowchart โดยมี Decision Symbol ซ้อน Decision Symbol ในลักษณะของการวนซ้ำ ทำให้เส้นทางจาก Start Symbol ไปยัง Stop Symbol มีเส้นหลายเส้นทาง



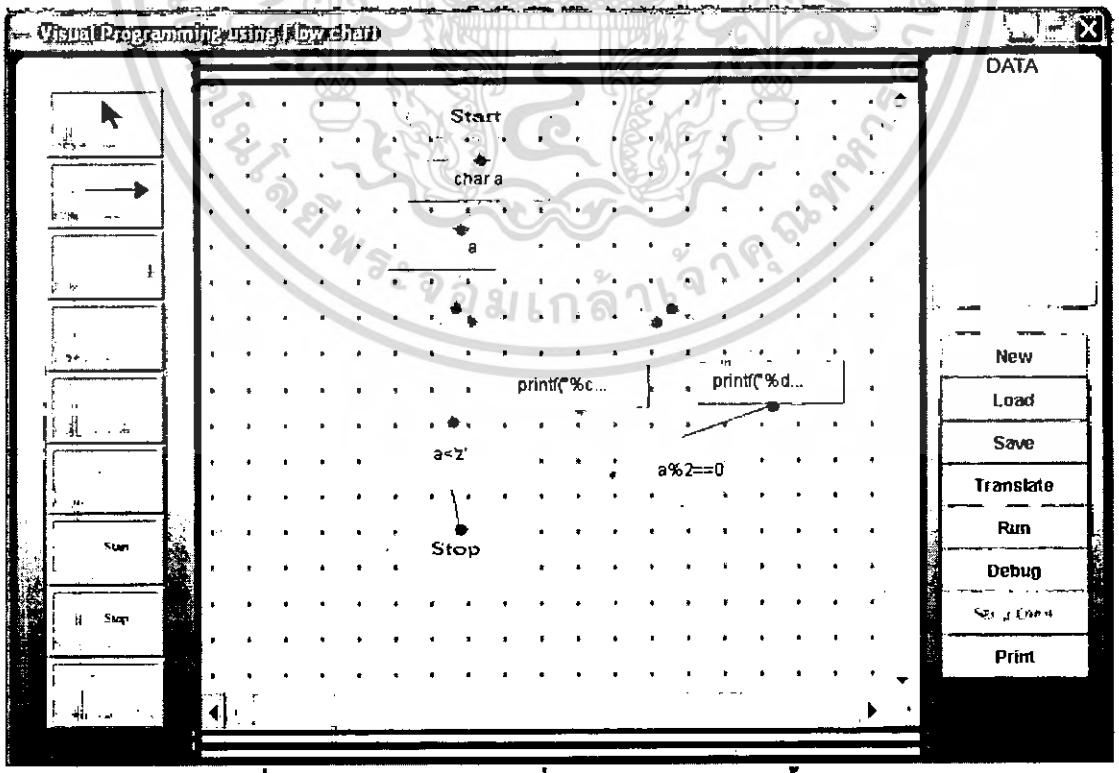
รูปที่ 4.9 flowchart แบบมีการวนซ้ำหลายชั้น

```
#include <stdio.h>

void  main(int argc, char *argv[])
{

   int n;
   int i;
   int j;
   scanf("%d",&n);
   i=0;
   while(i<n)
   {
      j=0;
      while(j<i)
      {
         printf("*");
         j++;
      }
      i++;
      printf("\n");
   }
}
```

รูปที่ 4.10 source code ที่ได้จากการแปลง flowchart ในรูปที่ 4.9

4.2.6. การทดลองแปลง flowchart แบบมีเงื่อนไขรวมกับการวนซ้ำ

จะทำการทดลองโดยการวาด flowchart โดยมี Decision Symbol ซ้อน Decision Symbol ในลักษณะของเงื่อนไข และลักษณะของการวนซ้ำ ทำให้เส้นทางจาก Start Symbol ไปยัง Stop Symbol มีเส้นหลายเส้นทาง



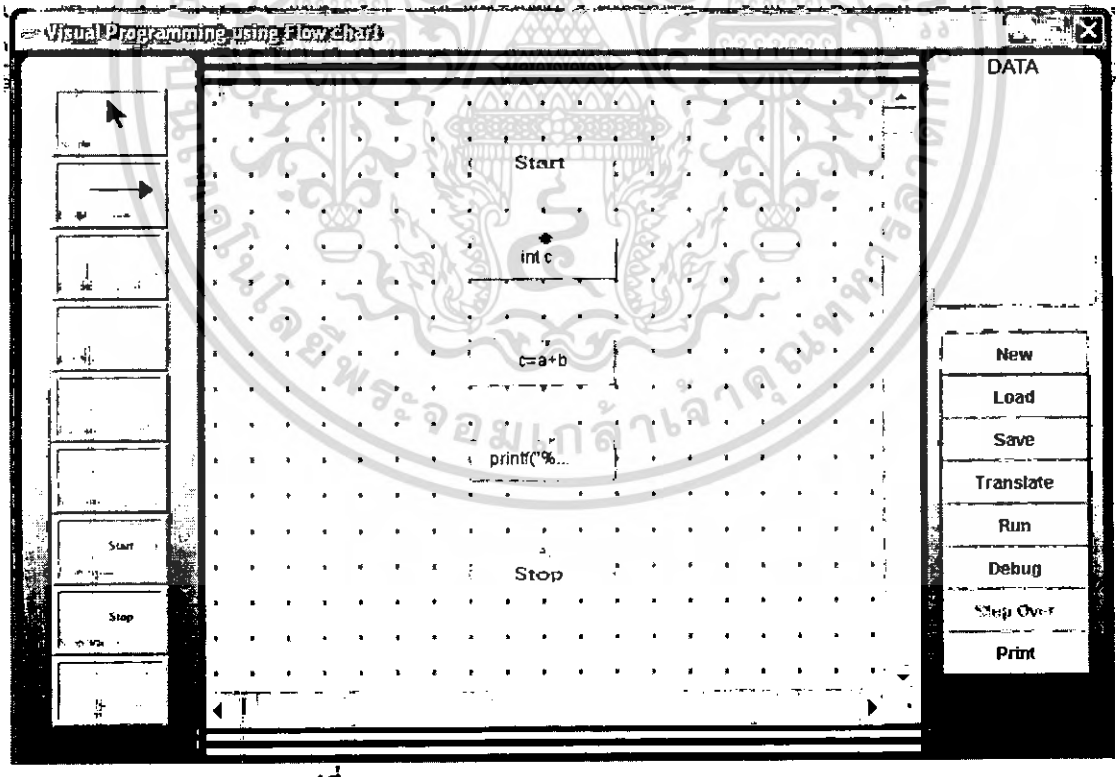
รูปที่ 4.11 flowchart แบบมีเงื่อนไขรวมกับการวนซ้ำ

```
#include <stdio.h>

void  main(int argc, char *argv[])
{

    char a;
    scanf("%c",&a);
    while(a<'z')
    {
       if(a%2==0)
       {
          printf("%c",a++);
       }
       else
       {
          printf("%d ,a++);
       }
    }
}
```

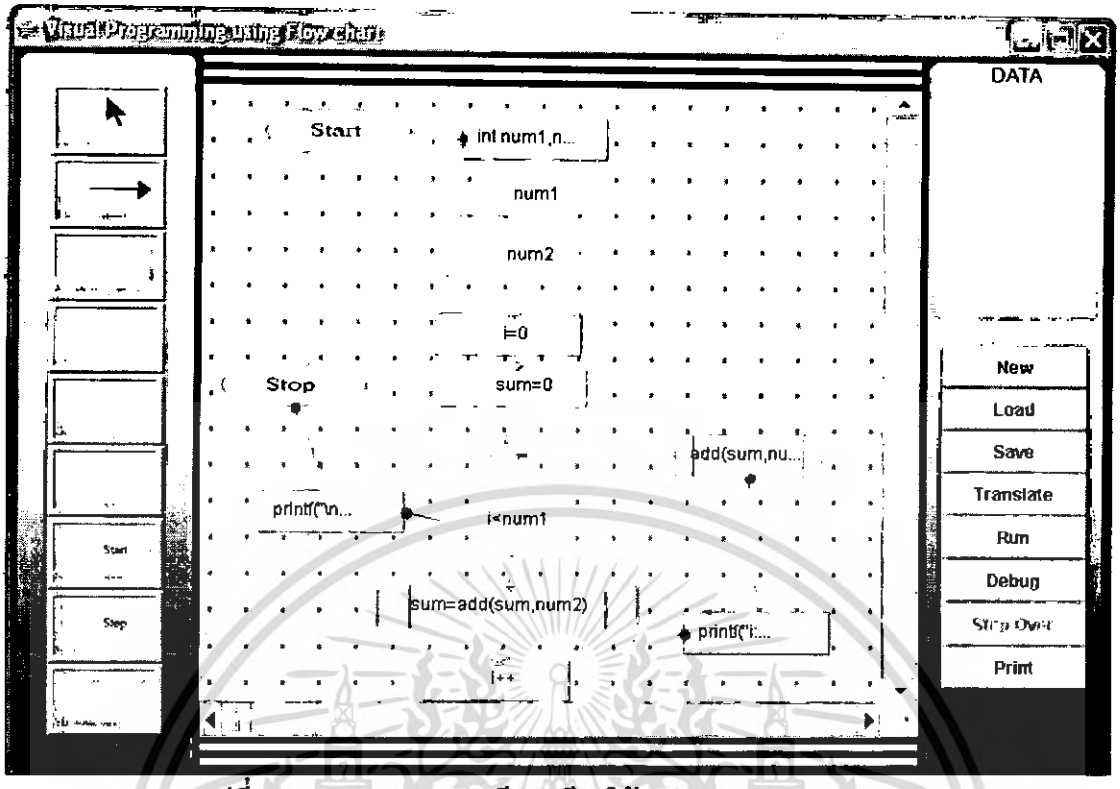
รูปที่ 4.12 source code ที่ได้จากการแปลง flowchart ในรูปที่ 4.11

4.2.7. การทดลองแปลง flowchart แบบมีการเรียกใช้งาน subroutine

จะทำการทดลองโดยการวาด flowchart โดยมี Subroutine Symbol เรียกใช้ flowchart ที่เคยวาดไว้แล้วมาทำงาน



รูปที่ 4.13 flowchart ของ function add



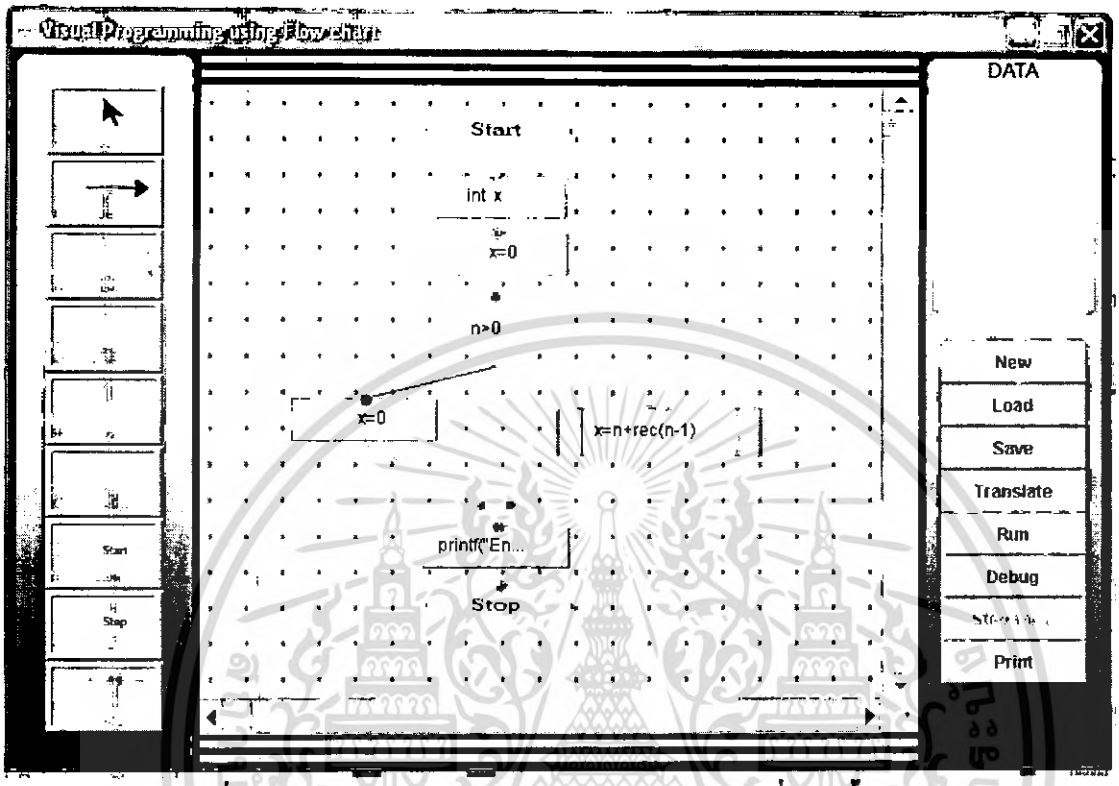
รูปที่ 4.14 flowchart แบบมีการเรียกใช้งาน subroutine

```
#include <stdio.h>

int add(int a,int b);

int  main(int argc, char *argv[])
{

    int num1;
    int num2;
    int i;
    int sum;
    scanf("%d",&num1);
    scanf("%d",&num2);
    i=0;
    sum=0;
    while(i<num1)
    {
       sum=add(sum,num2);
       i++;
       printf("i:%d",i);
       add(sum,num2);
    }
    printf("\nsum=%d",sum);
    return 0;
}
int add(int a,int b)
{

    int c;
    c=a+b;
    printf("%d",c);
    return c;
}
```

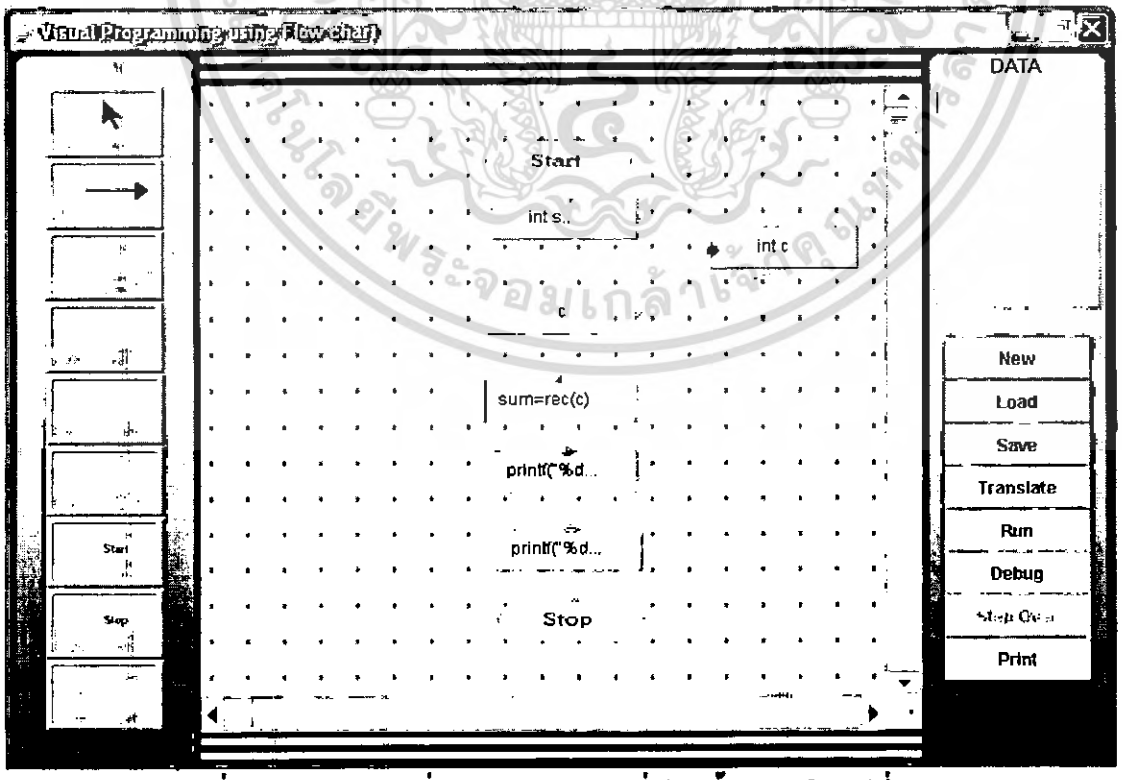
รูปที่ 4.15 source code ที่ได้จากการแปลง flowchart ในรูปที่ 4.14

4.2.8. การทดสอบการเขียน Flowchart แบบ recursive

จะทำการทดลองโดยการวาด flowchart 2 รูป รูปหนึ่งเป็นโปรแกรมหลัก และอีกรูปหนึ่งจะเป็นโปรแกรมที่ทำงานแบบเรียกซ้ำตัวเอง



รูปที่ 4.16 flowchart แสดงการทำงานของ function ที่เรียกซ้ำตัวเอง



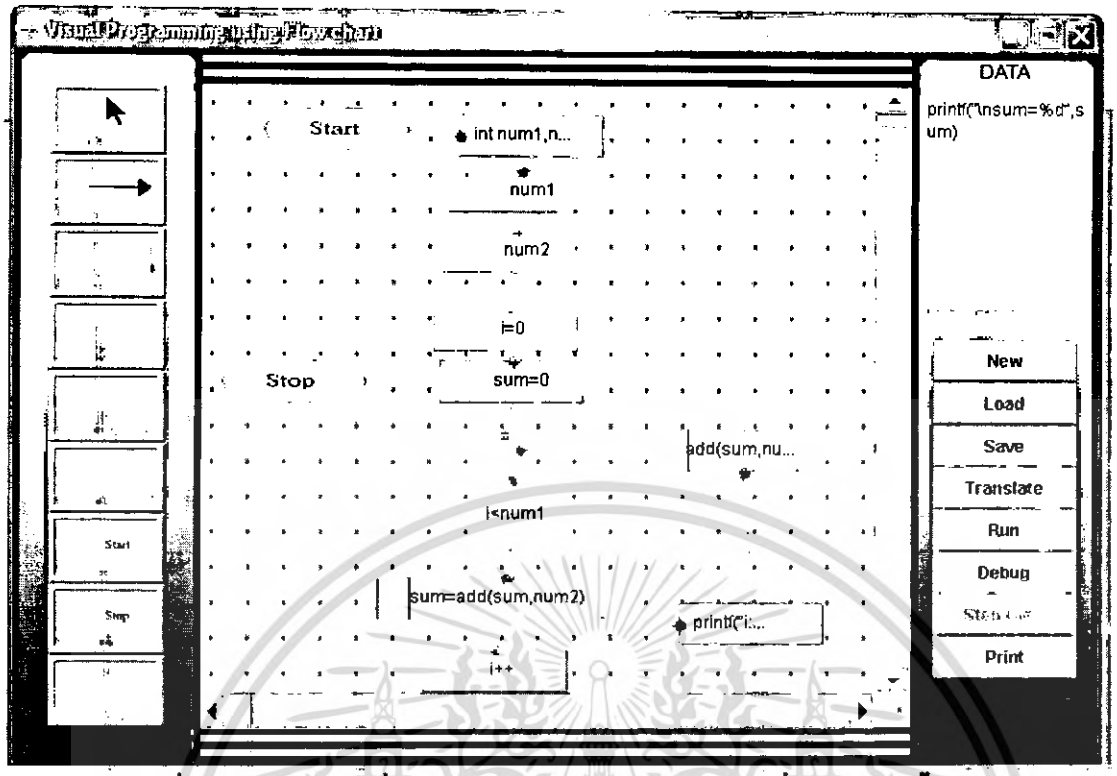
รูปที่ 4.17 flowchart ที่เรียกใช้ function ที่เรียกซ้ำตัวเองในรูปที่ 4.16

```
#include <stdio.h>

int rec(int n);

void  main(int argc, char *argv[])
{

   int sum;
   int c;
   scanf("%d",&c);
   sum=rec(c);
   printf("%d",sum);
   printf("%d",c);
}
int rec(int n)
{

   int x;
   x=0;
   if(n>0)
   {
      x=n+rec(n-1);
   }
   else
   {
      x=0;
   }
   printf("End Rountin");
   return x;
}
```

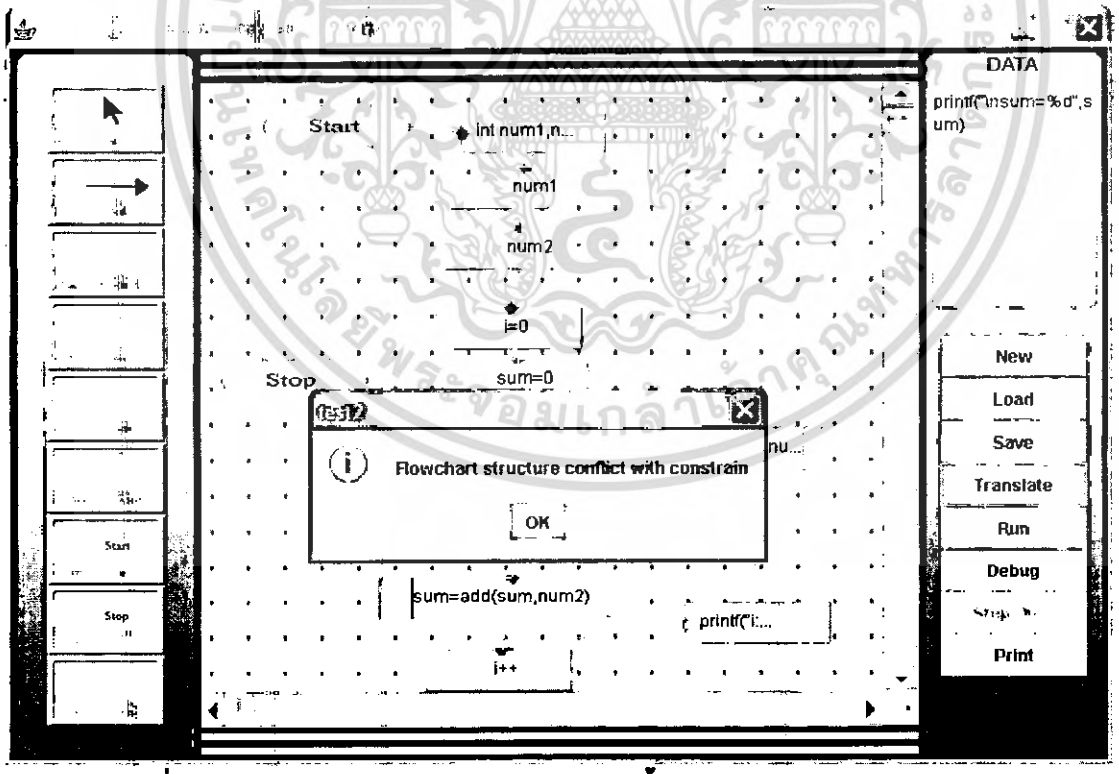
รูปที่ 4.18 source code ที่ได้จากการแปลง flowchart ในรูปที่ 4.17

4.2.9. การทดลองแปลง flowchart แบบที่ flowchart ไม่สมบูรณ์

จะทำการทดลองโดยการวาด flowchart ที่ไม่สมบูรณ์แล้วทำการเรียกใช้งานคำสั่งแปลง flowchart เป็น source code



รูปที่ 4.19 flowchart ที่ไม่สมบูรณ์คือไม่มีเส้นทางจากจุดเริ่มต้นไปจุดสิ้นสุด



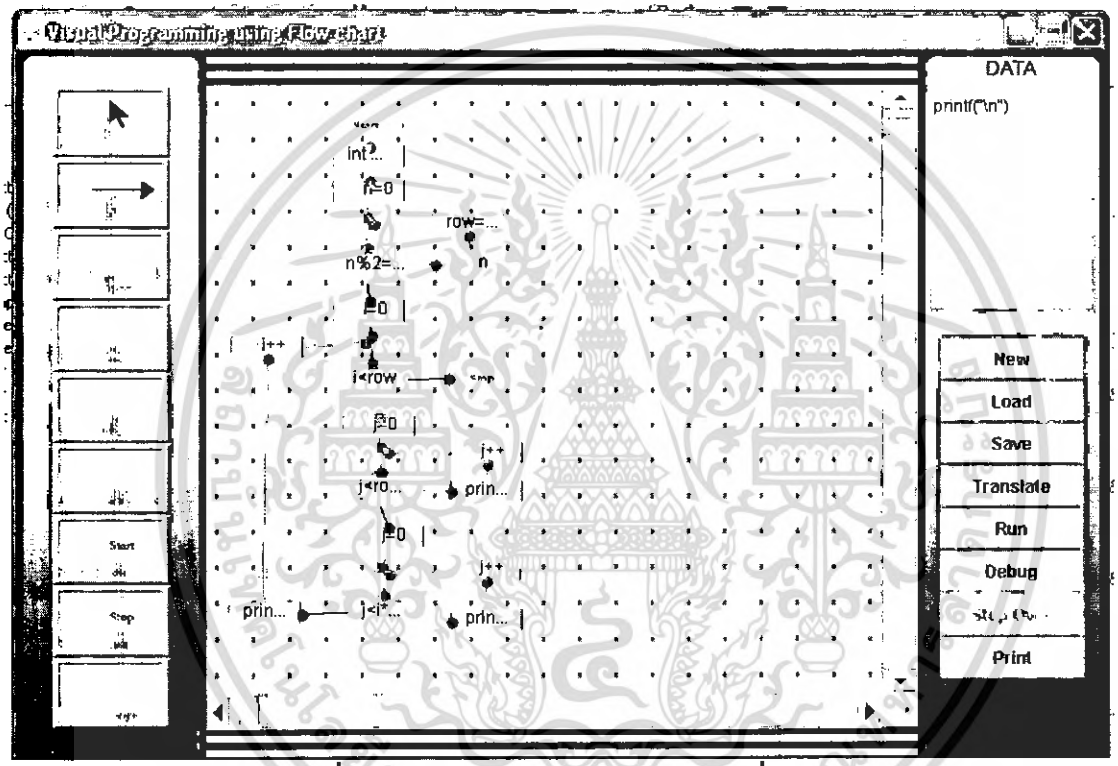
รูปที่ 4.20 ผลลัพธ์จากการแปลง flowchart จะขึ้นข้อความแสดงความผิดพลาด

4.3. การทดสอบความถูกต้องของโปรแกรมในการทำงานของโปรแกรม EXE และการ Debug

การทดลองในส่วนที่ 2 นี้เป็นการทดลองนำFlowchart มาcompile โดยใช้ C Compiler ของ Borland มา compile จากนั้นทดสอบความถูกต้องของโปรแกรมโดยการเรียกใช้โปรแกรม EXE ตลอดจนทดสอบความถูกต้องในการ debug ด้วย

4.3.1. การทดสอบความถูกต้องในการทำงานของโปรแกรม EXE ที่ได้จากการ Compile

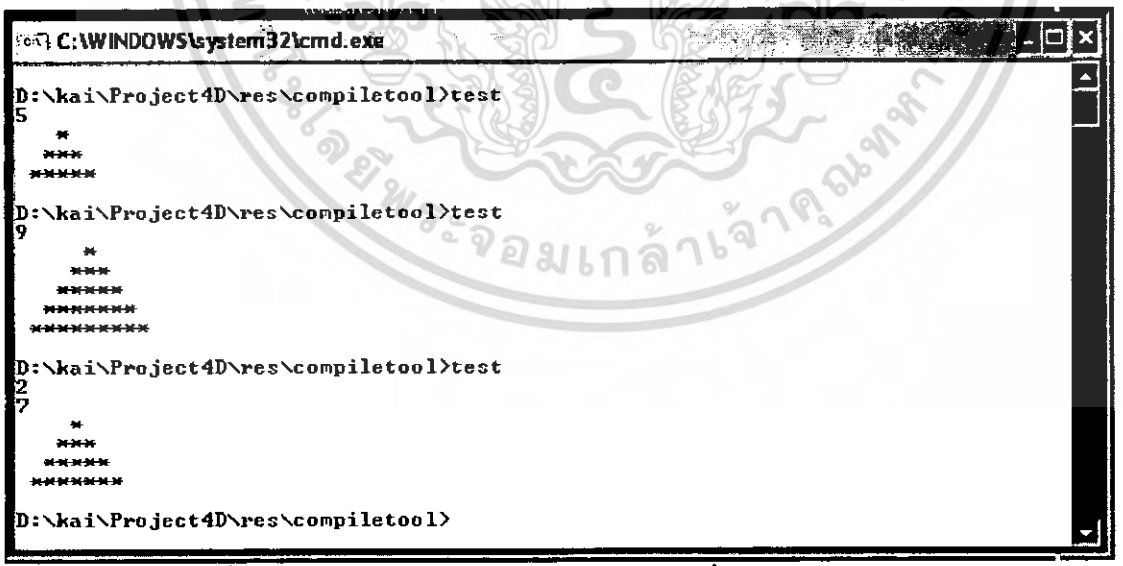
ในการทดลองในหัวข้อนี้จะทำการเขียน Flowchart ที่ทำการวาดรูปปีรามิดด้วยเครื่องหมาย "*" โดยรับค่าจำนวนตัวอักษรที่เป็นฐานเป็นจำนวนเต็มเลขคี่เท่านั้น หากใส่ค่าเป็นจำนวนเต็มคู่ให้รับค่าใหม่



รูปที่ 4.21 Flowchart สำหรับการทดลองที่ 4.3.1

```
#include <stdio.h>

void  main(int argc, char *argv[])
{

   int n;
   int i;
   int j;
   int row;
   n=0;
   while(n%2==0)
   {
      scanf("%d",&n);
      row=n/2+1;
   }
   i=0;
   while(i<row)
   {
      j=0;
      while(j<row-i)
      {
         printf(" ");
         j++;
      }
      j=0;
      while(j<i*2+1)
      {
         printf("*");
         j++;
      }
      printf("\n");
      i++;
   }
}
```

รูปที่ 4.22 Source code ที่ได้จากการแปลง Flowchart ในรูปที่ 4.21

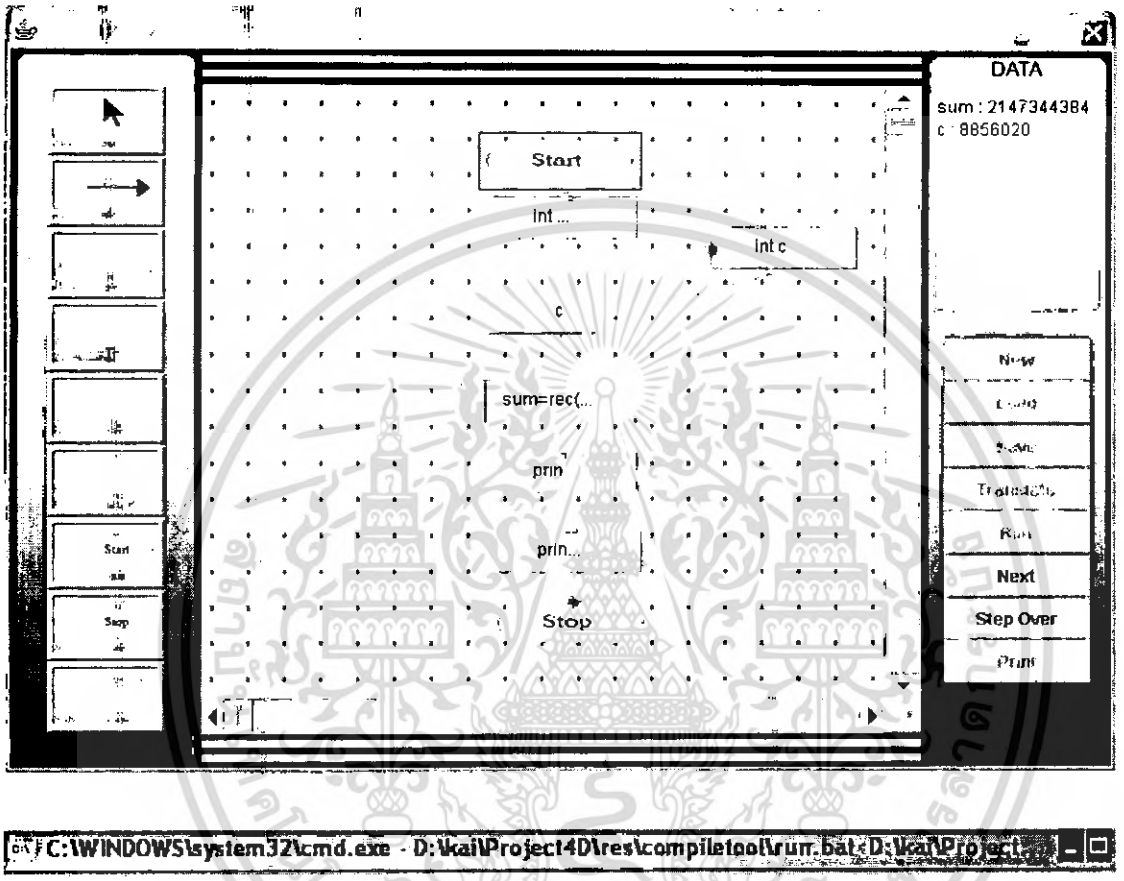


รูปที่ 4.23 รูปแสดงการทำงานโปรแกรม EXE ที่ได้จากการ compile

จากการทดลองพบว่าได้ผลลัพธ์ที่ถูกต้อง

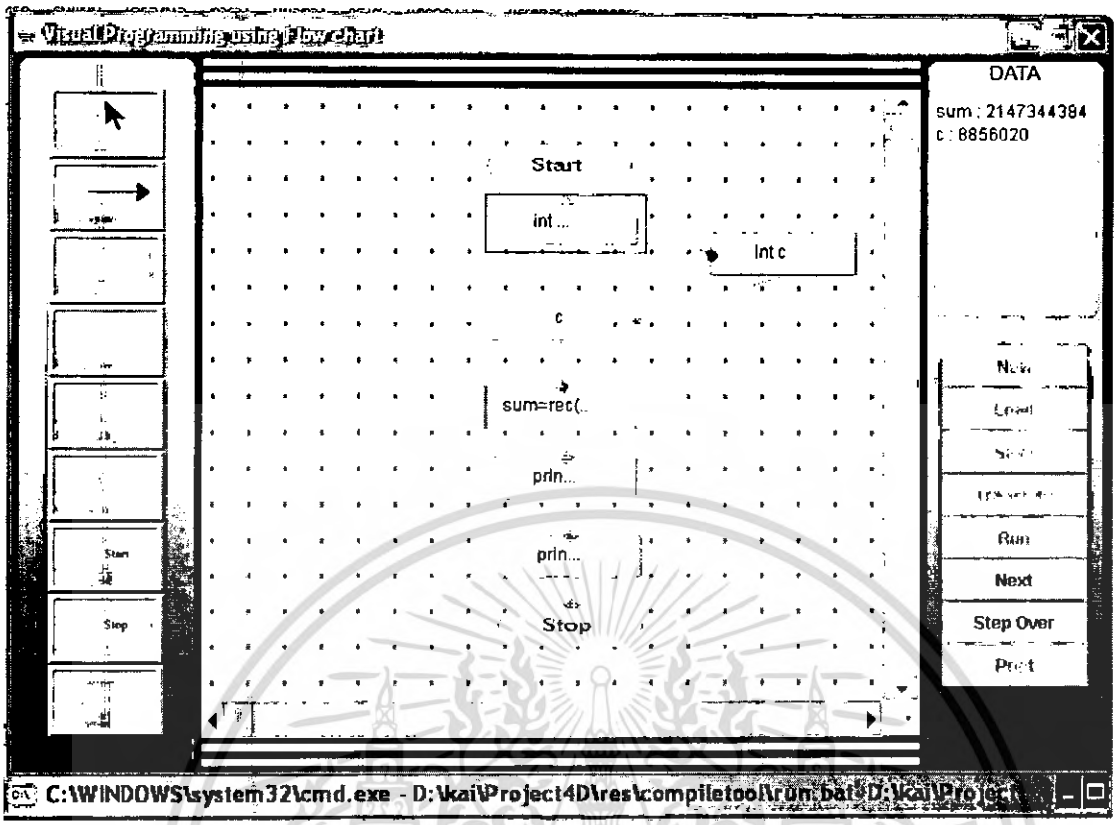
4.3.2. การทดสอบความถูกต้องของการ Debug

ในการทดลองจะทำการ debug การทำงานของโปรแกรมในการทดลองที่ 4.2.8 (การทดลองการทำงานที่มีการเรียกตัวเอง) โดยจะทดสอบทั้งการ debug แบบ step over และ step into อีกทั้งจะทำการทดสอบการดูค่าตัวแปรในขอบเขตของการทำงานที่ทำอยู่



รูปที่ 4.24 รูปแสดงการเริ่มต้น debug : ภาพบนแสดง VPF Environment ขณะ debug ภาพล่างแสดง console ที่ทำงานโปรแกรมที่ได้จากการ compile ที่ละคำสั่ง

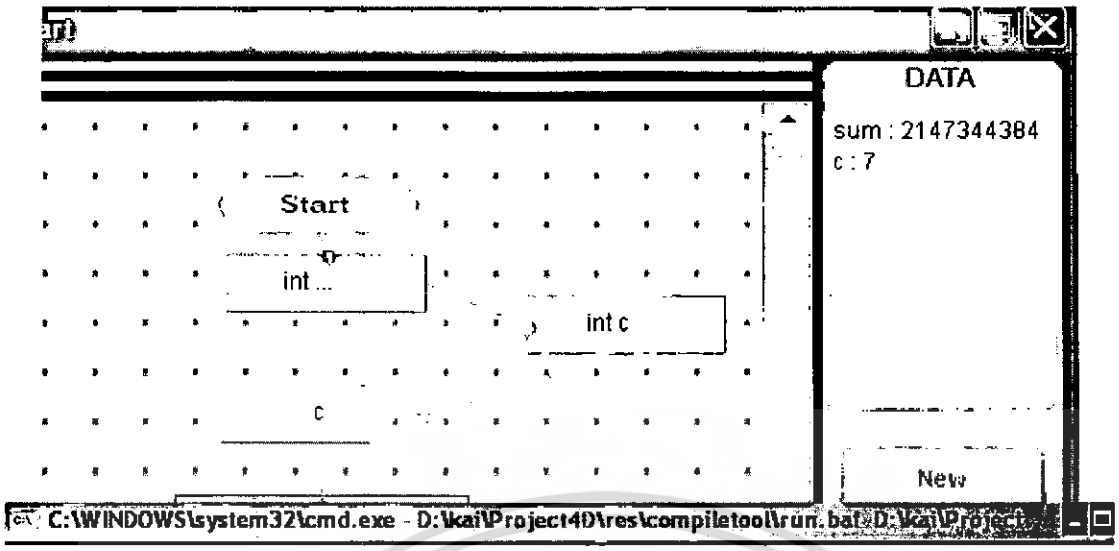
จากรูปที่ 4.24 จะสังเกตได้ว่า Flowchart นี้มีการประกาศตัวแปร 2 ตัวคือ sum และ c โดยเมื่อเริ่มทำงานยังไม่มีคำสั่งในการกำหนดค่าเริ่มต้นให้กับตัวแปร ดังนั้นค่าของตัวแปรจึงเป็นค่าเดิมที่ memory เก็บไว้ เมื่อกดปุ่ม Next สี่เหลี่ยมสีแดงจะเลื่อนไปที่คำสั่งถัดไปที่จะถูกประมวลผล ดังรูปที่ 4.25




```
D:\kai\Project4D\res\Visual Programming>d:
D:\kai\Project4D\res\Visual Programming>cd D:\kai\Project4D\res\compiletool\
D:\kai\Project4D\res\compiletool>Vdebug.exe
```

รูปที่ 4.25 รูปแสดงการเริ่มต้น debug : ภาพบนแสดง VPF Environment ขณะ debug ภาพล่างแสดง console ที่ทำงานโปรแกรมที่ได้จากการ compile ทีละคำสั่ง

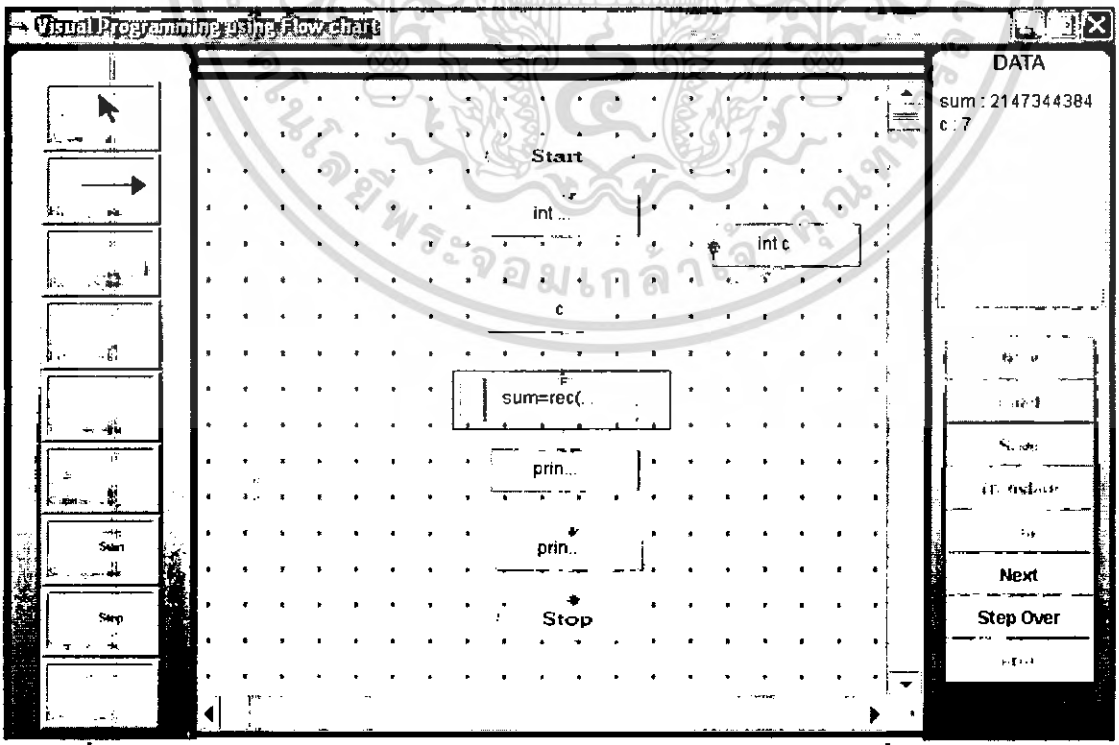
เมื่อถึงคำสั่งที่ต้องรอรับค่า หากผู้ใช้งานกดปุ่ม Next จะไม่มีผลใดๆเกิดขึ้น เนื่องจาก VPF Environment จะหยุดรอการทำงานของโปรแกรม EXE รับค่าจาก Keyboard และเมื่อมีการรับค่าจาก Keyboard ซึ่งเป็นการกำหนดค่าให้กับตัวแปรแล้ว เมื่อผู้ใช้งานกดปุ่ม Next ค่าของตัวแปร c จะเปลี่ยนไปตามค่าที่ใส่ผ่านทาง console ดังรูปที่ 4.26



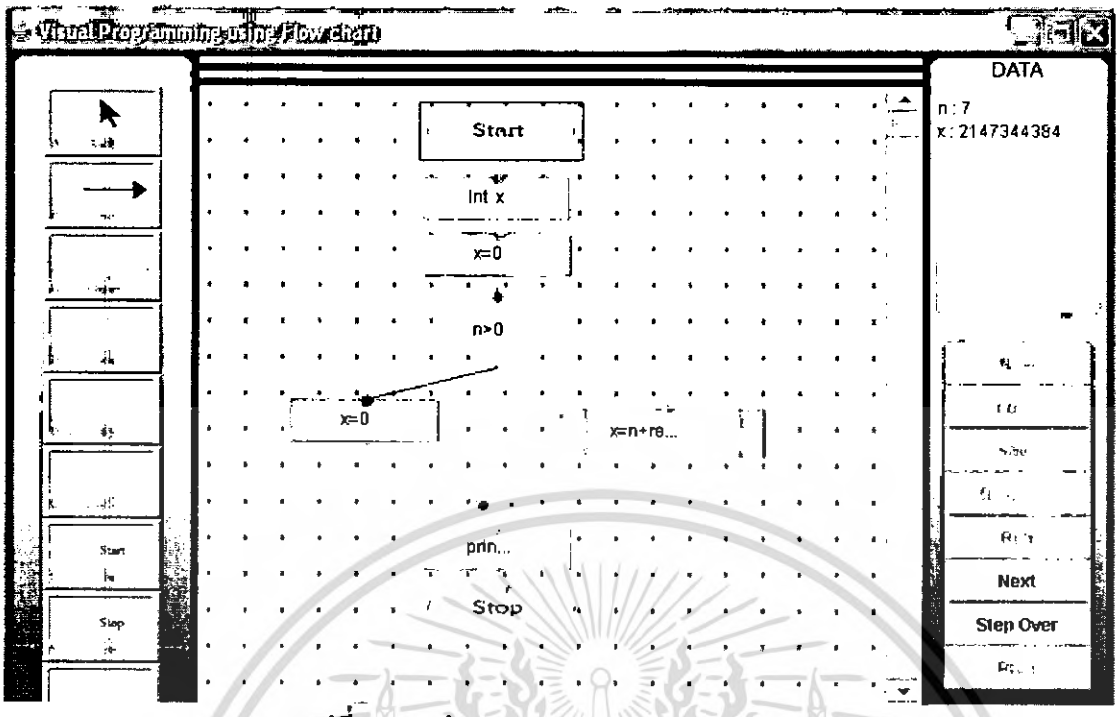

```
D:\kai\Project4D\res\Visual Programming>d:

D:\kai\Project4D\res\Visual Programming>cd D:\kai\Project4D\res\compiletool\

D:\kai\Project4D\res\compiletool>Vdebug.exe
7
```

รูปที่ 4.26 รูปแสดงการ debug หลังจากมีการรับค่าจาก Keyboard

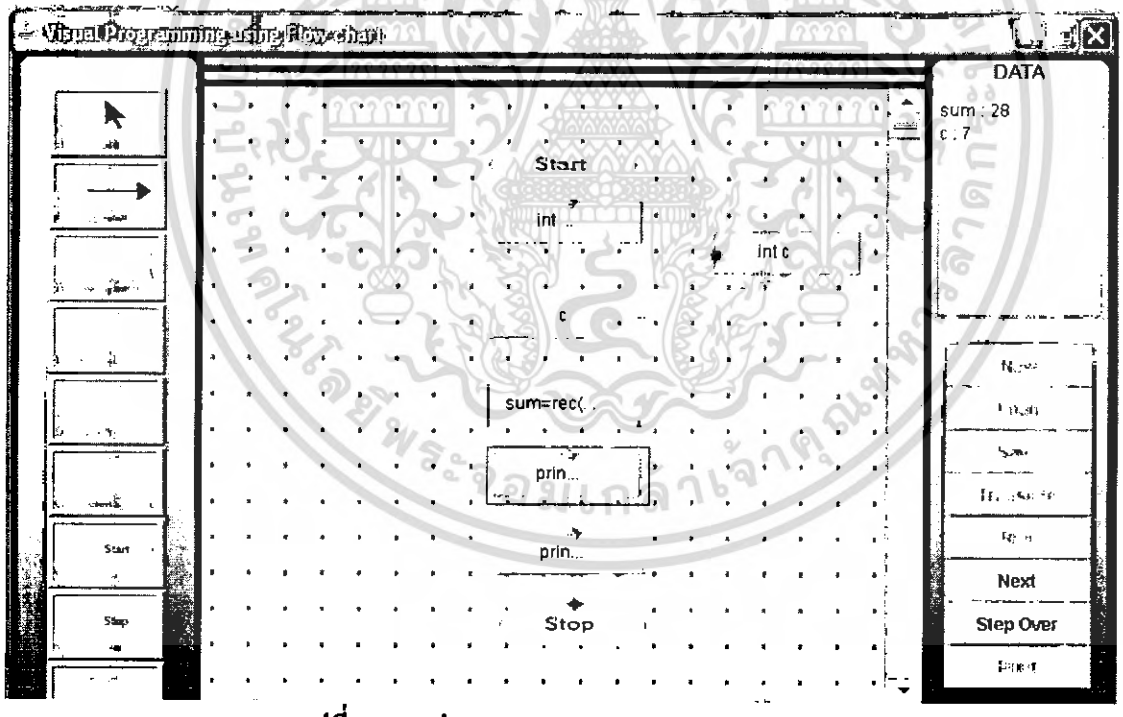
หากพบเครื่องหมาย Subroutine Symbol แล้วผู้ใช้กดปุ่ม Next จะเป็นการ Step Into เพื่อเข้าไปดูการทำงานภายใน Function หากผู้ใช้กดปุ่ม Step Over จะเป็นการข้ามการหยุดรอคำสั่งภายใน Function จากรูปที่ 4.27 หากผู้ใช้กดปุ่ม Next โปรแกรมจะทำการแสดงการทำงานภายใน Function ดังรูปที่ 4.28 หากผู้ใช้กดปุ่ม Step Over โปรแกรมจะไม่หยุดรอการทำงานภายใน Function และจะหยุดรอดังรูป 4.29



รูปที่ 4.27 รูปแสดงการ debug ขณะหยุดรอการกด Next หรือ Step Over เพื่อเรียก Function



รูปที่ 4.28 รูปแสดงการ debug แบบ Step Into



รูปที่ 4.29 รูปแสดงการ debug แบบ Step Over

บทที่ 5
บทวิจารณ์และสรุป

5.1. ข้อสรุป

Visual Programming using Flowchart Environment เป็นเครื่องมือที่ช่วยลดความซับซ้อนของการพัฒนาโปรแกรมคอมพิวเตอร์ จึงทำให้เป็นเครื่องมือที่เหมาะกับผู้เริ่มต้นที่ใช้ในศึกษาการพัฒนาโปรแกรมคอมพิวเตอร์ ซึ่งสามารถสร้างโปรแกรมได้ทุกรูปแบบโดยการวาด flowchart นอกจากจะทำให้การพัฒนาโปรแกรมเข้าใจง่ายแล้ว ยังมีส่วนของการ debug ที่มีประสิทธิภาพอีกด้วย ดังนั้นเครื่องมือนี้ไม่เพียงเหมาะสำหรับผู้เริ่มต้นเท่านั้น แต่ยังสามารถนำไปใช้ไปสื่อการสอนตลอดจนเป็นเครื่องมือที่ใช้อธิบาย algorithm ที่มีประสิทธิภาพได้อีกด้วย

5.2. วิจารณ์สิ่งที่ได้รับจากโครงงาน

ในการพัฒนาเครื่องมือที่เหมาะสมสำหรับผู้เริ่มต้นนั้น จะต้องเข้าใจถึงอุปสรรค์ของผู้เริ่มต้นในการพัฒนาโปรแกรม จากการศึกษาพบว่าวิธีการที่เหมาะสมในการอธิบาย Algorithm คือการเขียน Flowchart เพื่ออธิบายแนวคิดออกมาเป็นรูปธรรมแต่ไม่สามารถ Compile เป็นโปรแกรม EXE ที่ผู้เริ่มต้นต้องการได้ แต่ Visual Programming using Flowchart เป็นเครื่องมือที่สามารถแปลง Flowchart เป็น Source code และนำไป compile เป็นโปรแกรม EXE ได้ จึงเป็นเครื่องมือที่เหมาะสมสำหรับผู้เริ่มต้นในการศึกษาการพัฒนาโปรแกรมคอมพิวเตอร์เบื้องต้น เพราะVPF Environment มีเครื่องมือที่ใช้ในการวาด Flowchart การแปลง Flowchart เป็น Source code การ Compile และ Debug แบบเป็นรูปธรรม กล่าวคือมีรูปภาพแสดงคำสั่งที่กำลังทำงานอยู่ ซึ่งทำให้ผู้ใช้งานไม่ต้องจินตนาการเหมือนเช่นการ Debug ในการเขียนโปรแกรมแบบดั่งเดิม VPF ไม่เพียงเหมาะสมสำหรับผู้เริ่มต้นที่จะศึกษาการพัฒนาโปรแกรมคอมพิวเตอร์เท่านั้นแต่ยังสามารถนำไปใช้ในการอธิบาย Algorithm ที่ซับซ้อนได้อย่างมีประสิทธิภาพ

ข้อจำกัดของ VPF นั้นจะถูกจำกัดโดยข้อจำกัดของ Flowchart ที่เป็นการเขียนโปรแกรมแบบ Structural Programming และไม่เหมาะสมหากนำไปใช้ในการพัฒนาโปรแกรมขนาดใหญ่ ถึงแม้จะสามารถแบ่งเป็นส่วนย่อยๆได้ก็ตาม

5.3. ปัญหาอุปสรรค์และแนวทางแก้ไข

- เนื่องจาก VPF Environment ถูกพัฒนาโดยใช้ภาษา JAVA แต่โปรแกรมที่ได้จากการ compile โดย VPF Environment นั้นเป็นภาษา C จึงทำให้มีปัญหาในการติดต่อกันระหว่าง Process จึงจำเป็นต้องจำลองการเขียน File แทน Mailbox

- ภาษาคอมพิวเตอร์ที่มีอยู่ในปัจจุบันมีมากกว่า 1000 ภาษา ดังนั้นจึงต้องเลือก 1 ภาษาในการแปลง Flowchart จากการสอบถามจากผู้เริ่มต้นส่วนใหญ่เริ่มศึกษาการพัฒนาโปรแกรมจากภาษา C

- สัญลักษณ์ในการเขียน Flowchart มีมากมาย ดังนั้นจึงต้องเลือกเฉพาะสัญลักษณ์ที่เหมาะสำหรับผู้เริ่มต้น เพื่อป้องกันความสับสนของผู้เริ่มต้นจึงใช้สัญลักษณ์ I/O สำหรับรับค่า (Input) เท่านั้น

5.4. แนวทางการพัฒนาต่อ

- พัฒนาเครื่องมือให้สามารถพัฒนาโปรแกรมเชิงวัตถุได้
- พัฒนาในส่วนของชุดคำสั่งให้สามารถรองรับภาษาธรรมชาติ
- พัฒนาในส่วนการแปลง Source code เป็น Flowchart
- เพิ่มเติมส่วนที่ใช้ในการแจ้งเตือนความผิดพลาดที่เกิดจากการผิดไวยากรณ์

เอกสารอ้างอิง

เอกสารอ้างอิงที่เป็นประชุมวิชาการ

[1] Kanis Charntaweekhun, Somkiat Wangsiripitak, "Visual Programming using Flowchart", 2006

[2] Geoffrey G. Roy, Joel Kelso, Craig Standing, "Towards a Visual Programming Environment for Software Development"

[3] Min Hu, "A Case Study in Teaching Adult Students Computer Programming", Proceedings of the 16th Annual NACCQ Palmerston North New Zealand July , 2003, pp. 287-291

[4] Marian G. Williams, William A. Ledder, "Visual Prgramming Labs for Introducing Computer Science Concepts", Frontiers in Education Conference ,1993 ,pp. 797 - 802

เอกสารอ้างอิงที่เป็นหนังสือ

[5] Gary Nutt 2004 **Operating system** Pearson Addison Wesley ,Boston

[6] สมชาย กิตติชัยกุลกิจ **เรื่องพัฒนาซอฟต์แวร์มีแค่นี้** สำนักพิมพ์ ส.ส.ท. กรุงเทพฯ